ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
«СЕВЕРО-ЗАПАДНЫЙ ГОСУДАРСТВЕННЫЙ ЗАОЧНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем обработки информации и управления

ОРГАНИЗАЦИЯ ЭВМ и СИСТЕМ

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС

Институт информационных систем и вычислительной техники

Специальности:
230101.65 – вычислительные машины, комплексы, системы и сети
230102.65 – автоматизированные системы обработки информации и управления
230105.65 – программное обеспечение вычислительной техники и автоматизированных систем

Направления подготовки бакалавра
230100.62 – информатика и вычислительная техника

Санкт-Петербург
Издательство СЗТУ
2009

Утверждено редакционно-издательским советом университета
УДК 681.3

Организация ЭВМ и систем: учебно-методический комплекс (информация о дисциплине, рабочие учебные материалы, информационные ресурсы дисциплины, блок контроля освоения дисциплины) / сост.: М.В. Копейкин, В.В. Спиридонов, Е.О. Шумова. - СПб.: Изд-во СЗТУ, 2009. – 187 с.

Учебно-методический комплекс разработан в соответствии с государственными образовательными стандартами высшего профессионального образования.

Дисциплина посвящена изучению вопросов организации ЭВМ и систем. В основных ее разделах изучаются принципы построения, функционирования и оценки характеристик ЭВМ и систем, отдельных их устройств и блоков.

Рассмотрено на заседании кафедры автоматизированных систем обработки информации и управления 10 декабря 2008 г.; одобрено методическим советом института информационных систем и вычислительной техники 24 декабря 2008 г.

Рецензент: Г.И. Анкудинов, д-р техн. наук, проф. кафедры информационных систем и технологий СЗТУ.

Составители: М.В. Копейкин, канд. техн. наук, доц.,
В.В. Спиридонов, канд. техн. наук, доц.,
Е.О. Шумова, доц.

1. Информация о дисциплине

1.1. Предисловие

Дисциплина «Организация ЭВМ и систем» изучается студентами специальностей 230101.65, 230102.65 и 230105.65. всех форм обучения в одном семестре. Дисциплина включает в себя следующие разделы: основные понятия и определения; запоминающие устройства, арифметические устройства, устройства управления, процессоры и системные средства ЭВМ и оценка характеристик этих устройств, особенности архитектуры ЭВМ различных классов.

Дисциплина изучается в течение одного семестра, объем лекционных и лабораторных часов определяется формой обучения. Завершается изучение дисциплины защитой курсового проекта и экзаменом.

Целью изучения дисциплины является изучение организации и принципов построения современных ЭВМ и систем, теоретических основ их анализа, проектирования и исследования, взаимодействия их программных и аппаратных средств.

Задачи изучения дисциплины – усвоение основных принципов, особенностей построения и взаимосвязи характеристик технических средств современных ЭВМ и систем.

В результате изучения дисциплины студент должен овладеть основами знаний по дисциплине, формируемыми на нескольких уровнях:

Иметь представление:

- о принципах и особенностях организации ЭВМ и систем различных классов, их устройств и блоков;
- о взаимодействии аппаратных и программных средств ЭВМ и систем
- о возможностях применения и перспективах развития ЭВМ и систем.

Знать:

- основные принципы организации ЭВМ и систем;
- алгоритмы функционирования и структурную организацию основных устройств ЭВМ;
- методы оценки характеристик ЭВМ и систем и отдельных их устройств;
- основные системные требования к техническим средствам ЭВМ, входящим в состав различных информационных и управляющих систем;
- технические характеристики и экономические показатели лучших отечественных и зарубежных образцов ЭВМ и систем.

У*меть* применять их для решения задач проектирования, выбора конфигурации, настройки и эксплуатации современных ЭВМ и систем:

- оценивать производительность отдельных устройств и ЭВМ в целом, зная отдельные ее составляющие;
- определять класс и конфигурацию ЭВМ, наилучшим образом удовлетворяющую требованиям к функционированию ее в конкретной информационной, вычислительной или управляющей системе;
- обучать пользователей правилам и необходимым навыкам эксплуатации ЭВМ и систем.

Владеть:

- методами представления структурных и функциональных схем ЭВМ и систем;

- умением выбрать устройства и блоки, необходимые для построения вычислительной системы, отвечающей заданным требованиям.

Место дисциплины в учебном процессе

Теоретической и практической основами дисциплины являются курсы “Математика”, “Информатика”, “Дискретная математика”, “Программирование на языке высокого уровня”, “Математическая логика и теория алгоритмов”, “Теория автоматов”, “Электротехника и электроника”, “Операционные системы”, “Системное программное обеспечение”, “Моделирование”, часть из которых изучается параллельно с данной дисциплиной.

Приобретенные студентами знания будут непосредственно использованы при изучении следующих дисциплин различными специализациями: “Сети ЭВМ и телекоммуникации”, “Микропроцессоры”, “Интерфейсы периферийных устройств”, “Схемотехника ЭВМ”, “Базы данных”, “Проектирование информационных систем”, а также в курсовом и дипломном проектировании.

1.2. Содержание дисциплины и виды учебной работы

Содержание дисциплины по ГОС

Основные характеристики, области применения ЭВМ различных классов; функциональная и структурная организация процессора; организация памяти ЭВМ; основные стадии выполнения команды; организация прерываний в ЭВМ; организация ввода-вывода; периферийные устройства; архитектурные особенности организации ЭВМ различных классов; параллельные системы; понятие о многомашинных и многопроцессорных вычислительных системах (ВС).

Объем дисциплины и виды учебной работы

Вид учебной работы	Всего часов		
	Форма обучения		
	Очная	Очно-заочная	Заочная
Общая трудоемкость дисциплины (ОТД)	**140**		
Работа под руководством преподавателя (включая ДОТ)	**84**	**84**	**84**

В т.ч. аудиторные занятия: лекции практические занятия (ПЗ) лабораторные работы (ЛР)	 **42** **8** **20**	 **16** **4** **16**	 **4** **4** **8**
Самостоятельная работа студента (СР)	**56**	**56**	**56**
Промежуточный контроль, количество В т. ч.: курсовой проект	**1** **1**	**1** **1**	**1** **1**
Вид итогового контроля (зачет, экзамен)	**Экзамен**		

Перечень видов практических занятий и контроля:

- тесты (по разделам дисциплины);
- практические занятия – 8 (для очной формы), 4 часа (для остальных форм);
- лабораторные работы – 20 (для очной формы), 16 (для очно-заочной формы), 8 (для заочной формы);
- курсовой проект;
- экзамен.

2. Рабочие учебные материалы

2.1. РАБОЧАЯ ПРОГРАММА (объем 140 часов)

Введение (2 часа)
[1], с. 22…25, 43...45, 50

Цель и задачи курса, его роль в подготовке специалистов по вычислительной технике и взаимосвязь с другими дисциплинами специальности.

Основные исторические сведения, роль отечественных ученых в разработке теории и создании ЭВМ.

Цифровая, аналоговая и гибридная формы представления информации. Классификация вычислительных средств по формам представления информации и принципу организации вычислений. Перспективы и тенденции развития технических средств ЭВМ, методов их проектирования и применения.

Раздел 1. ОБЩИЕ СВЕДЕНИЯ ОБ ЭВМ (12 часов)

1.1. Основные типы ЭВМ (8 часов)
[1], с. 35...43, 148...150; [2], с. 40...48

Основные виды ЭВМ, обобщенная структура ЭВМ, принцип программного управления, принципы фон Неймана. Основные характеристики ЭВМ.

Классификация ЭВМ. Особенности и области применения ЭВМ различных классов. Режимы работы ЭВМ.

1.2. Общие принципы организации ЭВМ (4 часа)
[1], с. 23...25, 150...153; [5], с. 18...26

Системные принципы организации технических средств ЭВМ. Функционально-структурный подход, основные функции систем переработки информации. Взаимосвязь функциональных возможностей, структуры, функций и основных технических характеристик устройств ЭВМ.

Общие сведения о методах оценки производительности и эффективности ЭВМ.

Раздел 2. ЗАПОМИНАЮЩИЕ УСТРОЙСТВА ЭВМ (25 часов)

2.1. Основные характеристики и типы запоминающих устройств ЭВМ (6 часов)
[7], с. 3...9, 12...16, 21...27

Основные понятия и определения. Классификация запоминающих устройств. Организация памяти ЭВМ. Основные характеристики ЗУ.

Иерархическая организация многоуровневой памяти ЭВМ. ЗУ с последовательной и произвольной выборкой, адресные и безадресные ЗУ.

2.2. Оперативные и сверхоперативные ЗУ (9 часов)

[1], с. 213...225, 249...260; [7] с. 37...54, 59…61, 127…134

Назначение, структура и организация работы оперативных ЗУ (ОЗУ). Многоканальный доступ и расслоение обращений.

Полупроводниковые ОЗУ. Элементы памяти, структурная организация, диаграммы работы полупроводниковых ОЗУ.

Организация и основные разновидности модулей ЗУ на БИС.

Сверхоперативные ЗУ, организация их работы. Кэш-память

2.3. Организация ЗУ различных типов (10 часов)

[1], с. 226...236, 245...248, 271…275, 286…290; [7] с. 63...72, 79…83, 95…99

Постоянные ЗУ (ПЗУ), их разновидности и организация.

Флэш-память.

Ассоциативные и многофункциональные ЗУ.

ЗУ на жестких и гибких магнитных дисках. ЗУ на оптических дисках.

Новые технологии и перспективы развития ЗУ.

Раздел 3. ПРОЦЕССОРЫ ЭВМ (45 часов)

3.1. Общие сведения о структуре процессоров ЭВМ (5 часов)

[1], с. 126...131; [2], с. 151...168

Назначение процессора. Функциональная и структурная организация процессора.

Базовые функциональные узлы устройств процессора.

3.2. Арифметико-логические устройства процессоров (24 часа)

[1], с. 329...355, 370...385; [5], с. 4...25, 35...54

Арифметико-логические устройства (АЛУ). Назначение, принципы организации и основные характеристики АЛУ, их классификация.

Средства описания АЛУ. Базовые преобразования структур АЛУ. Обобщенные структурные схемы операционных устройств.

Структура АЛУ и алгоритмы выполнения основных арифметических операций.

Особенности построения АЛУ и алгоритмы выполнения арифметических операций над двоично-десятичными числами.

Выполнение логических операций в АЛУ.

3.3. Устройства управления ЭВМ (16 часов)

[1], с. 106...116, 138…144, 293…316; [6], с. 5...15, 24...33, 75...83

Основные понятия, назначение и классификация устройств управления (УУ), их функции.

Организация управления выполнением последовательности команд и операций. Основные стадии выполнения команды. Взаимодействие узлов УУ при реализации переходов, циклов, обращений к процедурам и др.

Системы адресации ЭВМ.

Схемные УУ. УУ на основе распределителей управляющих сигналов. УУ с жесткой логикой на основе микропрограммных автоматов.

Микропрограммные УУ.

Раздел 4. СИСТЕМНЫЕ СРЕДСТВА И АРХИТЕКТУРА ЭВМ (55 часов)

4.1. Системы прерывания программ и системы памяти ЭВМ (18 часов)

[1], с. 264...272; [3] с. 112…125; [4], с. 232...250; [7], с. 22...29, 135…149

Системы прерывания программ ЭВМ, виды прерываний. Организация прерываний в ЭВМ, основные структурные схемы и характеристики систем прерываний. Приоритетное обслуживание прерываний.

Прерывания в персональных ЭВМ.

Системы памяти ЭВМ, их классификация и характеристики.

Страничная и сегментная организация памяти.

Способы защиты памяти.

Управление обменом с внешней памятью, дисциплины обслуживания обращений к внешним ЗУ, дисковые массивы.

4.2. Организация ввода-вывода информации в ЭВМ (12 часов)

[1], с. 159…181, 387…411

Организация ввода-вывода. Управление вводом-выводом в многопрограммных ЭВМ. Алгоритмы и структура интерфейсов ввода-вывода при различных видах обмена: программно-управляемом, по прерыванию, с прямым доступом к памяти. Организация шин интерфейса. Типовые интерфейсы ЭВМ.

Каналы ввода-вывода. Периферийные устройства ЭВМ.

4.3. Архитектура ЭВМ и вычислительных систем (20 часов)

[1], с. 460...470, 496…505, 525…529, 553…575, 584…620

Архитектурные особенности организации ЭВМ различных классов.

Понятие о многомашинных и многопроцессорных вычислительных системах (ВС). Вычислительные комплексы (ВК). Параллельные системы.

Классификация и основные типы вычислительных систем.

Матричные, ассоциативные, конвейерные, потоковые ВС. Сети ЭВМ.

4.4. Принципы построения аналоговых и гибридных ЭВМ (5 часов)

[8], с. 9...28

Физическое и математическое моделирование. Основные характеристики аналоговых и гибридных вычислительных машин.

Принципы построения вычислительных устройств на основе операционного усилителя. Суммирующие, интегрирующие и дифференцирующие устройства.

Множительные и делительные устройства. Устройства и методы воспроизведения нелинейных функций.

Заключение (1 час)

[1], с. 7...8; [2], с. 28

Перспективные направления развития архитектуры и технических средств ЭВМ.

2.2. Тематические планы дисциплины

2.2.1. Тематический план дисциплины

для студентов очной формы обучения

№ п/п	Наименование раздела (отдельной темы)	Кол-во часов по дневной форме обучения	Виды занятий и контроля											
			Лекции		ПЗ (С)		ЛР		Самостоятельная работа	Тесты	Контрольные работы	ПЗ (С)	ЛР	Курсовые работы (проекты)
			аудит.	ДОТ	аудит.	ДОТ	аудит.	ДОТ						
ВСЕГО		**140**	**42**	**14**	**8**		**20**		**56**					
В	**Введение**	**2**		1					**1**					
1.	**Раздел 1. Общие сведения об ЭВМ**	**12**							**5**	№1				
1.1	Основные типы ЭВМ	8	2	1										
1.2	Общие принципы организации ЭВМ	4	2		2									
2.	**Раздел 2. Запоминающие устройства ЭВМ**	**25**							**12**	№2				
2.1	Основные характеристики и типы запоминающих устройств ЭВМ	6	2	1										
2.2	Оперативные и сверхоперативные ЗУ	9	3	1	2									
2.3	Организация ЗУ различных типов	10	3	1										
3.	**Раздел 3. Процессоры ЭВМ**	**45**							**11**	№3				
3.1	Общие сведения о структуре процессоров ЭВМ	5	2		2									
3.2	Арифметико-логические устройства процессоров	24	6	2			8						№1	
3.3	Устройства управления ЭВМ	16	6	2	2		4						№2	
4.	**Раздел 4. Системные средства и архитектура ЭВМ**	**55**							**27**	№4				
4.1	Системы прерывания программ и системы памяти ЭВМ	18	3	2										
4.2	Организация ввода-вывода информации в ЭВМ	12	3	1										
4.3	Архитектура ЭВМ и вычислительных систем	20	8	1			8						№3	
4.4	Принципы построения аналоговых и гибридных ЭВМ	5	2											
З	**Заключение**	**1**		1										

2.2.2. Тематический план дисциплины

для студентов очно-заочной формы обучения

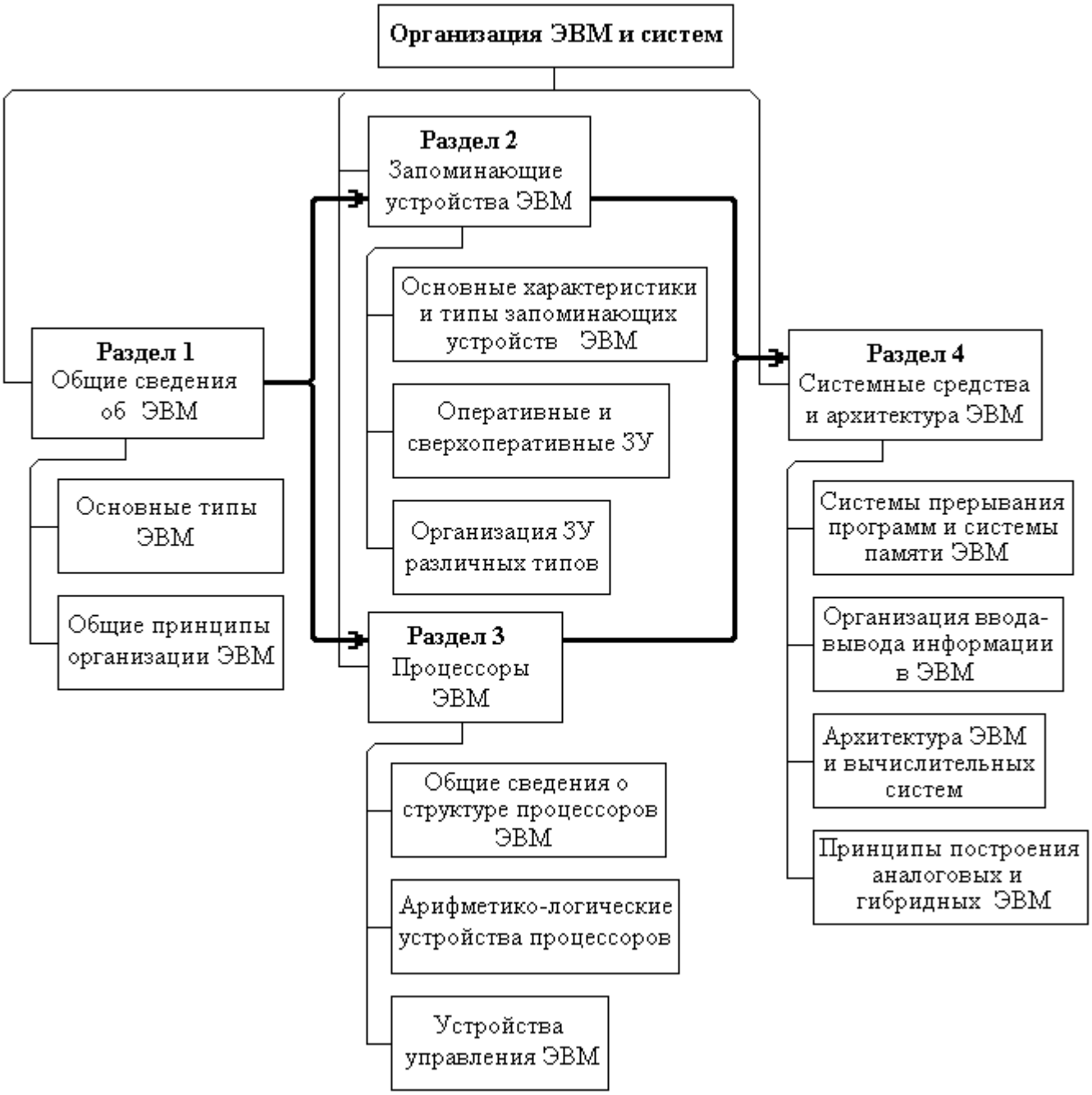
№ п/п	Наименование раздела (отдельной темы)	Кол-во часов по дневной форме обучения	Виды занятий и контроля											
			Лекции		ПЗ (С)		ЛР		Самостоятельная работа	Тесты	Контрольные работы	ПЗ (С)	ЛР	Курсовые работы (проекты)
			аудит.	ДОТ	аудит.	ДОТ	аудит.	ДОТ						
ВСЕГО		**140**	**16**	**44**	**4**	**4**	**16**		**56**					
В	**Введение**	**2**		1					**1**					
1.	**Раздел 1. Общие сведения об ЭВМ**	**12**							**5**	№1				
1.1	Основные типы ЭВМ	8		3										
1.2	Общие принципы организации ЭВМ	4	1	1		2								
2.	**Раздел 2. Запоминающие устройства ЭВМ**	**25**							**10**	№2				
2.1	Основные характеристики и типы запоминающих устройств ЭВМ	6	1	3										
2.2	Оперативные и сверхоперативные ЗУ	9	2	3		2								
2.3	Организация ЗУ различных типов	10		4										
3.	**Раздел 3. Процессоры ЭВМ**	**45**							**14**	№3				
3.1	Общие сведения о структуре процессоров ЭВМ	5	1	2	2									
3.2	Арифметико-логические устройства процессоров	24	2	5			6						№1	
3.3	Устройства управления ЭВМ	16	2	5	2		4						№2	
4.	**Раздел 4. Системные средства и архитектура ЭВМ**	**55**							**26**	№4				
4.1	Системы прерывания программ и системы памяти ЭВМ	18	2	6										
4.2	Организация ввода-вывода информации в ЭВМ	12	1	3										
4.3	Архитектура ЭВМ и вычислительных систем	20	4	5			6						№3	
4.4	Принципы построения аналоговых и гибридных ЭВМ	5		2										
З	**Заключение**	**1**		1										

2.2.3. Тематический план дисциплины

для студентов заочной формы обучения

№ п/п	Наименование раздела (отдельной темы)	Кол-во часов по дневной форме обучения	Виды занятий и контроля											
			Лекции		ПЗ (С)		ЛР		Самостоятельная работа	Тесты	Контрольные работы	ПЗ (С)	ЛР	Курсовые работы (проекты)
			Аудит.	ДОТ	аудит.	ДОТ	аудит.	ДОТ						
ВСЕГО		**140**	**4**	**60**	**4**	**4**	**8**	**4**	**56**					
В	**Введение**	**2**		1					**1**					
1.	**Раздел 1. Общие сведения об ЭВМ**	**12**							**5**	№1				
1.1	Основные типы ЭВМ	8		3										
1.2	Общие принципы организации ЭВМ	4	1	1		2								
2.	**Раздел 2. Запоминающие устройства ЭВМ**	**25**							**9**	№2				
2.1	Основные характеристики и типы запоминающих устройств ЭВМ	6		4										
2.2	Оперативные и сверхоперативные ЗУ	9	1	5		2								
2.3.	Организация ЗУ различных типов	10		4										
3.	**Раздел 3. Процессоры ЭВМ**	**45**							**13**	№3				
3.1	Общие сведения о структуре процессоров ЭВМ	5	1	2	2									
3.2	Арифметико-логические устройства процессоров	24		10			3						№1	
3.3	Устройства управления ЭВМ	16		10	2		2						№2	
4.	**Раздел 4. Системные средства и архитектура ЭВМ**	**55**							**28**	№4				
4.1	Системы прерывания программ и системы памяти ЭВМ	18		8										
4.2	Организация ввода-вывода информации в ЭВМ	12		4										
4.3	Архитектура ЭВМ и вычислительных систем	20	1	8			3						№3	
4.4	Принципы построения аналоговых и гибридных ЭВМ	5		3										
З	**Заключение**	**1**		1										

2.3. Структурно-логическая схема дисциплины



2.4. Временной график изучения дисциплины

№	Название раздела (темы)	Продолжительность изучения раздела (темы) в днях (из расчета – 4 часа в день)
1	Раздел 1. Общие сведения об ЭВМ	3 дн.
2	Раздел 2. Запоминающие устройства ЭВМ	6 дн.
3	Раздел 3. Процессоры ЭВМ	11 дн.
4	Раздел 4. Системные средства и архитектура ЭВМ	14 дн.
5	Курсовой проект	9 дн
	ИТОГО	43 дн.

2.5. Практический блок

Практические занятия (очная форма обучения)

Номер и название раздела (темы)	Наименование тем практических занятий	Кол-во часов
Тема 1.2	Оценка времени выполнения микропрограммы	**2**
Тема 2.2	Организация модулей ЗУ на больших интегральных схемах	**2**
Тема 3.1	Построение схемы разряда операционного устройства	**2**
Тема 3.3	Разработка алгоритмов управления выполнением команд	**2**

Практические занятия (очно-заочная и заочная формы обучения)

Номер и название раздела (темы)	Наименование тем практических занятий	Кол-во часов
Тема 1.2	Оценка времени выполнения микропрограммы	**1**
Тема 2.2	Организация модулей ЗУ на больших интегральных схемах	**1**
Тема 3.1	Построение схемы разряда операционного устройства	**1**
Тема 3.3	Разработка алгоритмов управления выполнением команд	**1**

Лабораторные работы (очная форма обучения)

Номер и название раздела (темы)	Наименование лабораторной работы	Кол-во часов
Тема 3.2	Исследование структуры и принципа действия двоичного арифметического устройства	**8**
Тема 3.3	Исследование устройства микропрограммного управления	**4**
Тема 4.3	Определение конфигурации и оценка производительности ПЭВМ	**8**

Лабораторные работы (очно-заочная форма обучения)

Номер и название раздела (темы)	Наименование лабораторной работы	Кол-во часов
Тема 3.2	Исследование структуры и принципа действия двоичного арифметического устройства	**6**
Тема 3.3	Исследование устройства микропрограммного управления	**4**
Тема 4.3	Определение конфигурации и оценка производительности ПЭВМ	**6**

Лабораторные работы (заочная форма обучения)

Номер и название раздела (темы)	Наименование лабораторной работы	Кол-во часов
Тема 3.2	Исследование структуры и принципа действия двоичного арифметического устройства	**3**
Тема 3.3	Исследование устройства микропрограммного управления	**2**
Тема 4.3	Определение конфигурации и оценка производительности ПЭВМ	**3**

2.6. Балльно-рейтинговая система

Изучение курса заканчивается сдачей экзамена. По курсу предусмотрены лабораторные работы, практические занятия и курсовой проект.

Дисциплина включает в себя четыре раздела, по завершении каждого из которых сдается контрольный тест. Тесты содержат 8, 9, 15 и 20 вопросов соответственно по первому, второму, третьему и четвертому разделам. Ответ на каждый вопрос оценивается в 0,5 балла. Итого по тестам можно получить в сумме 26 баллов

По курсу предусмотрено выполнение трех лабораторных работ, за правильное выполнение первой работы начисляется 8 баллов, второй – 5, а третьей – 7 баллов. Итого за выполнение цикла лабораторных работ можно получить 20 баллов.

Кроме того, также предусматривается выполнение четырех практических заданий, за каждое из которых можно получить по 3,5 балла, т. е. в сумме это составит 14 баллов.

Выполнение курсового проекта, состоящего из двух частей, позволяет получить за каждую часть по 18 баллов, т. е. 36 баллов за весь проект.

Итого максимальная сумма баллов, которую можно получить по дисциплине составляет 100 баллов. Для получения допуска к экзамену необходимо выполнить курсовой проект и набрать более двух третьих от этой суммы. **То есть если Вы при работе с самостоятельными заданиями набрали более 67 баллов, допуск к экзамену Вам обеспечен.**

3. Информационные ресурсы дисциплины

3.1. Библиографический список

Основной:

1. Цилькер, Б.Я. Организация ЭВМ и систем: учебник для вузов / Б.Я. Цилькер, С.А. Орлов. – СПб.: Питер, 2006. – 668 с.

2. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум. – СПб.: Питер, 2002, 2007.

Дополнительный:

3. Гук, М. Аппаратные средства IBM PC: энциклопедия / М. Гук. – 3-е изд. – СПб.: Питер, 2008. - 928 с.

4. Хамахер, К. Организация ЭВМ / К. Хамахер, З. Вранешич, С. Заки.- 5-е изд. – СПб.: Питер, 2003. - 848 с.

5. Спиридонов, В.В. Проектирование структур АЛУ: учеб. пособие / В.В. Спиридонов. – СПб.: СЗПИ, 1992. - 84 с.

6. Копейкин, М.В. Управление ЭВМ: учеб. пособие / М.В. Копейкин, В.Я. Пашкин, В.В. Спиридонов. - Л.: СЗПИ, 1988. - 84 с.

7. Копейкин, М.В. Организация ЭВМ и систем: (память ЭВМ): учеб. пособие / М.В. Копейкин, В.В. Спиридонов, Е.О. Шумова. – СПб.: Изд-во СЗТУ, 2004. - 153 с.

8. Каган, Б.М. Электронные вычислительные машины и системы: учеб. пособие для вузов / Б.М. Каган. - М.: Энергоатомиздат, 1991. - 552 с.

9. Организация ЭВМ и систем: метод. указ. к выполнению лаб. работ / сост.: М.В. Копейкин, В.В. Спиридонов, Е.О. Шумова. – СПб.: Изд-во СЗТУ, 2005. - 35 с.

10. Организация ЭВМ и систем: метод. указ. к выполнению курс. проекта / сост.: М.В. Копейкин, В.В. Спиридонов, Е.О. Шумова. – СПб.: Изд-во СЗТУ, 2005. - 51 с.

Средства обеспечения освоения дисциплины (ресурсы Internet)

1. http://www.ord.com.ru/files/org_evm
2. http://www.gpntb.ru/win/windows/

3.2. Опорный конспект лекций по дисциплине

ВВЕДЕНИЕ

Вы начинаете изучение дисциплины “Организация ЭВМ и систем”.

Наша специальность в своем первоначальном варианте возникла еще в тридцатых годах прошлого столетия. В то время она была связана с построением средств управления вооружением и носила название “Приборы управления стрельбой”.

Первая из кафедр, начавшая подготовку по данной специальности в СССР, была образована по Постановлению Совнаркома в Ленинградском электротехническом институте им. В.И. Ульянова (Ленина) в 1931 г. (ныне это Санкт-Петербургский электротехнический университет “ЛЭТИ”). Затем в 1937 году подготовка по аналогичной специальности – “Математические и счетно-решающие приборы и устройства” была организована в Ленинградском институте точной механики и оптики (ЛИТМО). Позже к этим кафедрам присоединились и аналогичные кафедры в московских вузах.

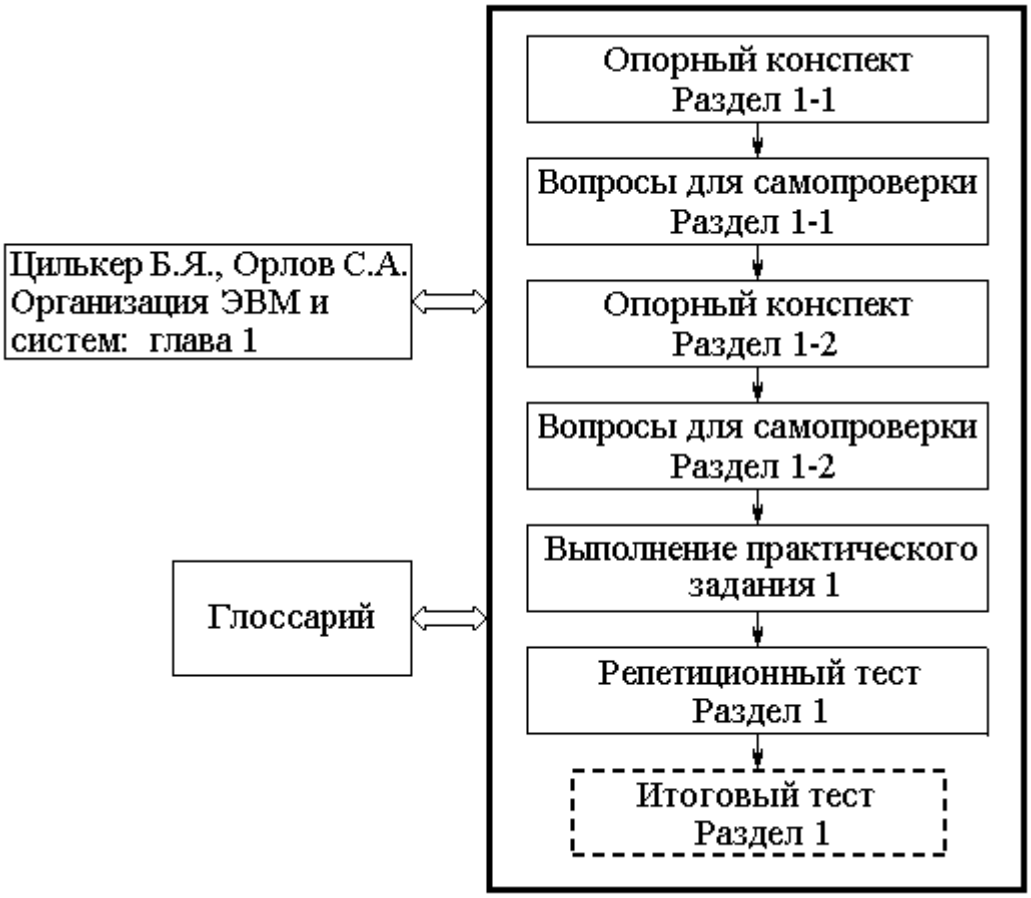
Название специальности достаточно долго оставалось “Математические и счетно-решающие приборы и устройства”, затем его изменили на “Электронные вычислительные машины”, а позже – на “Вычислительные машины, комплексы, системы и сети”. При этом из нее выделились такие специальности, как “Программное обеспечение вычислительной техники и автоматизированных систем”, “Автоматизированные системы обработки информации и управления” и некоторые другие.

Дисциплина “Организация ЭВМ и систем” также изменялась в процессе становления специальности. ЭВМ появились к концу 40-х годов прошлого века. В то время почти на равных сосуществовали электронные вычислительные машины непрерывного действия, или аналоговые ЭВМ, и вычислительные машины дискретного действия – цифровые ЭВМ. Одним из первых вариантов нашей дисциплины и был курс “Вычислительные машины дискретного действия”. Позднее дисциплину стали называть “Цифровые вычислительные машины”, “Теория и проектирование ЭВМ”. Затем из нее стали выделяться самостоятельные курсы, рассматривающие в отдельности вопросы, связанные с микропроцессорами, периферийными устройствами, вычислительными сетями и некоторые другие, а сама дисциплина приобрела ее сегодняшнее название “Организация ЭВМ и систем”.

В результате изучения курса Вы получите представление об организации и принципах построения современных ЭВМ и систем, теоретических основах их анализа, проектирования и исследования, взаимодействии их программных и аппаратных средств; научитесь оценивать технические характеристики ЭВМ и выбирать их конфигурацию в соответствии с требованиями конкретной информационной, вычислительной или управляющей системы.

В курсе также уделяется внимание вопросам иллюстрации принципов организации и функционирования ЭВМ на примере персональных ЭВМ.

Схема работы с разделом 1



Раздел 1. ОБЩИЕ СВЕДЕНИЯ ОБ ЭВМ

Первый раздел курса включает две темы: "***Основные типы ЭВМ"*** и "***Общие принципы организации ЭВМ"***. После изучения каждой темы Вам следует ответить на вопросы для самопроверки.

Работа с разделом 1 завершается сдачей контрольного теста. Кроме того, в данном разделе выполняется практическое задание 1.

Для того, чтобы Вы смогли успешно ответить на вопросы контрольного теста, Вам предоставляется возможность поработать с репетиционным тестом. Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к материалам файла Lect1.doc учебного сайта либо раздела сайта ord.com.ru/files/org_evm или к главе 1 учебника [1].

1.1. Основные типы ЭВМ

При изучении данной темы Вы должны познакомиться с основными видами ЭВМ, структурой цифровых ЭВМ, принципом программного управления и принципами фон Неймана. Также следует усвоить основные характеристики ЭВМ, их классификацию и области применения и рассмотреть режимы работы ЭВМ.

Для проверки изучения материала темы Вам предстоит ответить на вопросы для самопроверки.

Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к материалам файла Lect1.doc учебного сайта либо раздела сайта ord.com.ru/files/org_evm или к учебнику [1], глава 1.

1.1.1. Основные виды ЭВМ

Развитие вычислительной техники шло в двух главных формах: цифровой и аналоговой, временны**е** рамки. Основные их отличия состоят в *форме представления информации* и в способе *организации вычислительного процесса* и обусловливают особенности принципов действия, структурной организации и технической реализации этих классов ЭВМ.

В цифровых вычислительных машинах применяют *алфавитную* (цифровую, дискретную) форму представления информации, использующую фиксированное множество знаков (алфавит), с помощью которого и записывается любая информация. На аппаратном уровне этот алфавит обычно состоит всего из двух знаков: нуля и единицы.

В аналоговых вычислительных машинах (АВМ) используется *аналоговая* (непрерывная, модельная) форма представления информации, при которой математическим величинам в решаемых задачах ставятся в соответствие некоторые параметры каких-либо физических процессов. Значения этих параметров и соответствуют значениям моделируемых математических величин. Наиболее часто в аналоговых машинах моделирующим процессом является процесс протекания электрического тока в цепях, состоящих из резисторов, конденсаторов, усилителей и т.п., а параметром, представляющим математические величины, является напряжение постоянного тока.

В цифровых вычислительных машинах используется *программная организация* вычислительного процесса, при этом решение задачи сводится к выполнению последовательности шагов – команд программы.

В аналоговых вычислительных машинах решение задачи осуществляется с помощью схемы, собираемой из решающих блоков машины, поэтому говорят, что в АВМ имеет место *схемная организация* вычислительного процесса.

Эти различия обусловливают достоинства и недостатки, свойственные двум основным видам вычислительных машин.

К *достоинствам цифровых ЭВМ* относят, в первую очередь:

- универсальность;
- высокое быстродействие;
- высокую точность (малую погрешность) вычислений.

Недостатком цифровых ЭВМ является, главным образом, их высокая сложность, из которой следуют:

- относительно высокая стоимость;
- сложность эксплуатации;
- сложность программирования.

Аналоговые ЭВМ в настоящее время вытеснены цифровыми ЭВМ. Это связано с недостаточной универсальностью и высокой погрешностью вычислений на них, являющихся основными *недостатками аналоговых ЭВМ*.

К *достоинствам аналоговых ЭВМ* можно отнести:

- относительную простоту;
- высокое быстродействие.

Несмотря на утрату аналоговыми машинами своих позиций, принцип моделирования, положенный в основу их построения, не может быть исключен из теории и практики создания средств вычислительной техники.

Существуют и ЭВМ, сочетающие в себе черты аналоговых и цифровых ЭВМ, получившие название гибридных, или комбинированных.

1.1.2. Структура цифровых ЭВМ и принципы Дж. фон Неймана

В 30-40-х годах XX века велись работы по созданию быстродействующих вычислительных средств. Электроника тогда еще только завоевывала свои позиции, и разработка вычислительных устройств осуществлялась сначала на основе электромеханических схем. Но их быстродействие было невысоким.

Наибольшую известность среди первых электронных вычислительных машин получил проект, развернутый в 1942 г. в университете Пенсильвании. Задачи, для решения которых проектировался вычислитель, были связаны с баллистическими расчетами. Проект возглавили Дж. Мочли (John J. Mauchly) и Дж. Экерт (J. Presper Eckert), в роли консультанта в нем участвовал и Дж. фон Нейман (John von Neumann).

Работы над ЭНИАК официально были завершены в конце 1945 г. Он был построен на 18000 электронных ламп, занимал помещение 135 м^2, весил более 30 т и потреблял мощность 150 кВт. Машина работала на частоте 100 кГц и выполняла операции примерно в 1000 раз быстрее, чем электромеханические машины. Операции производились в десятичной системе счисления, а программирование осуществлялось посредством соединений, выполнявшихся проводами на коммутационной панели. Переход от одной программы к другой мог занять от 30 минут до 8 часов, требуя переключения до нескольких тысяч проводов.

Сразу стало очевидно, что такое программирование надо заменить хранением программы электронным способом. Да и надежность этой ЭВМ была невысокой, а поиск неисправностей мог занять от нескольких часов до суток, хотя одним из первых практических применений было использование ее при решении некоторых задач проекта атомной бомбы.

В июне 1946 г. Дж. фон Нейман с коллегами опубликовал отчет “Предварительное обсуждение логического конструирования электронного вычислительного устройства” (A.W.Burks, H.H.Goldstine, J. von Neumann, Preliminary discussion of the logical design of an electronic computing instrument, – Princeton, 1946). В нем содержалось обоснование выбора конструкции ЭВМ, выполненное на основе анализа сильных и слабых сторон ЭНИАК, рассматривалась ее структура и предлагался ряд принципов построения ЭВМ.

Обсуждаемую в отчете структуру ЭВМ можно изобразить так, как показано на рис. 1.1, объединив раздельные на оригинальной схеме устройства ввода и вывода. Назначение отдельных блоков отвечает их названиям.

Показанная на рис. 1.1 пунктиром связь между устройствами ввода-вывода (УВВ) и запоминающим устройством (ЗУ) отсутствовала в исходной структуре, но появилась в ЭВМ достаточно быстро. Впоследствии управление этой связью было возложено на выделенные в самостоятельные блоки каналы

(контроллеры) ввода-вывода, а основная часть устройства управления совместно с арифметическим (позже арифметико-логическим) устройством объединилась в блок, названный процессором.

В отчете также были даны рекомендации, касающиеся организации ЭВМ, часто называемые принципами Неймана. Часть этих рекомендаций оказалась ключевой для последующих поколений ЭВМ. К ним относятся следующие:

1. Машины на электронных элементах должны работать не в десятичной, а в двоичной системе счисления.

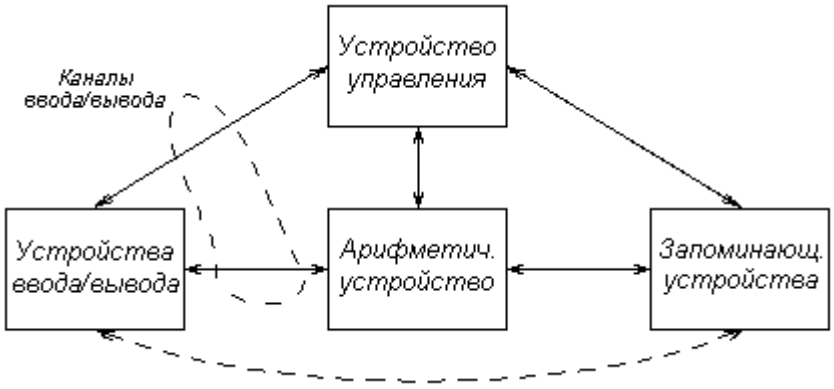


Рис. 1.1. Структурная схема цифровой вычислительной машины

2. Программа должна размещаться в запоминающем устройстве машины, обладающем достаточной емкостью и высокой скоростью выборки и записи команд программы.

3. Программа так же, как и данные, записывается в двоичном коде. Это дает однотипное представление команд и данных.

4. Невозможность технической реализации ЗУ большой емкости, быстродействие которого соответствовало бы быстродействию логических схем обрабатывающих блоков, требует иерархической организации памяти.

5. В машине должны реализовываться аппаратно параллельные операции сложения, вычитания и умножения. Требование аппаратной реализации операций деления и извлечения квадратного корня неочевидны.

Содержание работы Неймана и его коллег оказало значительное влияние на последующие поколения вычислительной техники, а соответствующую структуру ЭВМ стали называть “архитектурой фон Неймана”.

1.1.3. Классификация (цифровых) ЭВМ

Классифицировать ЭВМ можно по различным признакам, ряд которых приведен ниже.

1. По *количеству процессоров/ядер*:

- однопроцессорные;
- многопроцессорные (многоядерные).

Это разделение служит как общая характеристика архитектуры и сложности ЭВМ, ее функциональных возможностей и производительности.

2. По *функциональному назначению*:

- универсальные;

- специализированные.

Универсальные ЭВМ предназначены для решения различных задач, а *специализированные* – преимущественно или исключительно для решения задач определенного класса либо для работы в специфических условиях или при специальных требованиях.

Иногда выделяют проблемно-ориентированные ЭВМ, относя их к классу, промежуточному между универсальными и специализированными ЭВМ.

3. По *вычислительной мощности*:

- программируемые калькуляторы;
- карманные компьютеры (и коммуникаторы);
- ноутбуки (лэптопы);
- рабочие станции (десктопы);
- серверы (масштаба отдела, предприятия);
- мэйнфреймы (большие и сверхбольшие ЭВМ);
- вычислительные комплексы (локальные и распределенные).

Данное разделение базируется на значениях основных характеристик ЭВМ, особенностях их применения и реализации и др., и в рамках указанных видов имеются представители с заметно различающимися параметрами.

4. По *архитектуре* можно выделить следующие типы ЭВМ:

- с традиционной (фон Неймановской) архитектурой;
- с RISC- архитектурой;
- с конвейерной архитектурой;
- с векторной и матричной архитектурой;
- с динамической архитектурой;
- ЭВМ с управлением потоками данных.

Основные из перечисленных типов рассматриваются в разделе 4.4.

5. По *физическим принципам*, используемым для реализации ЭВМ:

- на электронных элементах;
- оптоэлектронные и оптические;
- криоэлектронные;
- биомолекулярные;
- квантовые.

Используются и другие признаки классификации ЭВМ.

1.1.4. Режимы работы ЭВМ

Первые ЭВМ работали в однопрограммном режиме: запущенная на исполнение программа полностью занимала процессор (такого термина тогда еще не было) ЭВМ и выполнялась до своего завершения. Помимо неудобств работы в таком режиме, низкой оказывалась и загрузка устройств ЭВМ.

Это привело к идее, а затем и реализации многопрограммного режима работы, при котором отдельные устройства ЭВМ поочередно использовались несколькими программами или несколькими частями (процессами) одной программы. Такой режим должен был повысить и производительность, и коэффициент (полезного) использования отдельных устройств и ЭВМ в целом.

Для реализации этого режима следовало обеспечить ряд требований и функциональных возможностей ЭВМ, в частности: независимость хранящихся в памяти программ и данных от абсолютных адресов памяти; систему приоритетов программ; средства прерывания программ и другие. Эти требования были сформулированы в работах конца 1950-х – начала 1960-х годов и реализованы уже в ЭВМ, созданных в начале 1960-х годов.

Возможность многопрограммной работы ЭВМ привела к появлению новых режимов их работы. Первоначально эти режимы подразделяли на режим пакетной обработки и режим разделения времени. Первый из них предполагал загрузку в ЭВМ группы (пакета) задач и запуск их на поочередное решение. Режим разделения времени, напротив, предполагал выделение для каждой из задач, ожидающих выполнения, некоторой порции (кванта) процессорного времени, в течение которого задача получала управление процессором.

Пакетный режим обеспечивал более эффективное использование ЭВМ, так как требовал минимального количества переключений с одной задачи на другую. Но работать в таком режиме, особенно, при отладке задачи, было сложно, так как контакта с ЭВМ программист не имел.

Напротив, режим разделения времени позволял работать, имитируя непосредственный контакт с ЭВМ, так как задачи попадали в очередь прямо при вводе с терминала. Такой режим гораздо удобнее для человека, но эффективность использования ЭВМ при этом меньше, чем в пакетном режиме.

Позднее появилось много модификаций этих режимов, в том числе смешанного типа, сочетающего различным образом их свойства.

Вопросы для самопроверки по теме 1.1

1. Какова погрешность вычислений на цифровой ЭВМ?
2. Что такое мэйнфрейм?
3. Назовите основные принципы фон Неймана
4. В каких единицах измеряется быстродействие ЭВМ?
5. Что такое поколение ЭВМ?
6. Какие существуют основные режимы работы ЭВМ?
7. Какие физические принципы могут быть положены в основу реализации ЭВМ?
8. Что такое аналоговая ЭВМ?

1.2. Общие принципы организации ЭВМ

При изучении данной темы Вы должны рассмотреть основные концепции и принципы, используемые при построении ЭВМ: базовые функции, иерархичность структуры, взаимосвязь функций и структуры, а также общие сведения о методах оценки ЭВМ

Для проверки изучения материала темы Вам предстоит ответить на вопросы для самопроверки. По данной теме также выполняется практическое задание 1, рекомендации по которому приведены в разделе методических указаний 3.4.

При затруднениях в ответе на вопросы обратитесь к материалам файла Lect1.doc учебного сайта либо раздела сайта ord.com.ru/files/org_evm или к главе 1 учебника [1].

1.2.1. Принципы функциональной и структурной организации ЭВМ

Функции, выполняемые вычислительной машиной или системой на любом уровне ее рассмотрения, могут быть разделены на четыре основные группы: обработка, хранение, передача и управление первыми тремя.

Как и другие технические системы, ЭВМ в своей организации подчиняются ряду закономерностей (системных принципов), обусловленных физическими законами, технологическими особенностями производства, экономическими условиями и психологией разработчиков и пользователей.

1. Противоречивая взаимосвязь показателей и характеристик ЭВМ

Показатели любой системы взаимосвязаны и в значительной мере противоречивы: выигрыш в одном показателе обычно ведет к потерям в другом (других). Исключения из этой закономерности могут иметь место при сменах парадигм и технологических прорывах. Например, при переходе от транзисторных ЭВМ к системам на интегральных схемах, или просто при улучшении технологических норм.

Проявление этой закономерности можно видеть при сопоставлении различных пар показателей ЭВМ: производительности, стоимости, надежности, потребляемой мощности, габаритов, эксплуатационных затрат и пр.

2. Иерархическая организация ЭВМ и систем

Организация ЭВМ и систем в целом, а также отдельных их подсистем и устройств подчиняется принципу иерархичности.

Типичные иерархии в организации ЭВМ:

- иерархия памяти,
- иерархия исполнительных блоков,
- иерархия управления вводом-выводом,
- иерархия блоков (систем) управления.

Иерархическая организация позволяет сбалансировать противоречия между основными характеристиками (устройств) ЭВМ, упростить управление системой.

3. Соответствие структуры системы ее функциональному назначению

Структура системы соответствует возлагаемым на нее функциям и подчиняется предъявляемым системе требованиям. В процессе развития ЭВМ имеют место два параллельно протекающих подпроцесса – эволюция функций и эволюция технологий [*Балашов Е.П. Эволюционный синтез систем. – М.: Радио и связь, 1985. - 328 с.*]. Эти подпроцессы взаимодействуют между собой в ходе основного процесса, однако ведущей среди них является эволюция функций. Новые функциональные требования к ЭВМ появляются по мере освоения и совершенствования имеющихся технологий. Со временем таким требованиям

становится все труднее вписываться в рамки имеющихся технологий, что приводит к технологическим прорывам.

4. Баланс пространственных (аппаратных) и временных (программных, микропрограммных) характеристик

Для достижения приемлемого соотношения характеристик системы используется баланс различных способов реализации требуемых ее функций:

Баланс между программной и аппаратной реализацией (e.g., обработка сигналов, ассоциативный доступ, определение параллелизма на уровне команд, RISC vs CISC)

Баланс между схемной и микропрограммной реализацией (семейства совместимых ЭВМ, микроконтроллеры и схемное управление)

Компромисс между последовательным и параллельным исполнением

обработка: ускоренные методы выполнения операций, многоблочные исполнительные подсистемы,

хранение: многоблочная оперативная память, RAID-массивы;

управление: конвейерная обработка, неупорядоченное исполнение;

передача: интерфейсы жестких дисков, оперативной памяти, PCI-Express.

Баланс между различными типами функций, их локализацией и структурой системы (например, ePOP, файл-сервер vs SQL-сервер, табличная реализация преобразований, “тонкий” клиент и “толстый” клиент etc.)

5. Виртуализация функций и структур

Поддержка одновременной работы различных ОС, эмуляция ОС

Эмуляция аппаратных средств

Система виртуальной памяти

Переименование регистров

1.2.2. Методы оценки производительности ЭВМ

Методы оценки производительности ЭВМ разделяются на *экспериментальные* и *теоретические*. Первые используют тестовые и измерительные программы, вторые – математические модели с применением аналитических и численных методов, а также статистического моделирования.

Обе группы методов, как правило, предназначены для получения оценок временн***ы***х характеристик и производительности ЭВМ, а также изучения влияния на эти характеристики различных параметров ЭВМ.

Экспериментальные методы оценки

Экспериментальные методы широко используются для оценки производительности ЭВМ. Суть их сводится к измерению той или иной характеристики действующей ЭВМ при выполнении на ней какой-либо работы (решении задачи или набора задач). Данные методы позволяют получить непосредственные значения интересующих величин с относительно малыми затратами, так как время измерения обычно невелико.

Процесс вычислений на ЭВМ имеет стохастический характер, и точность полученных оценок определяется соответствием выполняемой при измерениях

вычислительной работы реальной нагрузке ЭВМ и количеством прогонов выполняемой задачи или теста, по которому производится усреднение оценки.

Как правило, сравнение различных аппаратных конфигураций (системы памяти, ЭВМ в целом) можно корректно производить только в рамках одной и той же программы. Это вызвано тем, что даже при определении абсолютных значений характеристик эти значения для различных программ, как правило, не совпадают, иногда существенно, что связано с различными способами измерения одних и тех же параметров в разных программах.

Теоретические методы оценки

В тех случаях, когда экспериментальные методы оценки характеристик ЭВМ не дают необходимых результатов либо не могут быть использованы по какой-нибудь причине, например на этапе разработки архитектуры ЭВМ, применяют различные теоретические методы. Поскольку процесс функционирования ЭВМ имеет псевдостохастический характер, то и модели, используемые для оценки их характеристик, в основном вероятностные.

Одними из наиболее часто используемых моделей являются модели теории массового обслуживания (в англоязычной литературе – теории очередей). Ключевыми понятиями, используемыми в этих моделях, являются *поток запросов* на обслуживание (заявок) и *обслуживающий прибор*.

При представлении ЭВМ в виде системы массового обслуживания (СМО) процессору, отдельным ЗУ, контроллерам памяти и различных устройств, а при необходимости и трактам передачи (шинам) сопоставляются обслуживающие приборы, а командам процессора, обращениям к ЗУ, контроллерам, заявкам на циклы передачи по шинам – запросы на обслуживание.

Поток запросов на обслуживание характеризуется интенсивностью обращений λ, являющейся обратной величиной к математическому ожиданию $\bar{t}_r$ интервала времени t_r между поступлением соседних запросов, который представляет собой случайную величину с заданным законом распределения.

Процесс обслуживания запросов в обслуживающем приборе также считается случайным и характеризуется законом распределения времени обслуживания t_s и его математическим ожиданием $\bar{t}_s$, обратную величину к которому называют средним темпом (интенсивностью) обслуживания и обозначают μ.

Получение аналитических решений часто сопряжено с математическими трудностями и выполнимо лишь с принятием упрощающих предположений. Если упрощения неадекватно отображают протекание процессов в системе, используют более близкие к реальной ситуации потоки обращений к устройствам и законы распределения времени обслуживания. Обычно в этих случаях приходится использовать численные методы или методы статистического моделирования.

Одним из наиболее значимых для практики результатов, полученных с помощью таких моделей, можно считать то, что при загрузках системы (обозначаемых через $\rho = \lambda/\mu$) более 0,8 – 0,85 в ней имеет место резкий рост задержек. Поэтому, разрабатывая системы, следует стараться обеспечить резерв пропускной способности входящих в нее трактов передачи и устройств.

1.2.3. Оценка эффективности ЭВМ

Под эффективностью обычно понимается соотношение между затратами и достигаемым эффектом. Поскольку ЭВМ – это некоторый инструмент, то возможны два основных варианта трактовки эффективности: эффективность инструмента как такового и эффективность использования этого инструмента в некоторой вышестоящей системе.

Рассмотрение эффективности ЭВМ как таковой говорит и о потенциальной эффективности ЭВМ для пользователя в общем случае.

Отдельными показателями технической эффективности ЭВМ могут служить функциональная эффективность, энергетическая эффективность и эксплуатационная эффективность.

Можно сформулировать интегральный критерий технической эффективности ЭВМ, учитывающий различные ее показатели: производительность, объемы хранимой информации, надежность, габариты, вес и др. А соответствующими коэффициентами в этом критерии можно учесть вес (значимость) того или иного параметра для общей оценки эффективности ЭВМ применительно к анализируемой ситуации.

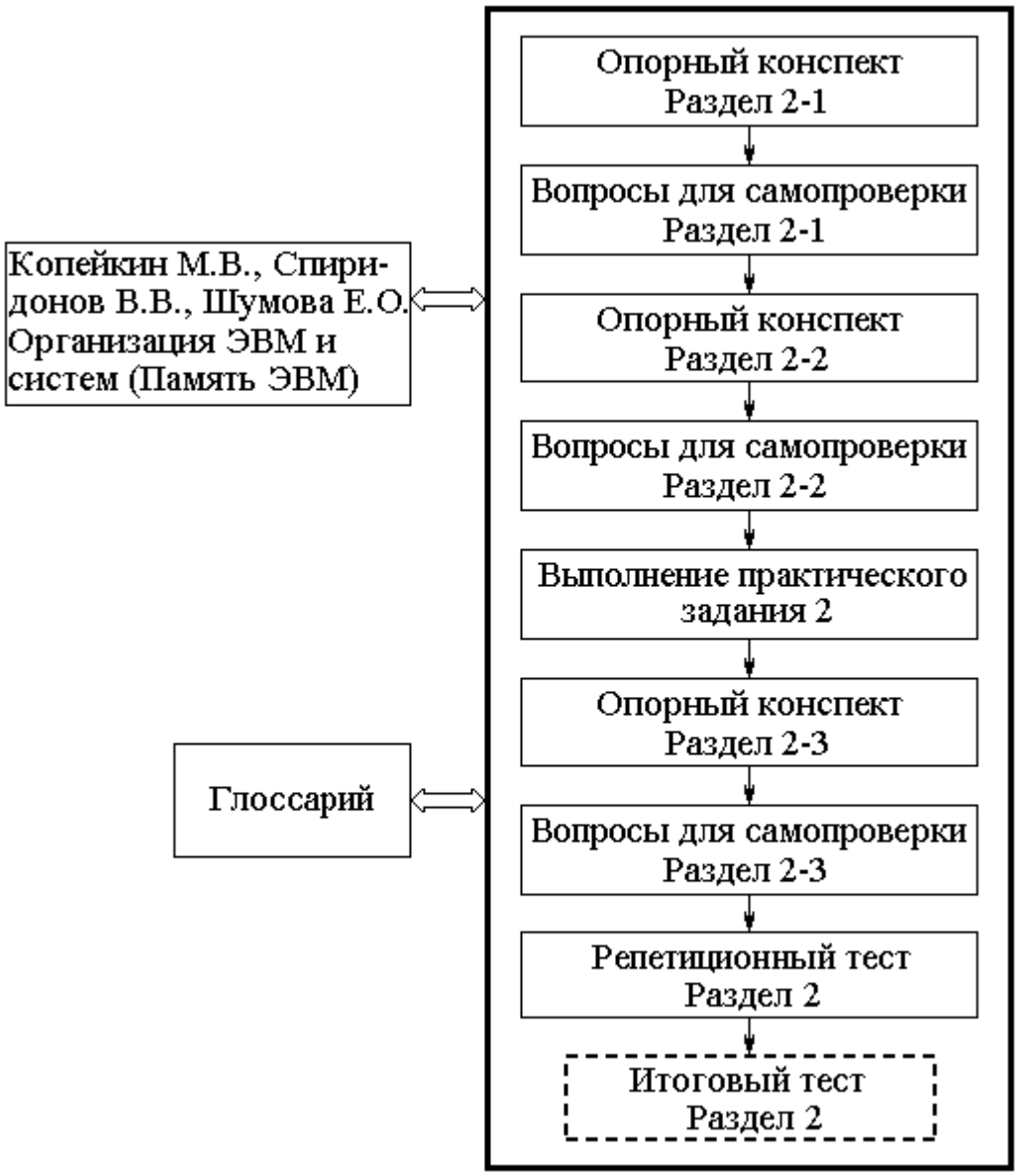
При второй трактовке эффективность ЭВМ определяют сопоставлением результатов от функционирования ЭВМ и затрат всех видов ресурсов. Хотя оценить долю результатов вышестоящей системы, приходящихся непосредственно на ЭВМ, может оказаться не так просто.

В общем случае также можно сформулировать интегральный критерий эффективности, учитывающий и различные частные показатели ЭВМ, и результаты функционирования включающей ее системы.

Вопросы для самопроверки по теме 1.2

1. Какие основные типы функций реализуются в ЭВМ?
2. Назовите примеры противоречий между характеристиками ЭВМ
3. В чем преимущества программной реализации функций? В чем – аппаратной?
4. Как связаны частота передачи и разрядность интерфейса?
5. Какие основные компоненты используются в моделях массового обслуживания?
6. В каких случаях используется имитационное моделирование?
7. Как можно оценивать эффективность ЭВМ?

Схема работы с разделом 2



Раздел 2. ЗАПОМИНАЮЩИЕ УСТРОЙСТВА ЭВМ

Второй раздел курса включает три темы: "***Основные характеристики и типы запоминающих устройств ЭВМ"***, "***Оперативные и сверхоперативные ЗУ"*** и "***Организация ЗУ различных типов"***. После изучения каждой темы Вам следует ответить на вопросы для самопроверки.

Работа с разделом 2 завершается сдачей контрольного теста. Кроме того, по разделу выполняется практическое задание 2.

Для того, чтобы Вы смогли успешно ответить на вопросы контрольного теста, Вам предоставляется возможность поработать с репетиционным тестом. Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к главе 5 учебника [1] или к учебному пособию [7].

2.1. Основные характеристики и типы запоминающих устройств ЭВМ

При изучении данной темы Вы должны рассмотреть основные виды запоминающих устройств, используемых в ЭВМ, их характеристики и особенности организации.

Для проверки изучения материала темы Вам предстоит ответить на вопросы для самопроверки.

Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к главе 5 учебника [1] или к главе 1 учебного пособия [7].

2.1.1. Классификация запоминающих устройств

Память ЭВМ почти всегда является "узким местом", ограничивающим производительность компьютера. Поэтому в ЭВМ и системах используется большое количество различных типов ЗУ.

Возможная классификация ЗУ представлена на рис. 2.1. В ней ЗУ подразделяются по функциональному назначению и принципу организации.

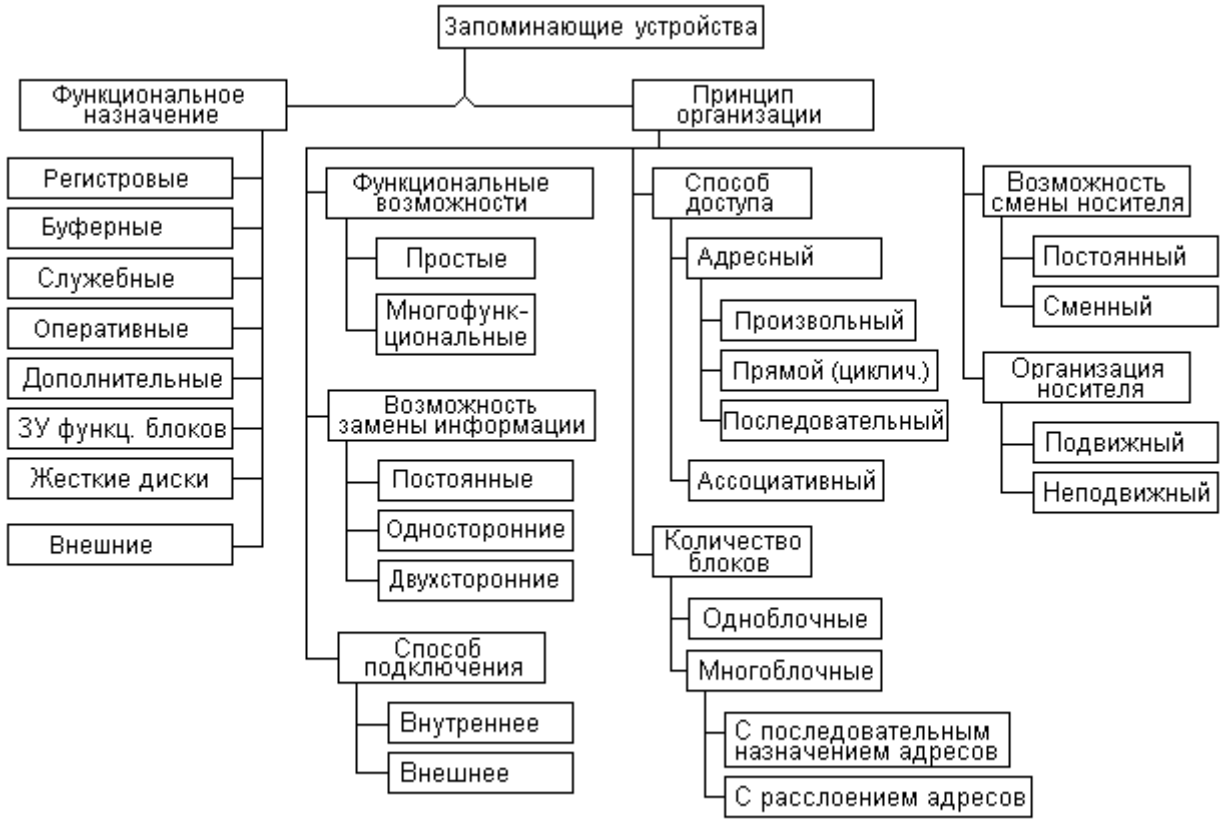


Рис. 2.1. Классификация запоминающих устройств

Классификация ЗУ по функциональному назначению

Общий вид иерархии памяти ЭВМ представлен на рис. 2.2. Не все из представленных на нем ЗУ обязательно входят в состав ЭВМ, а характер связей между устройствами может отличаться от показанного.

1. *Регистровые ЗУ* входят в состав процессора и часто рассматриваются не как самостоятельный блок ЗУ, а просто как набор регистров процессора. Они позволяют сократить время выполнения программ за счет использования

команд типа регистр-регистр и уменьшить частоту обменов информацией с более медленными ЗУ ЭВМ.

2. Назначение *буферных ЗУ* состоит в сокращении времени передачи информации между процессором и более медленными уровнями памяти компьютера. Буферная память может устанавливаться на различных уровнях, но ЗУ рассматриваемого уровня ранее в отечественной литературе называли сверхоперативными. Сейчас это название практически полностью вытеснил термин "кэш-память" или просто *кэш*.

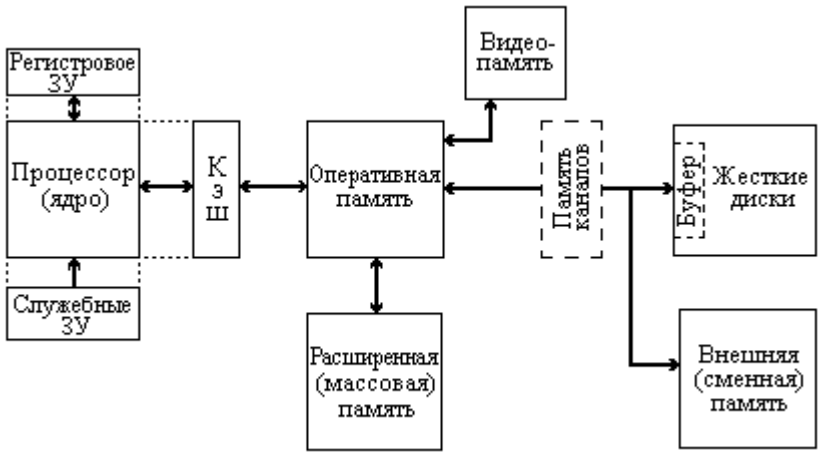


Рис. 2.2. Возможный состав системы памяти ЭВМ

3. С*лужебные ЗУ* имеют различное назначение. Примерами их служат ЗУ микропрограмм, называемые также управляющей памятью; вспомогательные ЗУ, используемые для управления многоуровневой памятью и др.

4. *Оперативное ЗУ* (ОЗУ) является основным запоминающим устройством ЭВМ, в котором хранятся выполняемые в настоящий момент процессором программы и обрабатываемые данные, резидентные программы, модули операционной системы и т. п.

5. *Дополнительная память*, иногда называемая расширенной или массовой, могла использоваться для наращивания емкости оперативной памяти с помощью подключения более дешевого и емкого, чем ОЗУ, запоминающего устройства.

6. *ЗУ, принадлежащие отдельным функциональным блокам* компьютера, обычно используются для буферизации данных, извлекаемых из каких-либо устройств и поступающих в них. Типичными примерами таких ЗУ являются *видеопамять* графического адаптера, *буферная память* контроллеров жестких дисков и др.

7. *Жесткие диски* хранят практически всю информацию, используемую более или менее активно, – от операционной системы и основных прикладных программ до редко используемых пакетов и справочных данных.

8. Все остальные запоминающие устройства можно объединить с точки зрения функционального назначения в одну общую группу, охарактеризовав ее как группу *внешних ЗУ*. Под словом “внешние” подразумевают то, что информация, хранимая в этих ЗУ, расположена на носителях, не являющихся частью собственно ЭВМ. Под это определение подпадают гибкие диски, компакт дис-

ки, накопители на сменных магнитных и магнитооптических дисках, внешние диски, разнообразные флэш-модули и карты, стримеры и др.

Классификация ЗУ по принципу организации

Особенности организации ЗУ определяются используемыми технологиями, логикой функционирования и некоторыми другими факторами, перечисляемыми ниже.

1. По *функциональным возможностям* ЗУ можно разделять:

- на простые, допускающие только хранение информации;
- многофункциональные, которые позволяют не только хранить, но и перерабатывать хранимую информацию без участия процессора, непосредственно в самих ЗУ.

2. По *возможности изменения информации* различают ЗУ:

- постоянные (или с однократной записью);
- односторонние (с перезаписью или перепрограммируемые);
- двусторонние.

3. По *способу доступа* различают ЗУ:

- с адресным доступом;
- с ассоциативным доступом.

При *адресном доступе* для записи или чтения место расположения информации в ЗУ определяется ее адресом. В зависимости от того, как работает механизм доступа, различают произвольный, прямой (циклический) и последовательный виды адресного доступа.

При *ассоциативном доступе* место хранения информации при чтении и записи определяется не адресом, а значением некоторого ключа поиска.

4. По *организации носителя* различают ЗУ:

- с неподвижным носителем;
- с подвижным носителем.

В первых из них носитель механически неподвижен в процессе чтения и записи информации. Обращения к ЗУ второй группы сопровождаются механическим перемещением носителя, как, например, в жестких и гибких дисках.

5. По *возможности смены носителя* ЗУ могут быть:

- с постоянным носителем;
- со сменным носителем.

В ЗУ первого вида носитель является частью самого устройства (оперативные ЗУ, жесткие диски). В ЗУ второй группы носитель может устанавливаться в ЗУ и извлекаться из него в процессе работы (гибкие диски, CD-ROM-дисководы, карты памяти и др.).

6. По *способу подключения* к системе ЗУ делятся:

- на внутренние (стационарные);
- внешние (съемные или сменные).

7. По *количеству блоков*, образующих модуль или ступень памяти, можно различать:

- одноблочные ЗУ;

- многоблочные ЗУ.

Если в многоблочных ЗУ блоки (или банки памяти) допускают возможность параллельной работы, то это позволяет повысить общую производительность модуля (ступени) ЗУ.

Приведенную классификацию можно дополнить и другими признаками такими, как физические принципы реализации, потребляемая мощность и пр.

2.1.2. Основные характеристики запоминающих устройств

Запоминающие устройства (ЗУ) характеризуются рядом параметров, определяющих возможные области применения различных типов таких устройств. К основным параметрам относятся их информационная емкость (*E*), время обращения (*T*) и стоимость (*C*).

Под информационной емкостью ЗУ понимают количество информации, измеряемое в байтах, килобайтах, мегабайтах или гигабайтах, которое может храниться в запоминающем устройстве. Обычно информационная емкость учитывает только полезный объем хранимой информации, не включающий объем памяти, расходуемый на служебную информацию, контрольные разряды или байты, резервные области, дорожки синхросигналов и пр.

Время обращения к ЗУ различных типов определяется по-разному. В качестве примера можно рассмотреть оперативные ЗУ и жесткие диски.

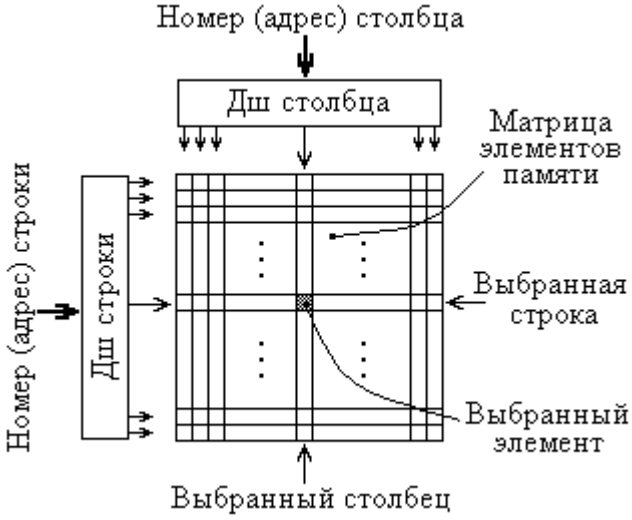


Рис. 2.3. Произвольный доступ к массиву элементов памяти

Оперативные ЗУ обычно реализуются как ЗУ с произвольным доступом. Это означает, что доступ к данным, физически организованным в виде двумерного массива (матрицы элементов памяти), производится с помощью схем дешифрации, выбирающих нужные строку и столбец массива по их номерам (адресам), как показано на рис.2.3. Поэтому время $T_{обр}$ обращения к ним определяется, в случае отсутствия дополнительных этапов (таких, например, как передача адреса за два такта), временем срабатывания схем дешифрации адреса и собственно временами записи или считывания данных.

Процесс обращения (чтения или записи) к жесткому диску показан на рис.2.4. Он включает в себя 3 этапа: перемещение блока головок чтения/записи на нужную дорожку (*а*), ожидание подхода требуемого сектора под головки чтения/записи (*б*) и передача данных с диска или на него (*в*). Все этапы связаны с механическими перемещениями, поэтому их времена сравнительно велики и составляют величины порядка единиц миллисекунд.

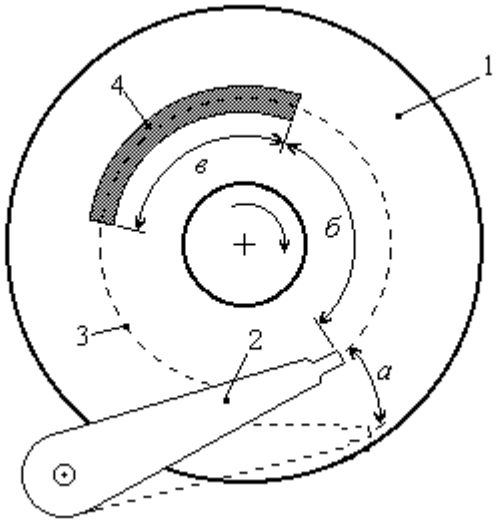


Рис. 2.4. Обращение к жесткому диску
(1 – пластина диска, 2 – блок головок чт/зп, 3 – дорожка (цилиндр), 4 – файл;
а – поиск дорожки (перемещение блока головок чт/зп, б – ожидание подхода файла под блок головок, *в* – передача данных)

Стоимость запоминающих устройств также представляет собой важную характеристику. Именно она является одной из причин иерархической организации памяти ЭВМ. Определения дорогие и дешевые понимаются в относительном измерении, исходя из стоимости хранения единицы информации (удельной стоимости) в ЗУ.

Используются и другие характеристики памяти: надежность, энергопотребление, габариты, способность сохранять информацию при отключении питания и др.

Вопросы для самопроверки по теме 2.1

1. Для чего используется кэш-память?
2. В качестве чего используются ЗУ с произвольным доступом?
3. Что дает использование иерархии ЗУ?
4. Чем определяется скорость передачи данных для жесткого диска?
5. Что такое ЗУ с последовательным доступом?
6. Что такое ассоциативные ЗУ? Для каких целей их применяют?
7. Что дает использование схем контроля в жестких дисках?

2.2. Оперативные и сверхоперативные ЗУ

При изучении данной темы Вы должны рассмотреть особенности организации и принципы работы оперативных и сверхоперативных ЗУ ЭВМ, основные их виды, особенности структуры и взаимодействия между собой и с другими устройствами ЭВМ.

Для проверки изучения материала темы Вам предстоит ответить на вопросы для самопроверки. По данной теме также выполняется практическое задание 2, указания к выполнению которого даны в разделе методических указаний 3.4.

При затруднениях в ответе на какой-либо вопрос следует обратиться к главе 5 учебника [1] или к материалам главы 2 учебного пособия [7].

2.2.1. Оперативные ЗУ

Оперативное ЗУ (ОЗУ) является основным запоминающим устройством ЭВМ, в котором хранятся выполняемые в настоящий момент процессором программы и обрабатываемые данные, резидентные программы, модули операционной системы и т.п. В англоязычной литературе для ОЗУ также используется термин *RAM* (*random access memory*), означающий память с произвольным доступом. Информация, находящаяся в ОЗУ, непосредственно доступна командам процессора.

Оперативная память часто организована в виде нескольких блоков, которые могут работать параллельно. Это делается для повышения ее пропускной способности (производительности). Причем параллельность работы возможна как внутри одного модуля памяти за счет наличия в нем нескольких банков, так и за счет одновременной работы нескольких модулей.

Оперативная память строится на полупроводниковых интегральных схемах. В качестве элементов памяти в них используются триггеры (статические ЗУ) или конденсаторы (динамические ЗУ).

Организация БИС ЗУ с произвольным доступом

Первые полупроводниковые оперативные ЗУ строились на схемах малой и средней степени интеграции и включали в себя несколько различных типов микросхем: собственно матрицы элементов памяти, усилители чтения-записи, дешифраторы и, при необходимости, регистры (адреса и данных).

С появлением больших интегральных схем (БИС) и повышением частоты их работы микросхемы памяти стали включать в себя не только элементы памяти, но и всю остальную электронику управления: дешифраторы, усилители, буферные регистры, схемы управления.

На функциональных схемах микросхема памяти изображается прямоугольником с левым и правым полями, как показано на рис. 2.5.

Микросхема имеет три группы входов: адресные входы, вход(ы) данных и управляющие входы. Количество адресных входов ($A_0 \div A_k$) определяется емкостью и организацией микросхемы памяти, а также способом подачи адреса. Емкость микросхемы E_{Cx}, равная произведению количества адресов (слов) N на разрядность хранимых слов n, не определяет однозначно требуемое число адресных входов. Для адресации любого из N слов требуется адрес разрядностью $\log_2 N$. Например, для адресации микросхемы емкостью E_{Cx} = 128 Мбит, имеющей организацию 16М × 8 (адресов × бит), достаточно $\log_2 16\text{M} = \log_2 (2^4 \times 2^{20}) = 24$ разряда.

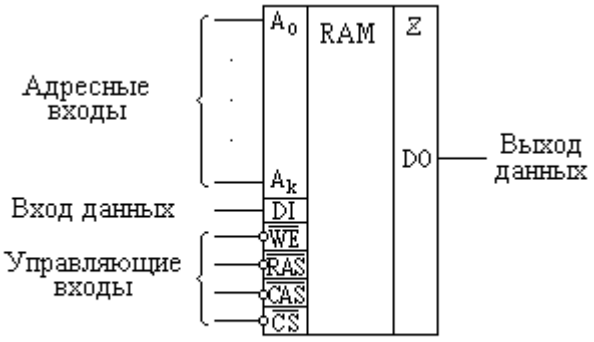


Рис. 2.5. Условное изображение ОЗУ на функциональных схемах

Способ подачи адреса также оказывает влияние на количество адресных входов микросхемы. В динамических оперативных ЗУ используется мультиплексирование адресных входов с поочередной подачей на них сначала старшей части (половины) адреса – адреса строки (*Row Address*), а затем – младшей части – адреса столбца (*Column Address*). В статических ЗУ все разряды адреса подаются на адресные входы одновременно.

Количество входов данных (*DI – Data Input*) и выходов данных (*DO – Data Output*) равно разрядности хранимых слов. Часто входы и выходы данных объединяются, что позволяет уменьшить вдвое количество выводов данных, а также упростить их подключение к шинам данных.

Выходы данных (или объединенные входы/выходы) имеют специальный выходной каскад – выход с тремя устойчивыми состояниями (*z*-выход) или выход с открытым коллектором. Тип выхода отмечается значком в верхней части правого поля изображения микросхемы. На рис. 2.5 показан *z*-выход.

Управляющие входы могут заметно различаться как по назначению, так и по обозначениям для разных типов микросхем памяти.

Во всех случаях присутствует вход управления режимом обращения – чтение или запись. Частым его обозначением является ***WE#*** (*Write Enable* – разрешение записи). Вход этот обычно инверсный (это и означает символ #), т. е. режим записи включается при нулевом значении сигнала на данном входе, а при единице на входе производится чтение.

Другим общим сигналом, имеющимся почти во всех микросхемах, является сигнал выбора микросхемы – ***CS#*** (*Chip Select*). Этот вход также обычно является инверсным и при единичном значении на нем микросхема переходит в “выключенное” состояние (выход данных микросхемы переходит в состояние высокого выходного сопротивления, если он является z-выходом, или в состояние “1”, если это инверсный выход с открытым коллектором). При нулевом значении сигнала на входе ***CS#*** микросхема находится в активном состоянии.

В динамических ОЗУ используются два управляющих входа сигналов строба – ***RAS#*** (*Row Address Strobe* – строб адреса строки) и ***CAS#*** (*Column Address Strobe* – строб адреса столбца). Сигналы на этих входах переводятся в активное состояние (в “0”) в тот момент, когда на адресных входах установлен адрес строки или адрес столбца соответственно.

Статические ЗУ с произвольным доступом

Элемент памяти статических ЗУ (*Static Random Access Memory* – SRAM) - триггер, конечно, сложнее, чем конденсатор с транзисторным ключом динамического ЗУ. Поэтому статические ЗУ обладают меньшей плотностью хранения информации, например емкость типовых микросхем статических ЗУ начала 2000-х годов не превосходила 16 Мбит.

Однако триггер является самым быстродействующим элементом памяти. Поэтому статическая память позволяет достичь наибольшего быстродействия.

Главными недостатками статической памяти являются ее относительно высокие стоимость и энергопотребление. Конечно, в зависимости от используемой технологии память будет обладать различным сочетанием быстродействия и потребляемой мощности. Например, статическая память, изготовленная по КМОП-технологии (CMOS память), имеет большое время доступа, но отличается малым энергопотреблением. В ПЭВМ ее применяют для хранения конфигурационной информации компьютера при выключенном напряжении сети.

По особенностям функционирования различают асинхронную (*Asynchronous*), синхронную пакетную (*Synchronous Burst*) и синхронную конвейерно-пакетную (*Pipeline Burst*) статическую память.

Первой появилась асинхронная память. Интерфейс этой памяти включает шины данных, адреса и управления, а в состав управляющих сигналов входят сигнал выбора микросхемы ***CS***# (*Chip Select),*сигнал разрешения записи ***WE***# (*Write Enable*) и сигнал включения выходов для выдачи данных ***OE***# (*Output Enable*). Все сигналы управления инверсные, т. е. их активный (вызывающий соответствующее действие) уровень низкий. Временн***ы***е диаграммы циклов чтения и записи (один из возможных вариантов) приведены на рис. 2.6.

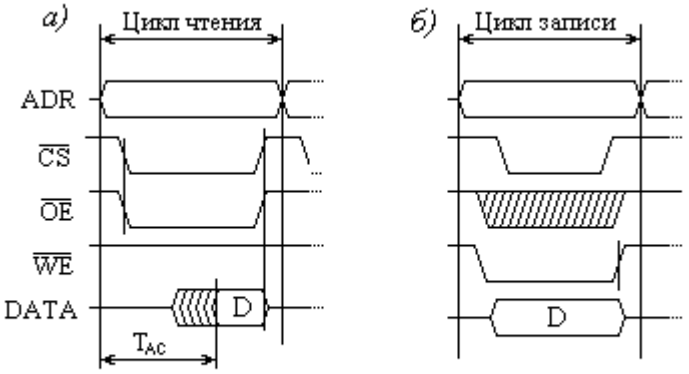


Рис. 2.6. Временна́я диаграмма простых циклов чтения *а)* и записи *б)* асинхронной статической памяти

Время доступа t_{AC} у типовых микросхем составляет порядка 10 нс. Поэтому реально такие микросхемы могут работать на частотах системной шины, только если эти частоты не превышают 66 МГц.

Позже появилась синхронная пакетная статическая память (SBSRAM), ориентированная на выполнение пакетного обмена информацией, который характерен для кэш-памяти. Эта память включает в себя внутренний счетчик адреса, предназначенный для перебора адресов пакета, и использует сигналы

синхронизации ***CLK***. Затем была разработана конвейерно-пакетная память PBSRAM, обеспечивающая еще более высокое быстродействие, чем SBSRAM.

Динамические полупроводниковые ЗУ с произвольным доступом

В качестве оперативных ЗУ в настоящее время чаще используются динамические ЗУ с произвольным доступом (DRAM). Такое положение обусловлено тем, что необходимость регенерации информации в таких ЗУ и относительно невысокое их быстродействие компенсируются б***о***льшим объемом микросхем этих ЗУ, а также низкой их стоимостью.

Первые такие ЗУ, которые позже стали называть асинхронными динамическими ОЗУ, выполняли операции чтения и записи, получив лишь запускающий сигнал (обычно сигнал строба адреса) без каких-либо внешних синхронизирующих сигналов. Диаграмма простых (не пакетных) циклов чтения и записи таких ЗУ представлена на рис. 2.7, *а* и *б*, соответственно. Любой цикл (чтения или записи) начинается по спаду (фронту "1" → "0") сигнала ***RAS#***.

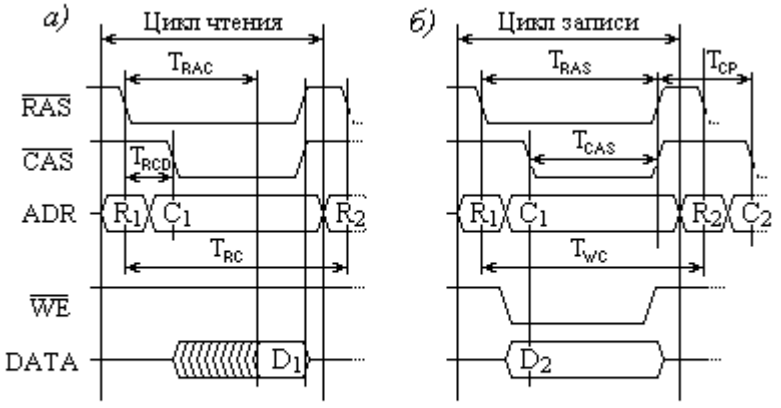


Рис. 2.7. Временна́я диаграмма простых циклов чтения *а)* и записи *б)* (асинхронной) динамической памяти

Адрес на шины адреса поступает двумя частями: адрес строки (R_1 или R_2) и адрес столбца (C_1 и C_2). В момент, когда на адресной шине установилось требуемое значение части адреса, соответствующий сигнал строба (***RAS#*** или ***CAS#***) переводится в активное (нулевое) состояние.

В цикле чтения (сигнал ***WE#*** в единичном состоянии) после подачи адреса строки и перевода сигнала ***CAS#*** в нулевое состояние начинается извлечение данных из адресованных элементов памяти, что показано на диаграмме сигнала ***DATA*** как заштрихованная часть. По истечении времени доступа T_{RAC} (*RAS Access Time* – задержка появления данных на выходе ***DATA*** по отношению к моменту спада сигнала ***RAS#***) на шине данных устанавливаются считанные из памяти данные. Затем по истечении времени, достаточного для фиксации данных, сигналы ***RAS#*** и ***CAS#*** переводятся в единичное состояние, что указывает на окончание цикла обращения к памяти.

Цикл записи начинается так же, как и цикл чтения, по спаду сигнала ***RAS#*** после подачи адреса строки. Записываемые данные выставляются на шину данных одновременно с подачей адреса столбца, а сигнал разрешения записи ***WE#*** при этом переводится в нулевое состояние. По истечении времени, доста-

точного для записи данных в элементы памяти, сигналы данных, ***WE#, RAS#*** и ***CAS#*** снимаются, что говорит об окончании цикла записи.

Помимо названного параметра T_{RAC} – времени доступа по отношению к сигналу **RAS#** на диаграмме на рис. 2.7 указаны:
T_{RCD} – минимальное время задержки между сигналами ***RAS#*** и ***CAS#*** (*RAS-to-CAS Delay*);
T_{RAS} и T_{CAS} – длительности (активного уровня) сигналов ***RAS#*** и ***CAS#***;
T_{RC} и T_{WC} – длительности циклов чтения и записи соответственно;
T_{RP} и T_{CP} – времена подзаряда строки и столбца соответственно (эти времена определяют минимальную задержку перед подачей повторного сигнала ***RAS#*** или ***CAS#***).

Подача адреса двумя частями удлиняет цикл обращения к памяти. Но большинство обращений к оперативной памяти выполняется при обмене с внешней памятью и кэш-памятью и производится по последовательным адресам. Адрес строки, являющийся старшей частью адреса, для последовательных адресов памяти одинаков (исключение составляет переход через границу строки). Это позволяет в серии (пакетном цикле) обращений по таким адресам задать адрес строки только для первого обращения, а для всех последующих задавать только адрес столбца. Такой способ был назван FPM (*Fast Page Mode* – быстрый страничный режим) и обеспечивал сокращение времени обращения к памяти для всех циклов пакета, кроме первого.

Следующей модификацией асинхронной динамической памяти стала память EDO (*Extended Data Output* – растянутый выход данных), в которой на выходе был установлен буфер-защелка, фиксирующий данные после их извлечения из памяти. Это позволило сократить длительность сигнала ***CAS#*** и циклов памяти до соотношения с циклами системной шины 5-2-2-2 (5 циклов шины на первое обращение и по 2 – на последующие).

Позже появилась еще одна модификация асинхронной DRAM – BEDO (*Burst EDO* – пакетная EDO память), в которой и адрес столбца подавался только в первом цикле пакета, а в следующих циклах адреса столбцов формировались внутренним счетчиком.

Затем на смену асинхронной памяти пришла синхронная – SDRAM.

Синхронная динамическая память имеет большее быстродействие, чем асинхронная, при использовании аналогичных элементов памяти. Основные сигналы интерфейса SDRAM схожи с сигналами интерфейса асинхронной памяти. Главные их отличия сводятся к появлению ряда новых сигналов, в первую очередь синхросигнала ***CLK***, по переднему фронту которого производятся все переключения в микросхеме.

Кроме того, SDRAM память сразу ориентирована на выполнение *пакетных передач* данных, причем длина пакета задается при инициализации микросхем памяти, хотя может быть программно изменена позднее.

Выигрыш в производительности SDRAM достигается за счет более гибкого управления процессами чтения и записи, возможности задания параметров и лучших алгоритмов работы контроллера памяти.

Основные временн***ы***е характеристики синхронной динамической памяти обозначают частотой синхронизации в мегагерцах с префиксом PC: PC100, PC133, PC167 и тремя числами, например 2-2-2 или 3-2-2 (называемыми *тай-*

мингами). Эти числа означают соответственно выраженное в количестве циклов синхронизации минимальное время: 1) между сигналами ***RAS#*** и ***CAS#***, обозначаемое t_{RCD}, 2) от задержки появления данных после подачи сигнала ***CAS#*** , или t_{CK} (*CAS Latency*), 3) необходимое на подзаряд строки, т. е. время t_{RP}.

Дальнейшее развитие синхронной динамической памяти пошло по пути повышения частоты синхронизации и скорости передачи данных.

Первым шагом в этом направлении стала память DDR SDRAM, обеспечивающая двойную скорость передачи данных (*DDR – Double* или *Dual Data Rate*), в которой за один такт осуществляются две передачи данных – по переднему и заднему фронтам каждого синхроимпульса. Во всем остальном эта память работает аналогично обычной SDRAM памяти (которую стали иногда называть SDR SDRAM – *Single Data Rate*).

Производительность DDR SDRAM при этом получается не в два раза выше, чем у обычной SDRAM, так как ускорение касается только собственно передачи данных, а основные задержки остаются теми же, поскольку элементы памяти в микросхемах SDR и DDR SDRAM работают на одинаковой частоте.

Развитием DDR SDRAM является стандарт DDR2. В нем обеспечивается учетверенная скорость передачи данных по отношению к частоте работы самих элементов памяти. На более высокой частоте времена задержек t_{RCD}, *CAS Latency* и t_{RP} в таких микросхемах требуют большего количества циклов, что также не дает удвоения производительности по сравнению с DDR памятью.

В настоящее время изготавливается и память стандарта DDR3.

Конденсаторы запоминающих элементов динамической памяти разряжаются из-за наличия токов утечки. Время, в течение которого информация сохраняется в элементе памяти, составляет до нескольких десятков миллисекунд. Это приводит к необходимости периодического (с периодом не больше, чем время сохранения информации) восстановления зарядов емкостей. Такая процедура и получила название регенерации (*refresh*) динамической памяти. Выполняется она одновременно для целой строки матрицы (банка) элементов памяти, поскольку регенерировать информацию по элементам (битам) или по словам (по 8 байтов) слишком долго.

Модули оперативных ЗУ

Оперативные запоминающие устройства всегда были ресурсом, допускающим увеличение емкости, а иногда и сокращение времени обращения с целью повышения общей производительности ЭВМ. Совершенствование технологий изготовления оперативных ЗУ привело к тому, что стойки памяти середины 1970-х годов емкостью 512 Кбайт, размером с двустворчатый платяной шкаф сменили маленькие платы размером с зажигалку. А наращивание или замену оперативной памяти ЭВМ, предполагавшие в то время проведение достаточно серьезных монтажных работ, теперь в течение 5 – 10 минут может провести даже пользователь.

Модули динамических оперативных ЗУ различаются типом и расположением используемых микросхем, емкостью, быстродействием, количеством и

расположением контактов. Имеются также и другие различия, в частности в возможности контроля хранимых данных и буферизации данных.

Для определения объема и типа установленной памяти после включения компьютера используют последовательный способ идентификации (*Serial Presence Detect - SPD*), при котором на плату модуля устанавливается специальная дополнительная микросхема, так называемый SPD-чип, представляющая собой небольшую постоянную память на 128 или 256 байтов с последовательным интерфейсом доступа. В этой микросхеме в стандартном формате записана информация об изготовителе микросхемы и ее параметрах.

2.2.2. Кэш-память (сверхоперативные ЗУ)

Кэш-память служит для сокращения времени передачи информации между процессором и более медленными уровнями памяти компьютера (обычно – оперативными ЗУ). Ранее такие буферные ЗУ в отечественной литературе называли сверхоперативными.

Принцип использования буферной памяти во всех случаях сводится к одному и тому же. Буфер представляет собой более быстрое (и более дорогое), но менее емкое ЗУ, чем то, для ускорения работы которого он предназначен. При этом в буфере размещается только та часть информации из более медленного ЗУ, которая используется в настоящий момент. Если доля h обращений к памяти со стороны процессора, удовлетворяемых непосредственно буфером (кэшем) высока (0,9 и более), то среднее время для всех обращений оказывается близким ко времени обращения к кэшу.

Пусть двухуровневая память состоит из кэш- и оперативной памяти, как показано на рис. 2.8, где t_c – время обращения к кэш-памяти, t_m – время обращения к ОП, h – доля обращений, обслуживаемых кэш-памятью,1 – h – доля обращений, обслуживаемых ОП. И пусть время обращения к кэшу t_c = 1 нс (10^{-9} с), время t_m обращения к оперативной памяти в десять раз больше – t_m = 10 нс, а доля обращений, удовлетворяемых кэшем, h = 0,95.

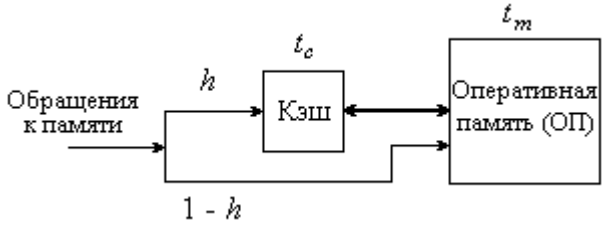


Рис. 2.8. К расчету среднего времени обращения

Тогда среднее время обращения к такой двухуровневой памяти T_{cp} составит T_{cp} = 1 * 0,95 + 10 * (1 – 0,95) = 1,45 нс, т. е. всего на 45 % больше времени обращения к кэшу. Значение h зависит от размера кэша и характера выполняемых программ и иногда называется отношением успехов, или попаданий (*hit ratio*).

Эта модель памяти, состоящей из двух ступеней, применима к любому случаю. Но хотя задачи управления иерархией памяти для разных уровней одинаковы по содержанию, реализация их различна, в первую очередь, из-за отличий в быстродействии и информационных емкостях разных уровней.

Кэш-память может состоять из двух (и даже трех) уровней – первого (L1) и второго (L2), отличающихся своей емкостью и временем обращения и конструктивно входящих в микросхему процессора.

Время обращения к кэш-памяти, обычно работающей на частоте процессора, составляет от десятых долей до единиц наносекунд, т. е. не превышает длительности одного цикла процессора.

Обмен информацией между кэш-памятью и более медленными ЗУ для улучшения временных характеристик выполняется блоками. Управляют этим обменом аппаратные средства процессора и операционная система.

В связи с высокими скоростями работы перечисленных устройств управление кэш-памятью должно обеспечить решение ряда задач, связанных:

- с быстрым *определением местоположения требуемой информации* в двухуровневом фрагменте (кэш L1–кэш L2 или кэш L2 – оперативная память) системы памяти;
- *выбором информации, которую можно удалить* из верхнего уровня при необходимости занесения в него новой информации и отсутствии в нем свободного места;
- *поддержанием соответствия между копиями* одной и той же информации, располагающейся в разных ступенях памяти.

Последнее иначе называют когерентностью данных, используя аналог физического термина.

Поскольку в кэш-памяти в каждый конкретный момент хранится только часть информации, размещенной в запоминающем устройстве более низкого уровня (для определенности – пусть это оперативная память), то при обращении к этому запоминающему устройству (со стороны процессора или другого узла) необходимо определить, находится ли копия требуемой информации в кэш-памяти. Если она там есть, то обращение может быть быстро обслужено кэш-памятью, в противном случае информацию придется извлекать из оперативной памяти (или заносить в нее), что займет примерно на порядок большее время.

Определять, имеется ли запрошенная информация в кэш-памяти или нет, приходится в процессе обращения к памяти, поэтому время данной операции должно быть существенно меньше собственно времени обращения к кэш-памяти. Память с ассоциативным доступом, позволяющая сделать это быстро, слишком дорога. Поэтому используют специальные ограничения на место расположения информации в кэш-памяти. Так реализованы *кэш прямого отображения* и *наборно-ассоциативный кэш*.

Вопросы для самопроверки по теме 2.2

1. В каких случаях используются динамические ЗУ? В каких статические?
2. Сколько уровней может иметь кэш-память?
3. Что такое регенерация информации в динамических ОЗУ?
4. На сколько различается быстродействие элементов памяти DDR и DDR2 SDRAM?
5. Что такое тайминги динамических ОЗУ?
6. Что такое наборно-ассоциативный кэш?
7. Чем различаются кэш прямого отображения и наборно-ассоциативный кэш?
8. Как поддерживается соответствие информации в кэш-памяти и оперативной памяти?

2.3. Организация ЗУ различных типов

При изучении данной темы Вы должны рассмотреть запоминающие устройства, используемые в качестве вспомогательных и внешних ЗУ ЭВМ, их характеристики и особенности организации, а также перспективы развития различных ЗУ.

Для проверки изучения материала темы Вам предстоит ответить на вопросы для самопроверки.

Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к главе 5 учебника [1] или к материалам глав 2 и 3 учебного пособия [7].

2.3.1. Постоянные ЗУ

Наибольшее распространение получили полупроводниковые постоянные ЗУ, но существуют и другие принципы реализации постоянных ЗУ.

Запись информации в постоянные ЗУ отличается от считывания по способу и времени выполнения. Процесс записи для полупроводниковых постоянных ЗУ получил также название “прожига”, или программирования, первое из которых связано со способом записи, сводящимся к разрушению (расплавлению, прожигу) соединительных перемычек в ЗУ.

В полупроводниковых ПЗУ в качестве элементов памяти, точнее, в качестве нелинейных коммутирующих и усилительных элементов обычно используются транзисторы. Они объединены в матрицу, выборка данных из которой производится, так же, как и в других ЗУ с произвольным доступом.

Различают две большие группы ПЗУ: *программируемые изготовителем* и *программируемые пользователем*.

В ЗУ первой группы, называемые иначе масочными, информация заносится в процессе изготовления с помощью формирования соответствующей конфигурации соединений. Эти ЗУ самые дешевые при массовом изготовлении.

В ПЗУ, программируемые пользователем, информация записывается после их изготовления самими пользователями. Существуют два основных типа таких ЗУ: *однократно программируемые* и *перепрограммируемые*.

В однократно программируемые ПЗУ запись производится посредством разрушения соединительных перемычек между выводами транзисторов и шинами матрицы (хотя есть и несколько иные технологии).

Перепрограммируемые ПЗУ позволяют записывать информацию многократно. В них обычно используются МОП-транзисторы с составным (или “плавающим”) затвором, который способен накапливать заряд, снижающий пороговое напряжение отпирания транзистора, и сохранять этот заряд при выключенном питании. Программирование таких ПЗУ состоит в создании зарядов на затворах тех транзисторов, где должны быть записаны данные (обычно “0”, так как в исходном состоянии в таких микросхемах записаны все “1”). Перед повторной записью требуется стереть ранее записанную информацию либо электрически, либо ультрафиолетовым светом.

2.3.2. Флэш-память

Флэш-память, появившаяся в конце 1980-х годов (*Intel*), относится к перепрограммируемым ЗУ с электрическим стиранием. Однако стирание в ней осуществляется сразу целой области ячеек – блока или всей микросхемы. Это обеспечивает более быструю запись информации. Для упрощения данной процедуры в микросхему включаются специальные блоки, делающие запись "прозрачной" (подобной записи в обычное ЗУ) для аппаратного и программного окружения.

Флэш-память строится на однотранзисторных элементах памяти (с "плавающим" затвором), что обеспечивает плотность хранения информации даже несколько выше, чем в динамической оперативной памяти. Различные технологии построения базовых элементов флэш-памяти отличаются количеством слоев, методами стирания и записи данных, а также структурной организацией. Наиболее широко известны NOR и NAND типы флэш-памяти, запоминающие транзисторы которых подключены к разрядным шинам соответственно параллельно и последовательно.

Первый тип имеет относительно большие размеры ячеек и быстрый произвольный доступ (порядка 70 нс), что позволяет выполнять программы непосредственно из этой памяти. Второй тип имеет меньшие размеры ячеек и быстрый последовательный доступ (обеспечивая скорость передачи до 16 Мбайт/с), что более пригодно для построения устройств блочного типа, например твердотельных дисков.

К минусам данного вида памяти можно отнести относительно невысокую скорость передачи данных и высокую стоимость устройств большой емкости.

2.3.3. Специальные ЗУ

Существуют различные ЗУ, представляющие собой модификации описанных выше типов, либо использующие иные технологические и физические принципы организации, либо обладающие дополнительными функциональными возможностями.

К модификациям типовых ЗУ можно отнести разновидности графической памяти: видео ОЗУ (*Video RAM*), оконное ОЗУ (*Window RAM*), синхронную графическую память (*SGRAM*) и др., а также схемы памяти с различными модификациями внутренних интерфейсов: память с виртуальными каналами (*Virtual Channel Memory*), EDRAM (улучшенная динамическая память), 3DRAM ("трехмерная"память) и др., позволяющими повысить быстродействие ЗУ.

К запоминающим устройствам, использующим иные технологические и/или физические принципы, относятся, например, ЗУ на приборах с зарядовой связью и ферроэлектрические ЗУ.

ЗУ на приборах с зарядовой связью (или с переносом зарядов) ПЗС относятся к классу ЗУ с циклическим доступом. Эти ЗУ построены на основе перемещения электрического заряда, для реализации которого используется МОП-транзистор, имеющий много затворов. Информация в таком транзисторе хранится в виде заряда под затвором. Подавая на соседние затворы сдвинутые по

фазе потенциалы, можно перемещать заряды в требуемом направлении от затвора к затвору. Разместив такие транзисторы и затворы на них соответствующим образом, можно построить логические аналоги сдвигающих регистров с циклическим сдвигом (или дорожек диска).

Матрицы светочувствительных ПЗС широко применяются в сканерах, цифровых фотоаппаратах и видеокамерах. В этих матрицах инжекция зарядов в проводящую область осуществляется под действием света.

Ферроэлектрические ЗУ подобны динамическим ЗУ, но вместо обычных конденсаторов в них используются конденсаторы на ферроэлектриках, обладающие свойством сохранения остаточной поляризации. Характеристика состояния таких элементов имеет вид гистерезисной петли, подобной той, которую имели элементы в ЗУ на ферритовых сердечниках. Этот тип памяти рассматривается как один из кандидатов на место оперативной памяти, позволяющей сохранять информацию при выключении питания.

К запоминающим устройствам с дополнительными функциональными возможностями относятся ассоциативные и многофункциональные ЗУ.

Доступ к информации в ассоциативных ЗУ (АЗУ) осуществляется по содержимому. Это позволяет выполнять в таких ЗУ быстрый поиск в несортированных и неиндексированных массивах данных.

В таких ЗУ часть матрицы элементов памяти используется как поисковая область, т. е., в каждом хранимом в АЗУ слове часть разрядов используется как поле ключа, по значению которого и производится доступ (чтение или запись нового значения) к остальным разрядам найденного слова. Элементы памяти этой части матрицы должны выполнять операцию сравнения хранимых в них значений со значением искомого ключа. Поиск в АЗУ может выполняться не только по критерию точного совпадения, но и по критериям “больше” и “меньше”, а также с маскированием части разрядов искомого ключа.

Хотя ассоциативные ЗУ позволяют эффективно решать задачи, связанные с поиском информации, их распространение ограничено высокой стоимостью.

Другим классом запоминающих устройств, обладающим б***о***льшими функциональными возможностями, являются многофункциональные запоминающие устройства (МФЗУ). Эти устройства позволяют выполнять не только поиск, но и логическую и арифметическую обработку данных в них самих. Это дает возможность выполнять операции без пересылки операндов в процессор, уменьшая трафик между процессором и памятью, а также производить операции сразу над массивами данных.

Для выполнения таких действий возможны несколько различных вариантов организации МФЗУ: на основе типовых накопителей ЗУ с изменением диаграмм их работы, с модификацией элементов памяти и/или системы шин накопителей и регистров данных ЗУ.

2.3.4. ЗУ с подвижным носителем

В запоминающих устройствах с подвижным носителем процесс доступа к данным включает в себя механическое перемещение среды, на которой хранит-

ся информация. В качестве такой среды обычно выступают специальные покрытия, наносимые на жесткие или гибкие поверхности и обладающие определенными магнитными, оптическими или магнитооптическими свойствами. Геометрическая форма несущей поверхности, как правило, круг с отверстием посередине, который называют диском. Эта форма позволяет поддерживать непрерывное механическое движение носителя, что, хотя и менее выгодно энергетически, но обеспечивает большее быстродействие, не требуя разгона и останова носителя, необходимых в других случаях. Существовали ЗУ с носителем в виде цилиндра, названного магнитным барабаном, и магнитной ленты. Такая лента используется в стримерах.

Из-за наличия механических перемещений такие запоминающие устройства не обеспечивают высокого быстродействия, особенно при поиске, хотя скорости передачи данных для них достаточно высоки.

Запоминающие устройства на жестких магнитных дисках

Накопители на жестких магнитных дисках (НЖМД), или в англоязычном варианте *hard disk drives* (HDD), являются одним из самых распространенных в настоящее время типов запоминающих устройств.

Запись информации на магнитных носителях (не только на жестких дисках) осуществляется за счет изменения состояния намагниченности отдельных участков их поверхности. Чем меньше геометрические размеры таких участков, тем большее количество информации удается записать на единице площади носителя, т. е. тем выше плотность записи информации.

Жесткие диски состоят из электромеханической и электронной частей.

Электромеханическая часть размещается в жестком корпусе, внутри которого закреплен шпиндельный двигатель с вращающимся шпинделем и смонтированными на нем дисками накопителя, а также установленный в этом же корпусе подвижный блок головок чтения/записи с приводом, обеспечивающим позиционирование (перемещение) головок.

Диски изготовлены из сплавов алюминия или специального стекла (иногда керамики). Последнее используется при высоких скоростях вращения шпинделя. Поверхность дисков имеет магнитное покрытие, на котором, собственно, и записывается информация.

Информация на диске располагается по окружностям, называемым *дорожками*, совокупность равноудаленных от центра дорожек поверхностей всех пластин НЖМД называют *цилиндром*. Дорожки разбиты на *секторы*, емкость которых в большинстве случаев 512 байтов. Сектор включает заголовок, поле данных и контрольный код этого поля.

Привод позиционирования головок чтения/записи чаще всего поворотный (для получения меньших его размеров) электромагнитный с подвижной катушкой, перемещающейся в поле постоянного магнита под действием протекающего по ней тока. Катушка механически связана с головками чтения/записи. После установки на требуемый цилиндр головки удерживаются на нем с помощью следящей системы, считывающей с диска специальные сервометки. Время по-

зиционирования на требуемую дорожку зависит от расстояния до нее от текущего положения головок чтения/записи.

В электронную часть диска входят контроллер, усилители сигналов интерфейсных шин и буферная память (кэш диска). Контроллер обеспечивает управление процессами разгона и останова шпинделя, позиционирования головок, чтения и записи информации, а также внешний интерфейс диска.

Подключение жестких дисков к ЭВМ осуществляется по специальным интерфейсам ATA, SCSI и SerialATA. Внешние диски могут подключаться по интерфейсам параллельного порта LPT, шинам USB и IEEE 1394.

Запоминающие устройства на оптических дисках

В отличие от жестких дисков, в состав которых входят и привод, и носитель, оптические диски съемные и являются самостоятельными компонентами, а привод представляет собой отдельное устройство.

Оптические диски изготавливаются из пластмассы, на поверхность которой наносится отражающий слой. Для записи информации используется различие интенсивности (или фазы) отраженного света, соответствующей нулю и единице данных, возникающее за счет углублений (ямок, *pits*), формируемых в отражающем слое, либо за счет изменения коэффициента отражения света от этого слоя. На большинстве оптических дисков информация размещается на одной спиральной дорожке, расположенной с одной стороны оптического диска (существуют и двусторонние диски) и начинающейся с его центральной части. Ширина ямок на дорожке 0,5 мкм, шаг спирали – 1,6 мкм, что дает более 22 тысяч витков спирали на диске диаметром 120 мм.

Информация считывается при освещении диска лазерным лучом с длиной волны порядка 0,6–0,8 мкм. Ровные поверхности (как ямки, так и их отсутствие) соответствуют нулевым битам, перепады высот (края ямок) – единичным битам (это соответствует методу записи “без возврата к нулю” на магнитных поверхностях). Минимальная длина ямки (две “1” подряд) составляет порядка 1 мкм. Можно подсчитать, что плотность записи информации при этом составит порядка 0,5 Гбит/кв. дюйм, т. е. примерно в 500 раз меньше, чем у жестких магнитных дисков.

Существующие приводы оптических дисков различаются по выполняемым действиям, типу дисков, исполнению, способу загрузки диска, виду интерфейса и др.

Типовой привод состоит из электромеханической, оптической и электронной частей. Электромеханическая часть включает в себя двигатель, вращающий шпиндель; систему позиционирования оптической головки (головок) чтения (и записи в записывающих приводах) и систему загрузки дисков. Оптическая часть включает в себя лазерный светодиод, систему фокусировки, фотоприемник и усилитель. Электронная часть представляет собой контроллер, обеспечивающий управление всеми процессами работы привода и интерфейс с шинами ЭВМ.

Накопители на гибких магнитных дисках

Гибкий диск (дискета) по размещению информации схож с жестким диском: у 3,5-дюймовой дискеты (диаметром около 85 мм) имеется по 80 концентрических дорожек с обеих сторон, на которых могут быть записаны по 9, 18 или 36 секторов размером по 512 байтов. Это дает емкость дискеты 720 Кбайт, 1,44 Мбайт и 2,88 Мбайт. Наиболее распространенным вариантом являются дискеты емкостью 1,44 Мбайт.

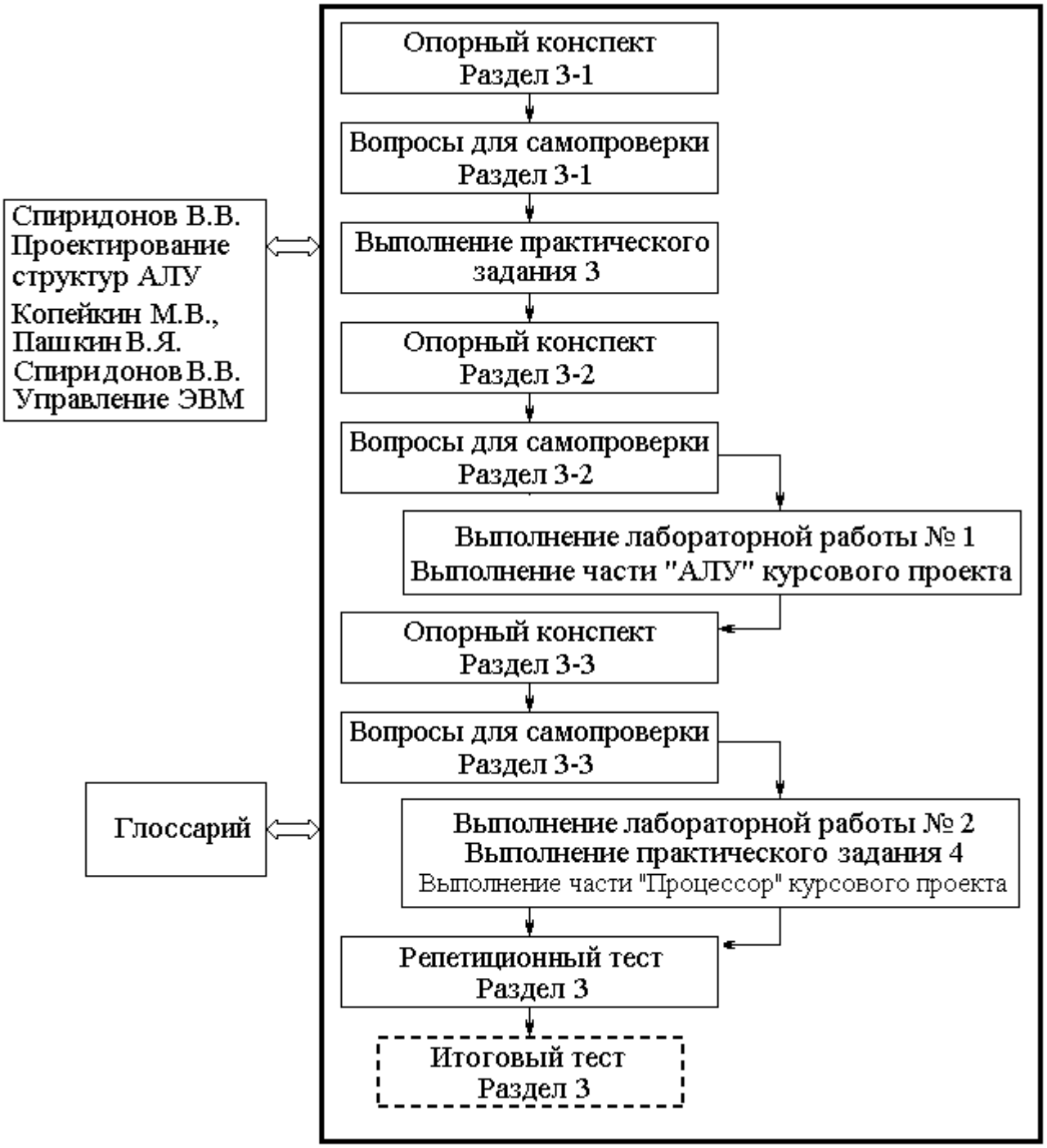
Привод накопителя на гибких магнитных дисках включает в себя электромеханическую часть с блоком головок чтения/записи и электронную часть. Электромеханическая часть состоит из шпиндельного двигателя, привода позиционирования головок чтения/записи и системы загрузки дискеты. Электронная часть НГМД содержит схемы управления двигателями, усилители сигналов для головок чтения/записи и дополнительные формирователи сигналов датчиков. Контроллер располагается в одной из микросхем чипсета и в электронику НГМД не входит.

Малая емкость накопителей на гибких магнитных дисках стимулировала разработки по созданию более емких устройств со сменными магнитными носителями. Однако они оказались вытесненными с приходом на рынок памяти новых технологий: флэш-памяти и перезаписываемых оптических дисков.

Вопросы для самопроверки по теме 2.3

1. В каких случаях используются постоянные ЗУ? Перепрограммируемые ЗУ?
2. Как записывается информация в постоянную память?
3. Для каких целей используется флэш-память?
4. Какие существуют разновидности флэш-памяти?
5. Что такое многофункциональные ЗУ?
6. Где применяются ЗУ с переносом зарядов?
7. Какие основные интерфейсы используются в жестких дисках?
8. Какие существуют разновидности оптических носителей?

Схема работы с разделом 3



Раздел 3. ПРОЦЕССОРЫ ЭВМ

Третий раздел курса включает три темы: “***Общие сведения о структуре процессоров ЭВМ”***, “***Арифметико-логические устройства процессоров”*** и “***Устройства управления ЭВМ”***. После изучения каждой темы Вам следует ответить на вопросы для самопроверки.

По материалам раздела также выполняется курсовой проект по дисциплине.

Работа с разделом 3 завершается сдачей контрольного теста. Кроме того, по данному разделу выполняются практические задания 3 и 4, а также лабораторные работы № 1 и № 2.

Для того, чтобы Вы смогли успешно ответить на вопросы контрольного теста, Вам предоставляется возможность поработать с репетиционным тестом. Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к главе 7 учебника [1] или к материалам электронного учебного пособия [5] (электронная редакция 2005 г.) и пособия [6].

3.1. Общие сведения о структуре процессоров ЭВМ

При изучении данной темы Вы должны познакомиться со структурой процессоров ЭВМ, назначением и составом основных их компонентов.

Для проверки изучения материала темы Вам предстоит ответить на вопросы для самопроверки. По данной теме также выполняется практическое задание 3, указания к выполнению которого даны в разделе методических указаний 3.4.

Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к главе 7 учебника [1].

3.1.1. Функциональная и структурная организация процессоров

Процессор представляет собой центральную часть ЭВМ, осуществляющую выполнение программ, обработку информации и координацию работы ЭВМ в целом. Состав и структура процессоров могут варьироваться в зависимости от функционального назначения процессора (ЭВМ), предъявляемых требований по производительности и заложенных архитектурных концепций. Но в любом случае процессор содержит в себе компоненты, реализующие все четыре основных класса функций: хранение, переработку, передачу и управление.

Функции *хранения* в процессоре реализуются регистрами и кэш-памятью (первого и второго уровней), входящими в состав процессора.

Регистры в свою очередь можно также разделить на регистры данных, адресов (сегментов и др.) и управляющих кодов. Непосредственно командам доступны в основном регистры данных, значения остальных устанавливаются самим процессором. Кэш-память, хотя и входит физически в состав процессора, все-таки относится к системе памяти ЭВМ.

Функции *обработки* реализуются операционными (исполнительными) блоками процессора, традиционно называемыми арифметико-логическими устройствами.

Функции *передачи* возлагаются на различные интерфейсные блоки процессора, обеспечивающие связь с памятью и остальными устройствами ЭВМ. Обычно эта связь осуществляется по различным шинам (например, системным шинам, шинам памяти), которыми управляют интерфейсные блоки.

Наконец, функции *управления* возлагаются на блоки управления, состав которых может в значительной степени варьироваться.

В общем виде структура процессора может быть представлена так, как показано на рис. 3.1, где *L1* и *L2* обозначают два уровня кэш-памяти.

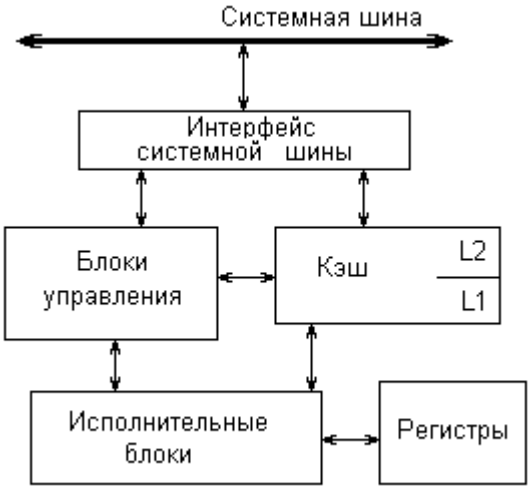


Рис. 3.1. Обобщенная структурная схема процессора

Особенности блоков процессоров определяются следующими факторами:
- функциональным назначением (универсальный процессор, сопроцессор, процессор для обработки сигналов);
- идеологией построения системы команд (обычная, RISC, VLIW);
- требованиями производительности, энергосбережения и условиями эксплуатации (требованиями надежности).

3.1.2. Базовые узлы процессоров

Базовые узлы процессоров и других устройств ЭВМ изучаются несколько позже в курсах “Электротехника и электроника” и “Схемотехника ЭВМ”. Однако для лучшего понимания структуры и порядка функционирования процессоров ЭВМ целесообразно рассмотреть основные моменты, связанные с представлением их на уровне базовых узлов и блоков ЭВМ.

В соответствии с разделением функций на четыре основных класса: преобразование, хранение, передача и управление – можно сгруппировать и базовые узлы ЭВМ.

Основными узлами, используемыми для преобразования информации в ЭВМ, являются сумматоры, счетчики, дешифраторы, шифраторы, сдвигатели, схемы сравнения, комбинационные схемы, реализующие различные логические функции.

Узлами, используемыми для хранения информации, служат триггеры, регистры и схемы накапливающего типа, сочетающие функцию хранения с преобразованием: сумматоры, счетчики, сдвигающие регистры.

Для передачи информации в ЭВМ служат простые соединения – шины, шины с разрешением передачи, коммутирующие схемы (мультиплексоры и демультиплексоры).

Управление реализуется различного рода управляющими автоматами и схемами, построенными на основе узлов преобразования, хранения и передачи, того же или более низкого уровня рассмотрения, чем сами управляющие блоки.

Все узлы в ЭВМ строятся на основе логических элементов, изучавшихся в курсах “Информатика” и “Электротехника и электроника”.

Методы синтеза узлов и блоков ЭВМ рассматриваются в дисциплинах “Теория автоматов” и “Схемотехника ЭВМ”.

Вопросы для самопроверки по теме 3.1

1. Какие классы базовых функций реализуются в процессорах ЭВМ?
2. Чем могут различаться исполнительные блоки процессоров?
3. Что такое суперскалярный процессор?
4. Какие существуют разновидности регистров процессора?
5. Назовите базовые узлы, реализующие функции преобразования, хранения, передачи.
6. Как описываются базовые узлы?
7. Что не могут выполнять узлы комбинационного типа?

3.2. Арифметико-логические устройства процессоров

При изучении данной темы Вы должны познакомиться со структурой арифметико-логических устройств процессоров ЭВМ, их назначением и составом, а также микропрограммами выполнения основных арифметических и логических операций в них.

По данной теме выполняется лабораторная работа № 1, а также раздел “Арифметико-логическое устройство” курсового проекта. Рекомендации по выполнению этих заданий приведены в методических указаниях [9] и [10].

Для проверки изучения материала темы Вам предстоит также ответить на вопросы для самопроверки.

Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к главе 5 учебника [1] или к материалам учебного пособия [5] (электронная редакция 2005 г.).

3.2.1. Назначение, принципы организации и классификация АЛУ

Все основные операции по преобразованию данных в ЭВМ производятся в операционных блоках, которые обычно называют арифметико-логическим устройством (АЛУ). Набор операций, выполняемых АЛУ универсальных ЭВМ, должен быть функционально полным, т. е. обеспечивать реализацию любого вычислительного алгоритма. Как правило, в АЛУ предусмотрена возможность выполнения четырех основных арифметических операций, нескольких логических операций, а также сдвигов.

Типовая структурная схема АЛУ показана на рис. 3.2, где *Р1, Р2, Р3, Р4* – регистры, *МП1, МП2* – мультиплексоры, *См* – комбинационный сумматор, *Сдв* – сдвигатель, *СхФормПр* – схема формирования признаков (флажков), *РП* – регистр признаков. В этой схеме основным узлом преобразования является сумматор, выполняющий операции суммирования и логические операции, но логические операции могут выполняться в специальных узлах. Преобразования осуществляются и в сдвигателе. Регистры *Р1...Р3* служат для хранения операндов и промежуточных результатов, регистр *Р4* – выходной, используется для промежуточного хранения результатов, снимаемых с выхода сумматора (сдвигателя). Мультиплексоры *МП1* и М*П2* обеспечивают коммутацию на входы

сумматора содержимого регистров *Р1, ... Р3,* а в некоторых случаях и инвертирование, т. е. получение обратных кодов их содержимого. *СхФормПр* формирует значения логических условий, отражающих получение на выходе сумматора нулевого числа, отрицательного числа, переполнения результата, переноса из старшего разряда, четности результата и др., а регистр признаков *РП* (или набор триггеров) сохраняет эти значения.

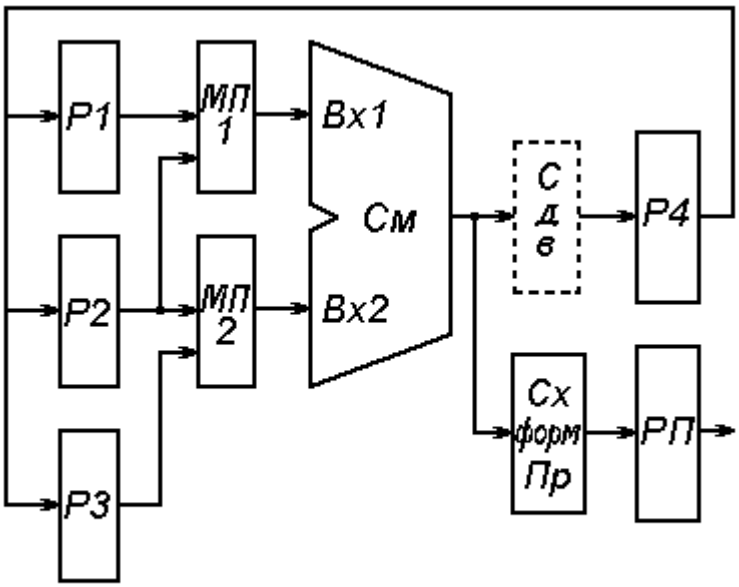


Рис. 3.2. Типовая структурная схема АЛУ

Рассмотренная структурная схема имеет обобщенный характер. На ней не показаны узлы управления, вспомогательные узлы.

Помимо набора операций и структурной организации, АЛУ характеризуются еще рядом показателей. К ним относятся разрядность обрабатываемых чисел (кодов), время выполнения различных операций, наличие дополнительных функциональных возможностей типа контроля правильности выполнения операций, устойчивости к отказам, а также конструктивные характеристики такие, как габариты, энергопотребление, надежность и пр.

Классификация АЛУ

В процессорах современных ЭВМ используются различные по своей организации АЛУ. Эти различия обусловлены функциональным назначением АЛУ, способами реализации операций, требованиями по быстродействию и др. Основные характерные особенности того или иного АЛУ можно отнести к одной из трех групп: особенности обрабатываемой информации, организации выполнения операций и структурной организации.

Обрабатываемая в АЛУ информация представляется по-разному. Это проявляется в основном в используемых формах представления данных, системах счисления, разрядности, применяемых кодах. По этим признакам АЛУ можно разделить следующим образом.

По *форме представления чисел*: АЛУ с фиксированной запятой; АЛУ с плавающей запятой; АЛУ с фиксированной и плавающей запятыми (универ-

сальные). Причем в рамках каждого представления имеются некоторые различия. Так, числа с фиксированной запятой могут быть представлены в виде целых или в виде дробных чисел, меньших единицы. Числа с плавающей запятой могут иметь мантиссу и порядок (целое со знаком) или мантиссу и характеристику (смещенный порядок).

Имеются и другие формы, например так называемая автоматическая запятая, используемая в калькуляторах.

По *используемой системе счисления*: АЛУ, работающие в позиционной системе счисления; АЛУ, работающие в непозиционной системе счисления.

Известно несколько позиционных систем счисления, используемых в ЭВМ. В первую очередь, это двоичная и двоично-десятичная системы счисления. Из непозиционных систем счисления в арифметике используется система остаточных классов (СОК), числа в которой представляются в виде остатков от деления исходного числа на набор взаимно простых чисел, называемых основаниями системы. Такое представление обеспечивает возможность независимой обработки разрядов (остатков) чисел.

По *разрядности* обрабатываемых чисел: АЛУ, выполняющие операции над числами (кодами) фиксированной разрядности; АЛУ, обрабатывающие операнды переменной длины.

По *кодам*, используемым для представления отрицательных чисел: АЛУ с использованием обратных кодов; АЛУ с использованием дополнительных кодов. Известны и устройства, в которых одни операции выполняются с использованием обратных кодов, а другие – дополнительных.

Особенности структурной организации АЛУ определяются составом операционных блоков устройства и характером связей между ними. В этой группе признаков АЛУ можно подразделить следующим образом.

По *количеству операционных блоков*: одноблочные АЛУ (иначе, универсальные или многофункциональные) и многоблочные АЛУ.

В первых из них имеется операционный блок, в котором может выполняться любая из операций АЛУ. Многоблочные АЛУ имеют в своем составе несколько операционных блоков, каждый из которых ориентирован на выполнение какой-либо одной операции, например умножения, или нескольких операций, например сложения и логики. Одновременная работа различных блоков обеспечивает более высокую производительность ЭВМ с такими АЛУ.

По *характеру связей*: устройства с магистральными и с непосредственными связями.

Для первых из них характерно наличие внутренней шины данных, по которой осуществляются все передачи информации между узлами АЛУ. В случае непосредственных связей имеется набор индивидуальных шин, связывающих пары узлов, между которыми должны выполняться передачи.

Структурные особенности могут определяться также и назначением ЭВМ, в состав которых входит АЛУ, в целом.

Особенности организации выполнения операций (процесса обработки) проявляются в принципах получения результатов и порядке обработки данных. По этим признакам возможны следующие подразделения.

По *принципу получения результата*: АЛУ с алгоритмической реализацией операций; табличные АЛУ; таблично-алгоритмические АЛУ.

В АЛУ с алгоритмической реализацией операций каждая операция (кроме самых простых) представляется в виде последовательности более простых преобразований – микроопераций. Последовательность этих преобразований определяется алгоритмом выполнения операций.

В табличных АЛУ результат операции не вычисляется каждый раз при ее выполнении. Он выбирается из таблицы – постоянной памяти, в которой заранее записаны назначения результатов, соответствующие всем возможным значениям операндов. Такой способ наиболее эффективен для вычисления сложных функций одного аргумента при небольшой его разрядности, например тригонометрических функций. Применим он и для реализации обычных арифметических операций.

При выполнении операций в табличных АЛУ значение аргумента (аргументов) используют в качестве адреса ячейки ПЗУ, в которой записан результат, соответствующий этому значению (значениям). Табличный способ обеспечивает высокую скорость обработки, так как независимо от сложности реализуемых преобразований все действия сводятся к считыванию готового результата из ПЗУ. Однако недостатком его является необходимость очень большого объема памяти (таблицы) при увеличении разрядности операндов.

Таблично-алгоритмические АЛУ представляют собой компромисс между первыми двумя способами. Часть разрядов операндов (обычно старшие разряды) используется для получения приближенного значения результата табличным способом. По остальным разрядам вычисляется поправка к предварительному результату. Это позволяет сократить объем таблиц при сохранении относительно высокой скорости.

По *порядку обработки данных*: последовательные АЛУ, параллельные АЛУ и конвейерные АЛУ.

Эти АЛУ различаются между собой по степени параллелизма в выполнении операций. Так, в АЛУ последовательного типа обработка операндов осуществляется последовательно разряд за разрядом. В АЛУ параллельного типа операции производятся одновременно над всеми разрядами операндов.

Известны также промежуточные варианты организации АЛУ – параллельно-последовательные, в которых обработка операндов осуществляется одновременно по группам разрядов, тогда как группы обрабатываются между собой последовательно.

В АЛУ конвейерного типа параллелизм имеет место на уровне операций, допуская выполнение нескольких операций одновременно. В ряде случаев термин «конвейерные АЛУ» применяют к многоблочным АЛУ. Например, АЛУ, имеющее в качестве отдельных блоков сумматор, устройство умножения и устройство деления, может обеспечивать конвейерную обработку. Но и сами устройства умножения и деления могут быть конвейерного типа и реализовы-

вать сразу несколько операций умножения или деления, которые в один и тот же момент времени пребывают в разных стадиях своего выполнения. Эти варианты конвейеров называют конвейерами последовательного типа в отличие от векторных конвейеров, выполняющих операции над векторами.

Возможны также подразделения АЛУ и по некоторым другим признакам.

3.2.2. Средства представления АЛУ. Формирование и преобразование структур операционных устройств

Необходимость в представлении АЛУ возникает в различных случаях: при описании устройств в литературе, при их проектировании, при моделировании с целью анализа характеристик и проверки правильности работы. Поскольку ЭВМ в целом и АЛУ можно отнести к классу многоуровневых систем, их представление может осуществляться на различных уровнях детализации и ориентировано на описание структуры устройства, его функционирования и его технической реализации.

Один из возможных вариантов, включающий шесть уровней представления структуры, соответствующие им процессы функционирования и техническую реализацию, а также средства представления, приведен в табл. 3.1.

Таблица 3.1. **Уровни и средства представления вычислительных устройств**

Уровень представления структуры	Процессы функционирования	Техническая реализация	Средства представления: структуры	функционирования	реализации
1.Электронные схемы	Токи в электрических цепях	Радиокомпоненты	Электрическая схема (принципиальная)	Диф. уравнения для токов и напряжений	Фотошаблоны масок интегральных схем (языка графического типа)
2. Логические схемы	Логические преобразования, переключение состояний элементов памяти	Интегральные схемы	Схема из логических элементов (электрическая принципиальная)	Булевы функции, конечные автоматы	Топология интегральных схем (языки графического типа)
3. Узлы и блоки	Выполнение микроопераций	Интегральные схемы, конструктивы (плата, ТЭЗ), их фрагменты	Схемы из логических элементов и узлов (электрическая функциональная)	Языки регистровых передач (микроопераций)	Планы кристалла (языки графического типа)
4. Устройства	Выполнение операций (микропрограмм)	Интегральные схемы , платы, ТЭЗы, панели, стойки	Схемы из узлов и блоков (электрические структурные)	Языки регистровых передач, языки описания микропрограмм	Укрупненный план кристалла, чертежи конструктива
5. ЭВМ	Выполнение команд (программ)	Конструктивы всех уровней	Структурные схемы	Языки команд, языки программирования	Чертежи конструктивов
6. Системы	Взаимодействие устройств, вычислительный процесс	Конструктивы всех уровней	Структурные схемы Для всех уровней: языки структурного описания	Языки моделирования систем, сетевые модели	Чертежи конструктивов

Реализация преобразований в системе переработки информации осуществляется на основе взаимодействия элементов (узлов) ее структуры. При решении задач разработки структуры отдельных устройств ЭВМ, в частности

арифметико-логического устройства, их рассмотрение осуществляется, как правило, на уровне регистровых передач и логических элементов.

При необходимости построить структуру, реализующую некоторое преобразование *F*, следует представить ее в виде связной совокупности известных структур (элементов). Для этого само преобразование *F* должно быть разбито (декомпозировано) на связную совокупность таких преобразований, выполнение которых возможно осуществить на известных структурах (элементах). На уровне рассмотрения структур устройств ЭВМ к таким преобразованиям могут быть отнесены преобразования, перечисленные в табл. 3.2. В этой таблице каждому функциональному преобразованию поставлен в соответствие структурный элемент, реализующий это преобразование, а также указан его тип: к – комбинационный, н – накапливающий.

К функциям Т, М и С (передачи (Т), хранения (М), управления (С)) типов на уровне регистровых передач можно отнести действия, представленные в табл. 3.3. В графе примечаний этой таблицы отражен тот факт, что разделение процессов и функций на Р, С, М и Т типы относительно, не абсолютно, как и любое разделение. Каждый тип процесса в той или иной мере сочетается с другими или даже может быть заменен ими. Например, преобразования невозможны без подачи информации на входы узла преобразования и съема результата с его выхода, условие для управления условным переходом часто приходится вычислять и т.д. Такие взаимосвязи, в частности, обусловливают, наряду с иными причинами, широкое разнообразие структур ЭВМ и неоднозначность проектных решений.

Таблица 3.2. **Основные функции Р-типа уровня регистровых передач**

Преобразование	Структурный элемент	Тип элемента
Суммирование двух кодов/битов	Сумматор	к/н
Добавление единицы (счет) Вычитание единицы	Счетчик Счетчик	к/н к/н
Сдвиг кода/бита	Сдвигающий регистр Сдвигатель	Н К
Сравнение кодов/битов (равенство, больше, меньше)	Схема сравнения Сумматор	К к/н
Сравнение с константой	Схема анализа содержимого	К
Логические функции: И, ИЛИ, исключающие ИЛИ, НЕ	Схема логических операций (сумматор)	к/н
Произвольная логическая функция	Комбинационная схема, ПЛМ	К
Шифрация (преобразование кода)	Шифратор	К
Дешифрация	Дешифратор	К

Этапы формирования структур АЛУ

Одной из наиболее полных методологий является эволюционный синтез систем, базирующийся на предложенном его автором [*Балашов Е.П. Эволюционный синтез систем. – М.: Радио и связь, 1985. - 328 с.*] функционально-структурном подходе к анализу и синтезу систем. Этот подход представляет

собой совокупность концепций, объективных закономерностей развития, положений и выводов, определяющих стратегию анализа и синтеза, и рассматривает процесс построения систем обработки информации как процесс перехода от функций разрабатываемой системы к ее структуре.

Таблица 3.3. **Основные функции Т-, М- и С- типа уровня регистровых передач**

Функция	Функциональный элемент, блок	Примечание
Т-тип		
Подача кода /бита из источника (узла, шины) на вход узла	Жесткая связь (выхода узла с источником) Связь (входа узла с источником) через блок вентилей, мультиплексор	Т+С
Подача фиксированного кода/бита на вход узла	Жесткая связь информационного входа узла с 0/1 Шина управления информационным/синхровходом узла	Т=С
Выдача кода/бита с выхода узла (в приемник: узел, шину)	Жесткая связь выхода узла с приемником Связь выхода узла с приемником через блок вентилей, мультиплексор	Т+С
М-тип		
Хранение кода/бита	Триггер, регистр, блок регистров, ЗУ	
Фиксация (запоминание) кода/бита	Триггер, регистр, блок регистров, ЗУ	М+С
Хранение фиксированного значения кода/бита	Управляющая шина	
С-тип		
Разрешение выполнения передачи, преобразования (синхронизация)	Управляющая шина (вход)	
Задание выполняемой функции	Управляющая шина Управляющий (кодовый вход)	С+Р (Дш)
Безусловный переход к следующему действию	Управляющий автомат (генератор синхроимпульсов)	
Условный переход к следующему действию	Управляющий автомат	С+Р

Формирование структуры системы осуществляется на основе декомпозиции и композиции функций и структур различных подсистем, а преобразование моделей системы производится в такой последовательности:

1) формирование дерева функций системы;

2) декомпозиция функций системы до уровня принятого базового набора операторов (преобразования 1 и 2 выполняются итеративно);

3) формирование функциональной структуры системы;

4) формирование технической структуры системы.

В этой последовательности можно выделить этапы анализа систем прототипов, исследования дерева противоречий системы, формирования концепции системы, дерева функций системы и др.

3.2.3. АЛУ для выполнения основных арифметических операций

Устройство для сложения и вычитания двоичных чисел

Пример структуры устройства для сложения чисел с фиксированной запятой, представленных в прямом коде, показан на рис. 3.3, где *Р1, Р2* и *Р3* – (*n*+1)-разрядные регистры, *n* разрядов у которых цифровые, а разряд 0 – знаковый; *ТПП* – триггер, хранящий значение признака переполнения; *МП1* и *МП2* – мультиплексоры, осуществляющие передачу прямых или инверсных значений цифровых разрядов в зависимости от поданного управляющего сигнала; A_1, ..., A_7 – управляющие сигналы, содержание которых следующее:

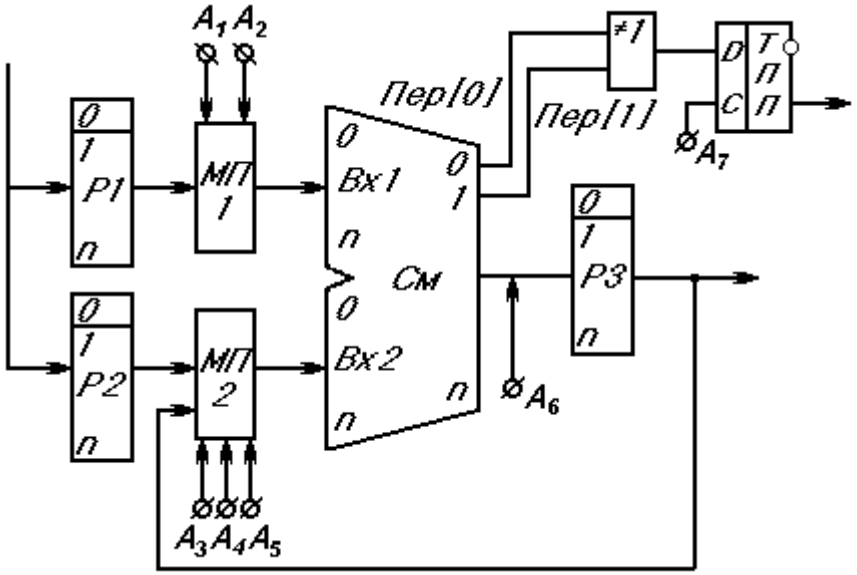


Рис 3.3. Структура АЛУ для сложения двоичных чисел с фиксированной запятой

A_1: *Вх1* := (*Р1[0 : n]*) – подача на вход 1 сумматора содержимого всех разрядов регистра *Р1*,

A_2: *Вх1* := (*Р1[0]*).($\overline{Р1[1:n]}$) – подача на вход 1 сумматора содержимого знакового разряда регистра *1* и инверсии содержимого всех цифровых разрядов регистра *Р1,* т. е., обратного кода числа, записанного в регистре *Р1,*

A_3: *Вх2* := (*Р2[0 : n]*) – подача на вход 2 сумматора содержимого всех разрядов регистра *Р2,*

A_4: *Вх2* := (*Р2[0]*).($\overline{Р2[1:n]}$)– аналогично A_2,

A_5: *Вх2* := (*Р3[0]*).($\overline{Р3[1:n]}$) – аналогично A_2,

A_6: (*Р3*) := *ВыхСм* – занесение в регистр *Р3* информации с выхода сумматора,

A_7: (*ТПП*)*:= Пер[0]*⊕*Пер[1]* – занесение в триггер признака переполнения суммы по модулю два значений сигналов переносов из нулевого и первого разрядов сумматора.

Во всех случаях в квадратных скобках указываются номера разрядов соответствующего регистра. Круглые скобки, означающие содержимое соответствующего узла, иногда для сокращения записи могут опускаться, например, можно записать A_2 как $Вх1 := Р1[0].\overline{Р1[1:n]}$.

В таком устройстве выполнение операции сложения осуществляется в зависимости от знака результата за один или два такта. В первом такте слагаемые подаются на входы сумматора, причем передача цифр осуществляется прямо или инверсно, что определяется знаками слагаемых. С выхода сумматора в этом же такте результат заносится в регистр Р3, а в триггер *ТПП* заносится значение признака переполнения при подаче на его С-вход разрешающего сигнала A_7. Второй такт необходим в случае получения отрицательной суммы для преобразования ее в прямой код инверсной передачей цифровых разрядов содержимого регистра *Р3* через *МП2*. Нужно учитывать, что для выполнения этого действия триггеры регистра *Р3* должны быть двухтактными (построенными по *М–S* схеме). В противном случае потребуется дополнительный такт – передачи суммы в регистр *Р2* перед преобразованием ее в прямой код.

Микропрограмма выполнения этой операции представлена на рис. 3.4. Предполагается, что операнды (слагаемые) к началу операции находятся в регистрах *Р1* и *Р2,* а результат должен размещаться по окончании операции в регистре *Р3*. Управляющий сигнал A_7, фиксирующий значение признака переполнения, можно подавать только при одинаковых знаках слагаемых, так как в противном случае переполнение невозможно.

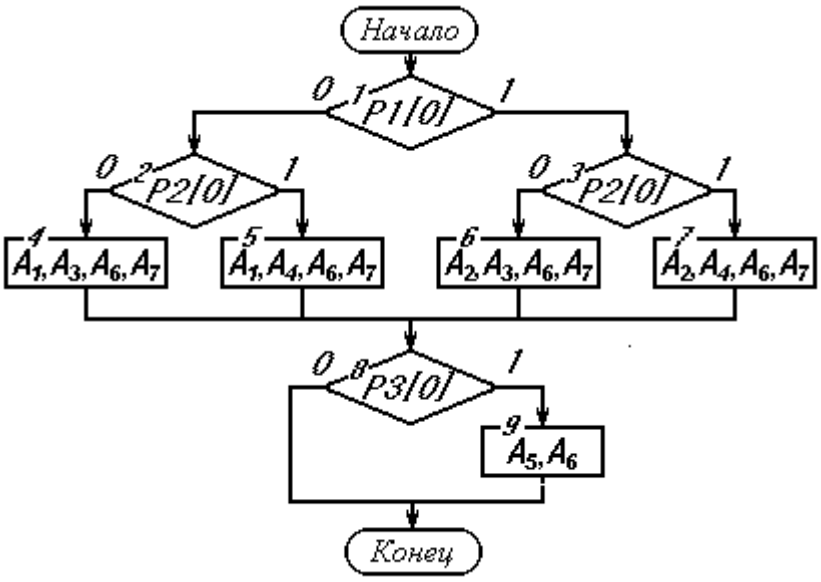


Рис 3.4. Микропрограмма для сложения двоичных чисел с фиксированной запятой

Выполнение операции вычитания сводится обычно к сложению уменьшаемого с вычитаемым, взятым с обратным знаком.

Рассмотренное устройство построено на основе комбинационного сумматора. Если использовать сумматор накапливающего типа, то структура АЛУ будет несколько иной.

Устройство для умножения двоичных чисел

Общий ход операции умножения в ЭВМ подобен умножению чисел вручную, но имеет некоторые отличия. Частичные произведения, получаемые при умножении на разряды множителя, не хранятся до конца операции, где они суммируются на последнем этапе ее выполнения, а каждое получаемое частич-

ное произведение сразу же добавляется к сумме частичных произведений, накопленной при умножении на предшествующие разряды множителя.

Возможны четыре варианта общей схемы умножения, различающихся порядком анализа разрядов множителя (начиная со старших или с младших разрядов) и сдвигаемым компонентом: множимым или суммой частичных произведений. Наиболее распространен вариант умножения с анализом множителя, начиная с младших разрядов, и со сдвигом суммы частичных произведений вправо при неподвижном множимом, схема которого показана на рис. 3.5 для множимого и множителя, имеющих по *n* цифровых разрядов.

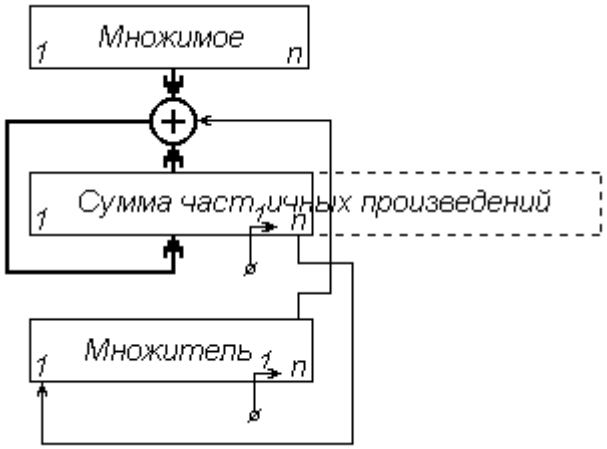


Рис. 3.5. Вариант структуры для умножения чисел с фиксированной запятой

При такой схеме умножения выход младшего разряда регистра множителя используется для управления суммированием множителя с ранее накопленной суммой частичных произведений: единичное значение на этом выходе говорит о необходимости суммирования. Регистры, в которых хранятся множитель и сумма частичных произведений, имеют цепи сдвига вправо на один разряд. Регистр суммы частичных произведений, которая в общем случае имеет двойную длину (*2n* разрядов), не обязательно должен иметь такую же разрядность. Поскольку множитель в процессе умножения выдвигается из регистра, то в освобождающиеся его разряды можно заносить младшие разряды суммы частичных произведений, что позволяет сделать регистр суммы такой же длины, как и регистры множимого и множителя.

Помимо собственно формирования произведения как суммы частичных произведений, необходимо также сформировать его знак и, возможно, выполнить округление.

Знак произведения определяется как сумма по модулю 2 (*mod2*) знаков сомножителей. Сомножители в случае представления их в прямом коде перемножаются без знаков.

В случае представления чисел (сомножителей и произведения) в дополнительном коде знаковый разряд множимого непосредственно участвует в умножении, а знаковый разряд множителя косвенно влияет на последний цикл умножения.

Устройство для деления двоичных чисел

Известны два основных способа деления чисел в ЭВМ: деление с восстановлением остатка и деление без восстановления остатка. При делении двоичных чисел чаще используется первый способ, как более быстрый.

Как и в случае умножения, возможны различные варианты выполнения операции, но более распространено деление с неподвижным делителем и сдвигом частичного остатка (остатка получаемого при определении очередной цифры частного) влево. Это позволяет сократить разрядность используемых в АЛУ регистров.

Такой способ выполнения деления можно представить в виде схемы, показанной на рис. 3.6.

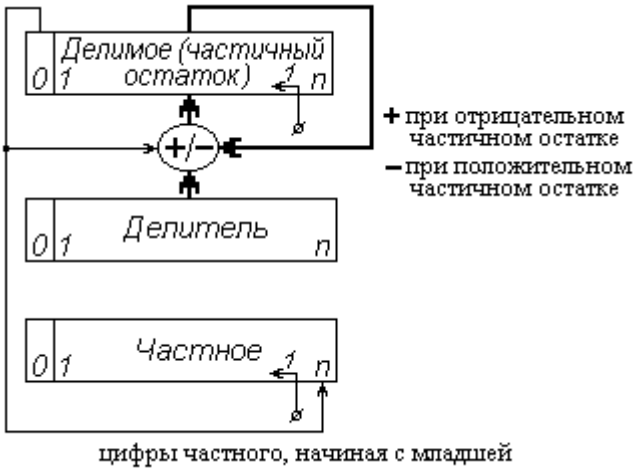


Рис. 3.6. Вариант структуры устройства для деления чисел с фиксированной запятой

В этой схеме цифры частного определяются по одной, начиная со старшего разряда, при каждом вычитании (или добавлении) делимого из делителя или частичного остатка от предыдущего шага.

Помимо собственно определения цифр частного, необходимо также определить знак результата и выполнить его округление. Знак результата, как и в умножении, определяется суммированием по модулю 2 знаков делимого и делителя. Для выполнения округления определяется дополнительная цифра частного, при единичном значении которой и производится добавление единицы в младший из сохраняемых разрядов частного.

Кроме того, если делятся дробные числа с фиксированной запятой, меньшие единицы, то в случае, когда модуль делимого больше модуля делителя, модуль частного должен быть больше единицы. Это не может быть представлено в разрядной сетке дробных чисел, поэтому следует зафиксировать переполнение.

Деление целых чисел выполняется аналогично. Но в этом случае переполнение разрядной сетки произойти не может, а пробное вычитание выполняется для определения количества разрядов целой части частного, предварительно принимаемого равным количеству сдвигов делителя до получения отрицательного остатка при пробном вычитании.

3.2.4. Выполнение десятичных и логических операций в АЛУ

Наиболее распространенным вариантом двоичного кодирования десятичных чисел в ЭВМ является представление каждой десятичной цифры X_i десятичного числа X четверкой двоичных разрядов, называемой тетрадой. Веса двоичных разрядов тетрады соответствуют весам обычного двоичного числа (8-4-2-1), а коды десятичных цифр при этом соответствуют их обычному двоичному представлению: 0 – 0000, 1 – 0001, 2 – 0010, . . . , 9 – 1001.

Такая форма представления, называемая двоично-десятичной, удобна как для обработки чисел в ЭВМ, так и для их восприятия человеком. Однако в каждой тетраде может быть представлено 16 кодовых комбинаций (фактически соответствующих шестнадцатеричным цифрам), шесть из которых не являются десятичными цифрами: 1010, 1011, . . ., 1111. Поэтому сложение и вычитание двоично-десятичных чисел несколько отличается от сложения и вычитания обычных двоичных чисел.

Одним из наиболее распространенных вариантов выполнения этих операций является добавление кода “6” при сложении (вычитании) двоично-десятичных чисел, позволяющее легко идентифицировать те случаи, когда результат в тетраде попадает в диапазон недействительных для десятичных цифр комбинаций 1010 – 1111.

Структура АЛУ для сложения двоично-десятичных чисел подобна АЛУ для сложения двоичных чисел с фиксированной запятой, но включает в свой состав дополнительный узел фиксации межтетрадных (десятичных) переносов и схему десятичной коррекции, формирующую код коррекции в соответствии с сигналами десятичных переносов.

Выполнение логических операций в АЛУ

Логические операции являются наименее сложными преобразованиями, выполняемыми в АЛУ. В реальности АЛУ выполняют не только логические операции, причем для реализации последних, как правило, не требуется дополнительных узлов к тем, которые уже имеются в АЛУ. Однако в ряде случаев они оснащаются специальными логическими схемами, выполняющими только такие операции, если по каким-либо соображениям выполнение их в сумматоре не оказывается целесообразным. В этих случаях добавляются и соответствующие связи входных и выходных регистров с этими схемами.

Вопросы для самопроверки по теме 3.2

1. Для каких целей могут служить описания АЛУ?
2. Какие основные узлы входят в состав АЛУ?
3. Для чего может выполняться преобразование структур АЛУ?
4. Каковы основные этапы формирования структуры АЛУ?
5. Чем различается сложение с использованием обратных и дополнительных кодов?
6. Какие существуют основные варианты структур АЛУ для умножения?
7. Как можно ускорить выполнение операции умножения?
8. Чем различаются сложение с фиксированной и плавающей запятой?

9. Как и в каких случаях выполняется округление результата при делении чисел с фиксированной запятой?

10. В чем заключаются особенности десятичного сложения?

3.3. Устройства управления ЭВМ

При изучении данной темы Вы должны познакомиться с организацией процесса выполнения команд в процессорах ЭВМ и особенностями построения устройств их управления.

По данной теме выполняется лабораторная работа №3, практическое задание 4, а также раздел “Процессор” курсового проекта. Рекомендации по выполнению этих заданий приведены в методических указаниях [9] и [10], а также в методических указаниях к практическим занятиям в разделе 3.4.

Для проверки изучения материала темы Вам предстоит также ответить на вопросы для самопроверки.

Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к главам 2 и 3 учебника [1] или к материалам учебного пособия [6] (электронная редакция 2007 г.).

3.3.1. Устройства управления ЭВМ. Назначение, функции, классификация

Устройства управления (УУ) управляют конечным числом блоков ЭВМ; вырабатывают конечные последовательности (вследствие свойства результативности алгоритмов) управляющих сигналов; имеют конечное число внутренних состояний, входных и выходных сигналов. Поэтому для описания и проектирования УУ широко применяют теорию конечных автоматов.

Одной из разновидностей конечных автоматов являются программные автоматы, к которым и относятся цифровые вычислительные машины. В основу их работы положен принцип программного управления, предполагающий разделение всей информации, с которой оперирует автомат, на две части: собственно информацию (входные данные) и информацию об алгоритме (программу). Эти части представляются в виде информационных слов (ИС) и управляющих слов (УС), хранимых в памяти ЭВМ, что влечет за собой в большинстве современных ЭВМ операционно-адресную организацию процесса выполнения алгоритма, при которой управляющие слова задают операции алгоритма и местоположение в памяти операционных слов (операндов), участвующих в операции. Алгоритм решения задачи на ЭВМ описывается последовательностью УС (программой), определяющей очередность и характер выполняемых действий.

Управляющие слова в зависимости от сложности и времени выполнения задаваемых ими преобразований информации разделяются на команды и микрокоманды.

Командой называют записанную в некотором алфавите совокупность всех сведений, необходимых для выполнения некоторой операции в машине. Под структурой команды понимают перечень сведений (элементов команды), необходимых для выполнения операций.

Форматом команды называют распределение отдельных элементов команды по символам (цифрам) слова, изображающего команду.

Большинство команд ЭВМ включает операционную и адресную части. Структура и формат операционной части команды определяются перечнем отличающихся друг от друга операций, которые могут выполняться конкретной ЭВМ, и построением УУ. Структура и формат адресной части команд зависит от количества указываемых в команде адресов, объема адресуемой памяти и способов адресации, реализуемых УУ.

Совокупность всех выполняемых машиной команд, отличающихся друг от друга операционной частью и/или форматом адресной части, называется набором команд, или системой команд, ЭВМ. Универсальные ЭВМ должны обладать универсальной системой команд.

Понятие универсальности трактуется для ЭВМ как возможность выполнения ей заранее заданного алгоритма. Это означает, что универсальная ЭВМ может решить любую задачу, для которой имеется или может быть составлен алгоритм ее решения. В то же время известно, что существуют алгоритмически неразрешимые задачи.

Выполнение команды сводится к выполнению более мелких операций – микроопераций, задаваемых последовательностью микрокоманд.

Набор микроопераций, реализуемых операционной частью ЭВМ, должен позволять составить микропрограмму выполнения любой команды, входящей в систему команд данной ЭВМ, т. е. должен обладать *полнотой* по отношению к заданной системе команд.

С учетом названных особенностей организации вычислительного процесса ЭВМ с традиционной структурой устройство управления ЭВМ должно реализовывать следующие функции:

1. Формирование адреса команды, подлежащей выполнению;
2. Выборку очередной команды из памяти и хранение команды или ее части во время выполнения задаваемых командой операций;
3. Определение типа команды и/или операции и формирование соответствующего цикла выполнения команды (последовательности тактов или циклов процессора, необходимых для выполнения команды);
4. Формирование адресов операндов;
5. Извлечение операндов из памяти и отсылку их в арифметико-логическое устройство (АЛУ);
6. Запуск АЛУ на выполнение операции (при наличии отдельного блока управления АЛУ) или формирование управляющих сигналов, для управления выполнением операций в АЛУ (при отсутствии отдельного блока управления АЛУ);
7. Формирование адреса результата и запись его в память.

Первые три функции относят к управлению выполнением последовательности команд, последние четыре – к управлению выполнением операций. Для определенных типов команд реализация последних функций может не требоваться.

Классификация устройств управления

В зависимости от особенностей ЭВМ устройства управления имеют различную структуру. Основными признаками, по которым разделяются УУ, являются функциональная ориентация, принцип организации цикла выполнения команды, количество уровней управления и другие. Каждый признак позволяет выделить различные типы устройств управления:

1. *Функциональная ориентация* – универсальные УУ; специализированные УУ.

Универсальные УУ с учетом ограничений по быстродействию позволяют выполнять на ЭВМ программы решения любых задач, для которых имеется алгоритм решения.

Специализированные УУ используются в ЭВМ, предназначенных для решения определенного набора задач (одной задачи), программы (микропрограммы) для которых часто составлены заранее.

2. *Принцип организации цикла выполнения команды* – синхронные УУ; асинхронные УУ; смешанные (синхронно-асинхронные) УУ.

В синхронных УУ время выполнения каждой команды постоянно. При этом УУ вырабатывает последовательность тактирующих сигналов, обеспечивающую выполнение любой команды. В асинхронных автоматах управления очередная команда начинает выполняться сразу же по окончании выполнения предыдущей.

В большинстве УУ современных ЭВМ используются различные сочетания синхронного и асинхронного принципов.

3. *Порядок следования команд программы* – УУ с естественным порядком следования команд; УУ с принудительным порядком следования команд.

В первом случае команды располагаются в памяти в порядке их записи в тексте программы, а во втором – в произвольном порядке.

4. *Формат реализуемых команд* – одноадресные, двухадресные, трехадресные УУ; УУ с переменной адресностью команд; безадресные УУ.

Наличие одного, двух или трех адресов в команде позволяет задать местоположение в оперативной памяти или на регистрах общего назначения одного операнда, двух операндов (операнда и результата) и двух операндов и результата соответственно. Большее количество адресов в команде обеспечивает удобство и эффективность программирования, но приводит к большей разрядности команды, увеличивая расход памяти для их хранения.

В УУ с переменной адресностью команд используются сочетания названных выше форматов.

5. *Способ построения управляющего автомата* – схемные УУ; микропрограммные УУ.

Схемные УУ строятся на основе распределителей импульсов или микропрограммных автоматов Мили и Мура. В микропрограммных УУ имеется специальный блок постоянной памяти, в котором записаны

микропрограммы всех выполняемых ЭВМ команд и других операций. Такие УУ иногда называют устройствами управления с программируемой логикой.

6. *Способ реализации команд* – централизованные УУ; распределенные УУ.

При централизованном управлении все управляющие сигналы, необходимые для выполнения любой команды, вырабатываются одним управляющим автоматом, что характерно для ЭВМ с микропрограммным управлением. При распределенном управлении в отдельных устройствах ЭВМ, например в АЛУ, имеются собственные управляющие сигналы, необходимые для работы этих устройств.

Возможны и другие классификационные признаки УУ, связанные с особенностями операционной части процессора и другими чертами ЭВМ.

3.3.2. Управление выполнением команд и операций

Управление выполнением последовательности команд

Функции УУ, относящиеся к управлению выполнением последовательности команд, должны обеспечивать автоматический переход к очередной команде программы по окончании выполнения текущей команды. Для этого устройству управления необходимо, прежде всего, определять местоположение в памяти (адрес) очередной команды.

Существуют два основных варианта определения варианта адреса очередной команды. В первом из них в выполняемой команде в явном виде указывается адрес очередной команды. Такой способ приводит к принудительному порядку следования команд (ППСК) программы, а располагающиеся в программе друг за другом команды могут размещаться в ячейках памяти с произвольными адресами. Во втором случае адрес очередной команды явно не указывается, а получается из адреса выполняемой команды посредством добавления к нему величины, равной числу слов (байтов), занимаемых выполняемой командой. При таком способе говорят о естественном порядке следования команд (ЕПСК) программы, а сами команды размещаются в ячейках памяти с последовательными адресами.

ППСК приводит к некоторому сокращению общего числа команд в программе за счет исключения команд переходов. Однако сами команды в этом случае имеют увеличенную разрядность из-за необходимости включения в них поля адреса очередной команды.

Естественный порядок следования команд не требует указания в команде адреса очередной команды, однако разветвления и циклы в программах приводят к нарушению ЕПСК. Поэтому в современных ЭВМ в качестве основного режима используется естественный порядок следования команд, который в необходимых случаях сочетается с принудительным порядком следования команд. Это обусловливает наличие в системе команд любой ЭВМ, по меньшей мере двух типов команд: операционных команд, задающих преобразования информации, и команд управления, вызывающих переход к

некоторому участку программы. Выполнение команд первого типа не нарушает ЕПСК, команды второго типа являются средством реализации ППСК.

Рассмотрим управление выполнением последовательности команд на примере УУ, оперирующего с одноадресными командами. В адресной части в этом случае указывается: в операционных командах – адрес операнда, в командах управления – адрес очередной команды программы.

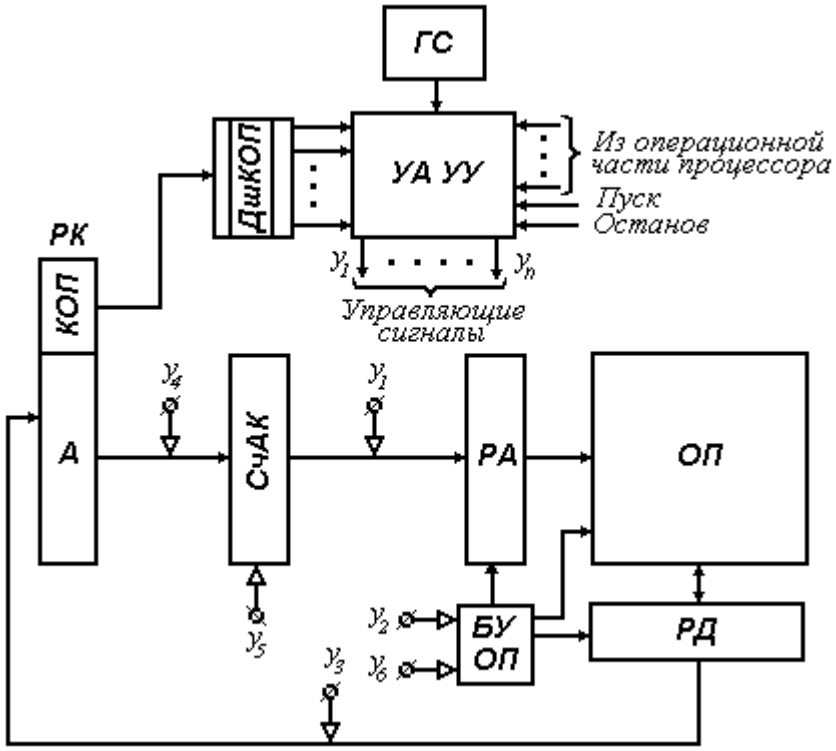


Рис. 3.7. Структурная схема устройства управления ЭВМ с одноадресной системой команд

Структурная схема УУ представлена на рис. 3.7. На схеме показаны только те узлы УУ, которые участвуют в реализации функции управления выполнением последовательности команд, и оперативная память (*ОП*), где хранятся команды выполняемой программы. На рисунке выделены следующие узлы: *РК* – регистр команд, *СчАК* – счетчик адреса команд, *ДшКОП* – дешифратор кода операций, *УА УУ* – управляющий автомат УУ, *ГС* – генератор импульсов синхронизации (синхроимпульсов), *РА* – регистр адреса *ОП*, *РД* – регистр данных, *ОП* и *БУ ОП* – блок местного управления оперативной памятью. Последние три блока относятся к оперативной памяти, а не к устройству управления, причем *БУ ОП* в некоторых типах *ОП* не выделяется в самостоятельный блок.

В регистре команд *РК* выделены два поля: *КОП* – поле кода операции и адресное поле *А*, в которых размещаются собственно операционная и адресная части команды.

Символами y_1 ...y_n обозначены управляющие сигналы, вырабатываемые *УА УУ*, причем подача сигналов y_1... y_6 в соответствующие управляющие шины вызывает выполнение следующих микроопераций [*)]:

y_1: (*РА*):=(*СчАК*) – передача содержимого счетчика адреса команд на регистр адреса;

y_2: *ЧтОП* – запуск *ОП* на выполнение микроопераций чтения информации из ячейки, адрес которой установлен на *РА*; по окончании этой МО считанная информация находится в регистре данных;

y_3: (*РК*):=(*РД*) – передача содержимого регистра данных на регистр команд;

y_4: (*СчАК*):=(*РК[А]*) – передача содержимого поля *А* регистра команд в *СчАК* (передача адреса);

y_5: (*СчАК*):=(*СчАК*)+1 – увеличение содержимого счётчика адреса команд на единицу;

y_6: *ЗпОП* – запуск *ОП* на запись в память содержимого регистра данных (ниже y_6 не используется).

Счётчик адреса команд предназначен для хранения адреса выполняемой команды и формирования адреса очередной команды программы при естественном порядке следования команд. В рассматриваемом УУ предполагается, что каждая команда занимает одну ячейку памяти, поэтому увеличение адреса выполняемой команды на единицу, осуществляемое микрооперацией y_5, даёт адрес очередной команды.

Регистр команд используется для хранения выполняемой команды программы на время ее обработки. В некоторых ЭВМ в *РК* хранится только код операции, а адресная часть команды поступает на *РА* или в другой узел процессора.

Управляющий автомат УУ формирует цикл выполнения команды в соответствии с ее типом, определяемым с помощью дешифрации на *ДшКОП* выполняемой команды. В каждом такте цикла *УА УУ* выдаёт необходимые для реализации конкретной команды управляющие сигналы y_1...y_n, поступающие в различные узлы процессора. Последовательность управляющих сигналов может зависеть от сигналов состояния отдельных узлов операционной части. Эти сигналы тоже поступают на вход *УА УУ*. Синхронизация работы управляющего автомата осуществляется генератором импульсов синхронизации ГС, вырабатывающим одну или несколько серий синхроимпульсов.

Рассматриваемое УУ функционирует следующим образом. Пусть в исходном состоянии в *СчАК* занесен адрес команды, подлежащей выполнению. Занесение адреса может производиться при начальном сбросе, передаче управления, прерывании или каким-либо иным образом.

В первом такте вырабатывается сигнал y_1: (*РА*):=(*СчАК*) и адрес, подлежащий выполнению команды, передаётся в *РА* оперативного ЗУ.

[*)] В дальнейшем управляющий сигнал y_i и соответствующая ему микрооперация обозначаются одним и тем же символом y_i и не различаются, если это не оговорено особо.

Во втором такте вырабатывается сигнал y_2: *ЧтОП* и в оперативной памяти выполняется микрооперация чтения команды. По окончании этой МО команда, подлежащая выполнению, оказывается в регистре данных *ОП*.

В третьем такте вырабатывается сигнал y_3, вызывающий передачу считанной из *ОП* команды в регистр команд. Одновременно вырабатывается сигнал y_5, по которому к содержимому *СчАК* добавляется единица и формируется адрес следующей команды программы. Эта МО может быть выполнена и во втором такте. Тем самым УУ оказывается подготовленным к выборке очередной команды.

Дальнейший ход работы УУ зависит от типа выбранной команды. Если выбранная из памяти команда относится к операционному типу, то в следующих тактах *УА УУ* вырабатывает управляющие сигналы, необходимые для выполнения задаваемой командой операции и, возможно, записи результата. По окончании операции (цикла выполнения команды) УУ переходит к формированию нового цикла выполнения команды, т. е. начинает повторять описанные выше первый, второй и третий такты. Поскольку при этом в начале первого такта в *СчАК* установлен адрес следующей команды, то процессор начинает обработку очередной команды программы, реализуя тем самым естественный порядок следования команд.

Но прежде, чем перейти к выполнению следующей команды, необходимо проверить, не возникли ли условия, препятствующие продолжению выполнения текущей программы. К таким условиям могут относиться запрос прерывания, задание режима покомандной обработки и др. Если такое условие возникает (на рис. 3.8 это условная вершина 7), то дальнейшее выполнение программы прекращается и управление передается операционной системе.

Если после выполнения третьего такта будет установлено, что выбранная команда относится к командам управления, то *УА УУ* в следующем такте вырабатывает управляющие сигналы, необходимые для определения адреса очередной команды, который может отличаться от находящегося к этому моменту в *СчАК* адреса выполняемой команды, увеличенного на единицу.

Существуют различные команды управления. Самые распространённые из них – команды безусловной передачи управления (безусловного перехода) и условной передачи управления (условного перехода) по значению какого-либо условия. Ниже обсуждаются эти команды, а их выполнение показано в правой части микропрограммы, изображенной на рис. 3.8.

Команда безусловной передачи управления (БП) указывает на то, что дальнейшее выполнение программы после этой команды следует продолжить, начиная с команды, адрес которой указан в адресном поле команды БП.

Команда условной передачи управления (УП) по заданному условию (например, при получении отрицательного результата после выполнения предшествующей команды – “УПО”) указывает на необходимость продолжения программы (начиная с команды, адрес которой записан в адресном поле команды УП) в том случае, если выполняется заданное условие. В противном случае, при рассматриваемом одноадресном формате команд, осуществляется переход к следующей за командой УП команде программы.

Тогда при выполнении команды БП в четвертом такте *УА АА* вырабатывается управляющий сигнал y_4, вызывающий передачу содержимого разрядов адресного поля *РК* в *СчАК*. На этом выполнение команды БП заканчивается, и *УА АА* переходит к формированию первого такта цикла выполнения новой команды. Поскольку в *СчАК* теперь находится адрес, указанный в выполненной команде БП, то очередная команда будет выбрана из памяти по этому адресу. Тем самым реализуется принудительный порядок следования команд.

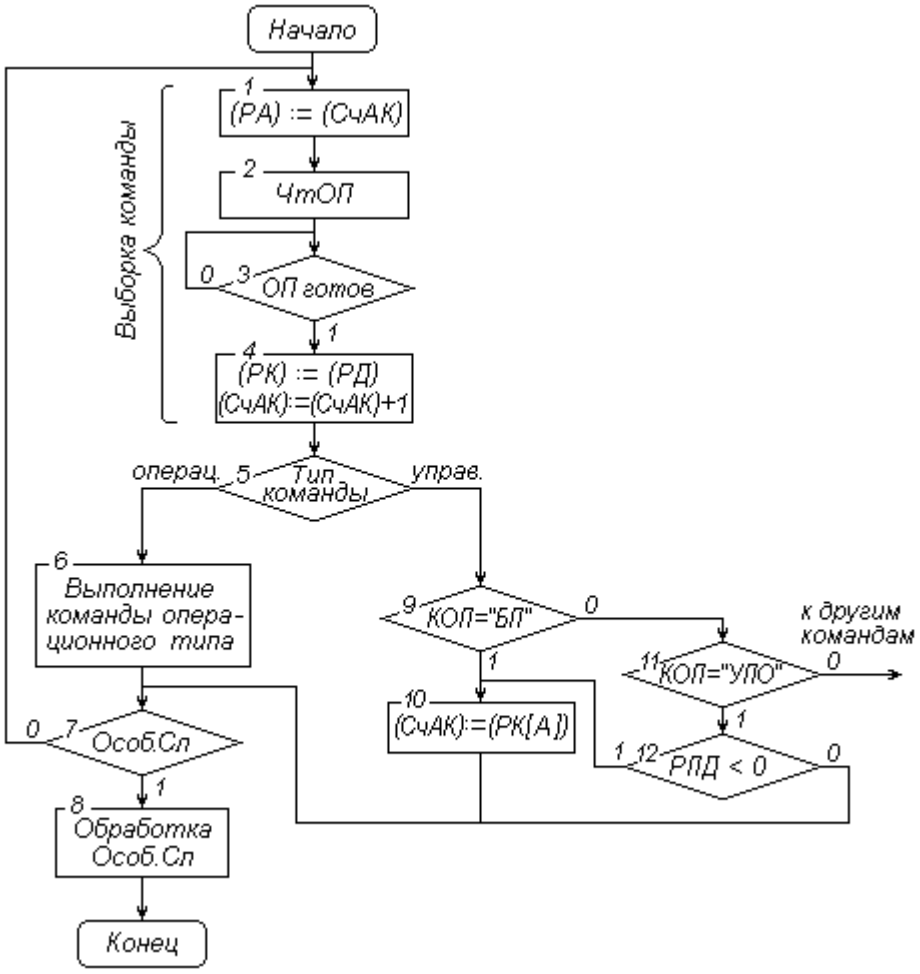


Рис. 3.8. Часть микропрограммы выполнения последовательности команд программы

При выполнении команды УП в четвертом такте проверяется заданное условие перехода, значение которого поступает в *УА УУ* из операционной части процессора. Если это условие выполняется, то вырабатывается управляющий сигнал y_4 и команда УП действует так же, как и команда БП. Если проверяемое условие не выполняется, то в четвертом такте не производится никаких действий и содержимое *СчАК* не изменяется.

Таким образом, рассмотренное УУ осуществляет управление выполнением последовательности команд, при котором естественный порядок следования команд является основным режимом, при необходимости нарушаемым командами управления.

Более сложные команды управления, в первую очередь вызов подпрограммы и управление циклом, следует проанализировать по литературе, указанной в начале раздела.

Управление выполнением операций

Функции УУ, относящиеся к управлению выполнением операций, обеспечивают автоматическое выполнение всех преобразований информации, соответствующих указанной в команде операционного типа операции над операндами, адреса которых заданы в команде. Для этого *УА УУ* необходимо выработать последовательность управляющих сигналов, позволяющих, сформировав исполнительные (см. ниже) адреса требуемых операндов, извлечь их из оперативной памяти, переслать их в АЛУ, реализовать в АЛУ микропрограмму заданной операции и при необходимости сформировать исполнительный адрес результата и записать полученный результат в ОП.

Способы адресации информации в ЭВМ

Извлечение команд и операндов из оперативной памяти ЭВМ, а также запись результатов в память осуществляется с помощью указания адреса ячейки, слова или байта оперативной памяти, в которой располагается необходимая информация. Адрес, по которому производится обращение, принято называть исполнительным адресом $\boldsymbol{A_E}$ (используется также термин физический адрес). Но в большинстве случаев в адресном поле команды исполнительный адрес не указывается, а записывается некоторый код, называемый далее исходным адресом $\boldsymbol{A_O}$, с помощью которого может быть получен исполнительный адрес. Способ получения исполнительного адреса из исходного называется способом адресации.

Основные причины применения различных способов адресации:

- необходимость получения меньшей разрядности адресного поля команды при адресации оперативной памяти большой емкости;
- требование удобства адресации элементов массивов данных в циклических программах, которые в каждом новом цикле обрабатывают новый элемент массива;
- обеспечение возможности написания программ, которые могут загружаться в любое свободное место оперативной памяти, т. е. перемещаемых программ, не зависящих от абсолютных адресов памяти.

В различных ЭВМ используются разные способы адресации, но все они базируются на следующих наиболее распространенных механизмах.

При *прямой адресации* в адресном поле команды указывается исполнительный адрес. Этот способ является наиболее простым с точки зрения его реализации на ЭВМ, но не обеспечивает перечисленных выше требований.

Непосредственная адресация – способ адресации, при котором в адресном поле команды размещается сам операнд, подлежащий обработке. Обычно этим способом адресуются константы, не изменяемые в процессе выполнения программы. Команды с непосредственной адресацией не требуют обращения к памяти за операндом и выполняются быстрее, однако его разрядность часто ограничивается.

Косвенная адресация – способ адресации, при котором в адресном поле команды указывают адрес ячейки (слова) памяти, в которой хранится исполнительный адрес. Таким образом, для извлечения операнда необходимо

обратиться в оперативную память по исходному адресу $\boldsymbol{A_O}$, который при данном способе адресации называют косвенным адресом, прочитать записанную в этой ячейке информацию и использовать ее как адрес для нового обращения к памяти. По этому адресу и будет располагаться искомый операнд. Для извлечения операнда потребуется выполнить два обращения к памяти, что увеличивает время выполнения команды с косвенной адресацией.

Относительная адресация (в некоторых случаях называемая иначе индексной) – способ адресации, при котором исходный адрес $\boldsymbol{A_O}$ состоит из адреса индекса $\boldsymbol{A_X}$ и смещения D: $\boldsymbol{A_O} = <\boldsymbol{A_X}, D>$. Адрес индекса является номером регистра или адресом ячейки (в зависимости от типа ЭВМ), где хранится значение индекса X. Исполнительный адрес $\boldsymbol{A_E}$ при относительной адресации получают посредством сложения содержимого регистра или ячейки $(\boldsymbol{A_X})$=X, указываемых адресом индекса $\boldsymbol{A_X}$, и смещения D, т. е.

$$\boldsymbol{A_E} = (\boldsymbol{A_X}) + D = X + D,$$

где $(\boldsymbol{A_X})$ – содержимое регистра с номером $\boldsymbol{A_X}$ или ячейки с адресом $\boldsymbol{A_X}$.

Такой способ позволяет либо эффективно составлять циклические программы обработки массивов информации, либо создавать перемещаемые в памяти программы, но предоставляют обе эти возможности одновременно.

Развитием относительной адресации является адресация с двойным индексированием (базированием и индексированием). Исходный адрес $\boldsymbol{A_O}$ при этом состоит из трех частей: адреса базы $\boldsymbol{A_B}$, адреса индекса $\boldsymbol{A_X}$, смещения D, $\boldsymbol{A_O} = <\boldsymbol{A_B}, \boldsymbol{A_X}, D>$. Адрес индекса и смещения имеют такое же назначение, как при индексной адресации. Адрес базы $\boldsymbol{A_B}$ определяет номер регистра или адрес ячейки памяти, где хранится величина, называемая базой B, т. е. $(\boldsymbol{A_B}) = B$. База используется аналогично индексу, а исполнительный адрес $\boldsymbol{A_E}$ получают при двойном индексировании посредством суммирования базы, индекса и смещения:

$$\boldsymbol{A_E} = (\boldsymbol{A_B}) + (\boldsymbol{A_X}) + D = B + X + D.$$

Данный способ позволяет использовать относительную адресацию для написания циклических программ, которые за счёт базирования могут при этом перемещаться в памяти.

Отдельные способы адресации могут сочетаться друг с другом. Например, косвенная адресация может использоваться совместно с относительной.

В 32-разрядных ПЭВМ реализованы все основные виды способов адресации: прямая, косвенная, индексная (относительная) и непосредственная. Особенности реализации каждого из способов адресации зависят от режима работы процессора – *реального* или *защищенного*. Реальный режим обеспечивает совместимость с первыми 16-разрядными процессорами.

В *защищенном* (РМ – protected mode) режиме физический адрес памяти представляется тридцатью двумя двоичными разрядами, что позволяет обращаться к оперативной памяти объемом 4 гигабайта (2^{32} байтов). В этом режиме

для формирования адреса используется более сложная схема, которая в общем виде представлена на рис. 3.9.

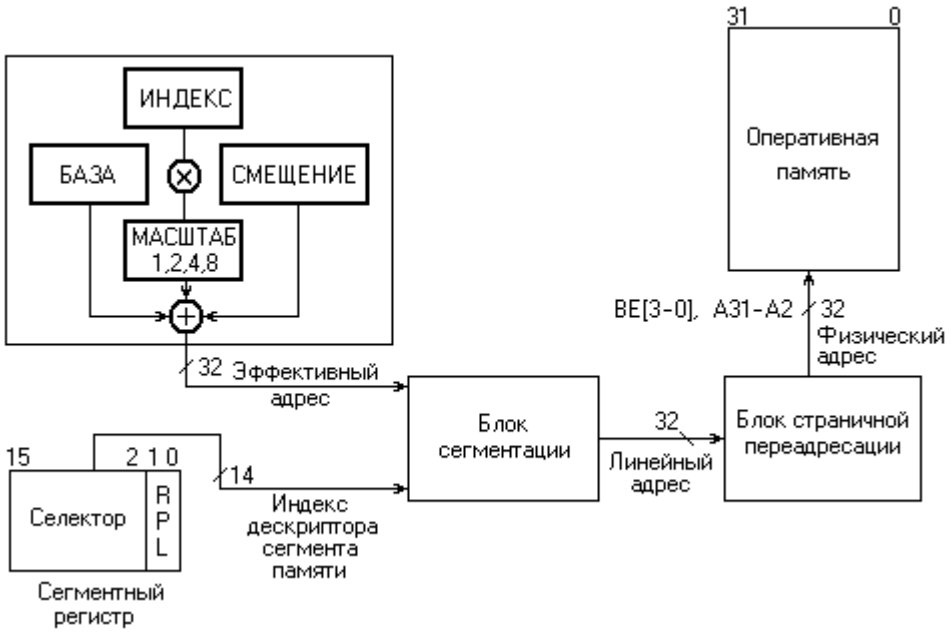


Рис. 3.9. Общая схема формирования физического адреса памяти в защищенном режиме

Обозначение RPL (requested privilege) на схеме означает хранимую в двух младших разрядах сегментного регистра привилегию, с которой производится обращение к памяти. Основное ее назначение – защита памяти (см. 4.1.4).

На этой схеме в процессе формирования физического адреса можно выделить три этапа:

формирование эффективного адреса (являющегося, по сути, относительным адресом в логическом или физическом сегментах оперативной памяти);

формирование линейного адреса – представляющего собой либо физический адрес оперативной памяти, либо виртуальный адрес логического пространства страничной памяти при выключенном или включенном блоке страничной переадресации соответственно;

формирование физического адреса оперативной памяти в блоке страничной переадресации.

Последний из этих этапов не является строго обязательным и может быть отключен установкой в “0” бита 31 (PG – paging enable) в управляющем регистре CR0 процессора.

Способы адресации реализуются на этапе вычисления эффективного адреса. При этом может использоваться до четырех компонент: *база, индекс, масштаб* и *смещение*. База, индекс и смещение рассматривались выше, а *масштаб (scale)* – это множитель, используемый для увеличения индекса, который может принимать значения 1, 2, 4 и 8, обеспечивая соответствие формату данных (байту, полуслову, слову, двойному слову) определяемой индексом позиции в области (массиве) данных. Масштаб задается непосредственно в коде команды.

Эффективный адрес определяется по соотношению

эффективный адрес = [*База*]+ [*Индекс* * *Масштаб*] + [*Смещение*],

где [ ] указывают необязательность соответствующей компоненты.

Такое формирование эффективного адреса позволяет реализовать все основные способы адресации (прямую, косвенную, относительную, непосредственную).

Система команд ПЭВМ обеспечивает еще два способа адресации, не связанных с обращением к оперативной памяти: прямая регистровая адресация, при которой операнд размещен в регистре процессора, и непосредственная адресация, при которой операнд находится в самой команде.

3.3.3. Способы построения устройств управления

По принципу реализации различают схемные и микропрограммные устройства управления.

Схемные устройства управления

Схемными называют устройства управления, в которых управляющие сигналы формируются с помощью специальных блоков, состоящих главным образом из комбинационных схем и элементов памяти (часто организованных в счетчики или регистры). Эти УУ также называют устройствами управления с жесткой логикой. Существуют две основные разновидности таких устройств: УУ на основе распределителей импульсов и УУ на основе автоматов Мили и Мура, интерпретирующих заданный набор микропрограмм.

Устройства управления на основе распределителей импульсов

Выполнение команды предполагает последовательное во времени осуществление некоторых преобразований в различных узлах ЭВМ. Для запуска этих преобразований необходимо в последовательные моменты времени подать управляющие сигналы в соответствующие узлы. В качестве формирователя таких управляющих сигналов удобно использовать схему, имеющую несколько выходов, сигналы на которых появляются поочередно. Этой схемой и является распределитель импульсов (сигналов).

Распределитель импульсов (РИ) имеет входы синхронизации *C* и начальной установки *R* и *n* выходов *T1*...*Tn* (рис. 3.10,а). При подаче на вход *R* сигнала начальной установки (*НУ*) РИ устанавливается в состояние, при котором выходной сигнал присутствует только на первом его выходе *T1*. При подаче синхроимпульса (*СИ*) на вход синхронизации выходной сигнал снимается с *i*-го выхода *Ti* и переходит на (*i* + 1)-й выход *Ti*+1, причем с *n*-го выхода сигнал переходит на выход *T1*. Таким образом, при подаче на вход *C* распределителя серии синхроимпульсов на его выходах будут поочередно появляться сигналы, как показано на временной диаграмме (рис. 3.10,б).

РИ можно реализовать на основе пары из счетчика и дешифратора или кольцевого сдвигающего регистра, а также на соединенных последовательно линиях задержки.

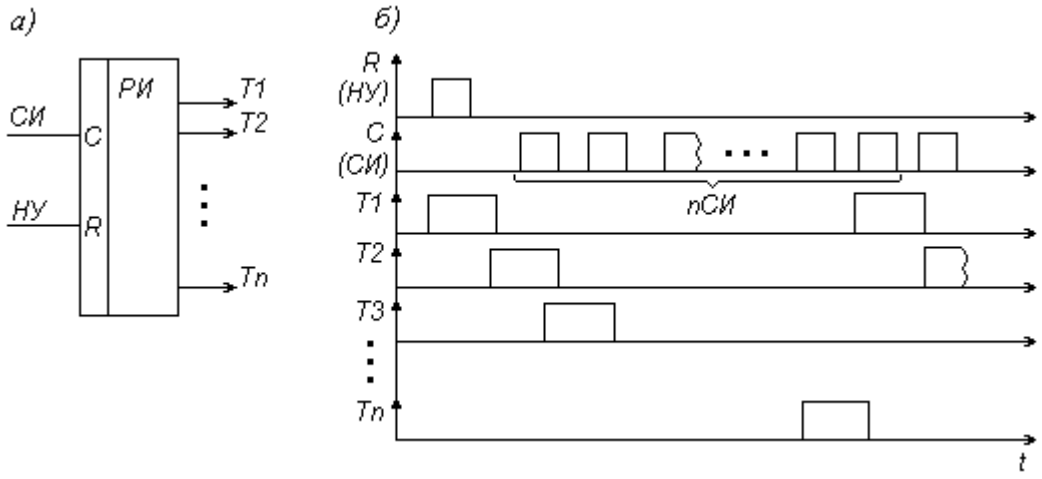


Рис. 3.10. Обозначение распределителя импульсов на схемах (а) и временная диаграмма его работы (б).

Построение устройства управления на основе РИ сводится к выбору количества распределителей и числа их выходов с последующим синтезом комбинационной схемы, формирующей управляющие сигналы y_i в нужные такты в соответствии с заданной микропрограммой.

Устройства управления на основе автоматов Мили и Мура, интерпретирующих заданную микропрограмму

Схемные устройства управления можно построить, используя автоматную интерпретацию микропрограмм. В этом случае схемно реализуется автомат Мили или Мура, закон функционирования которого соответствует заданной микропрограмме.

Автомат представляет собой последовательностную схему, полностью описываемую шестеркой элементов:

$$\mathbf{A} = \{A,\ Z,\ U,\ \varphi\ ,\ \psi\ ,\ a_0\},$$

где $A = \{a_0, a_1, ..., a_{m-1}\}$ – множество состояний автомата, $Z = \{z_0, z_1, ..., z_r\}$ и $U = \{u_0, u_1, ..., u_j\}$ – множества входных (входной алфавит) и выходных сигналов (выходной алфавит) соответственно; φ – функция переходов автомата, задающая отображение декартова произведения $A \times Z$ множеств A и Z на множество A, т.е. соответствие между всевозможными парами состояние – входной сигнал $\{a_i, a_j\}$ и новым состоянием a_k автомата; ψ – функция выходов, задающая отображение $A \times Z \to U$ для автомата Мили или отображение $A \to U$ для автомата Мура; $a_0 \in A$ – начальное состояние автомата.

Определяемый таким образом автомат называют абстрактным.

При технической реализации автомата с соответствующим заданной микропрограмме законом функционирования элементы множества A, т. е. состояния абстрактного автомата, кодируются и в схему автомата вводится набор элементов памяти по числу разрядов в кодах состояний. Множество Z входных сигналов отображается на множество сигналов логических условий $X = \{x_0, x_1, ..., x_p\}$, поступающих в УУ из операционной части процессора, а в схему автомата вводится столько входов, сколько имеется различных логических условий. Множество U выходных сигналов отображается на

множество управляющих сигналов (сигналов микроопераций) $Y = \{y_0, y_1, ..., y_n\}$, вырабатываемых УУ, а в схему автомата вводится n выходных шин по числу различных управляющих сигналов. Кроме того, для обеспечения дискретности автоматного времени в схеме автомата применяется синхронизация синхроимпульсами, подаваемыми на специальный вход синхронизации автомата. Такой автомат (рис. 3.11) называют структурным.

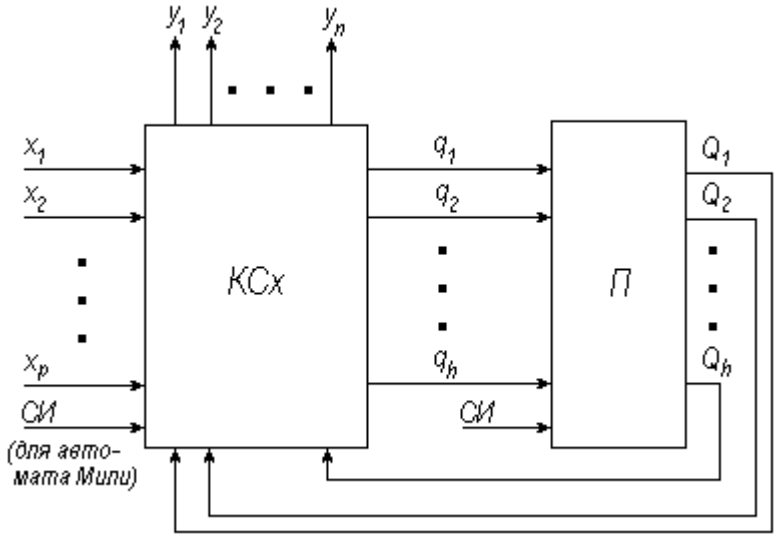


Рис. 3.11. Структурный автомат

Структурный автомат состоит из комбинационной схемы *КСх* и памяти *П*. На вход *КСх* поступают входные сигналы x_i (логические условия) и синхросигналы СИ, а на ее выходе формируются управляющие сигналы y_j. *КСх* вырабатывает также сигналы возбуждения q_k, вызывающие переключения элементов памяти, т.е. переходы автомата в новые состояния. Между *КСх* и *П* имеется обратная связь, по которой сигналы состояний Q_k элементов памяти автомата поступают на входы комбинационной схемы.

Имеется развитый формальный аппарат синтеза автоматов, вырабатывающих управляющие сигналы согласно заданной микропрограмме, изучаемый в дисциплине “Теория автоматов”.

Микропрограммные устройства управления

Альтернативой схемному управлению ЭВМ является микропрограммное управление, позволяющее построить достаточно экономичные УУ.

Модель микропрограммного управления была предложена М. Уилксом в 1951 г. В настоящее время микропрограммные устройства управления (МПУУ) используются в различных ЭВМ.

Структурная схема МПУУ представлена на рис. 3.12. В состав устройства входят постоянное запоминающее устройство (ПЗУ), регистр адреса микрокоманды (РАМК), регистр микрокоманды (РМК), схема формирования управляющих сигналов (СФУС), схема формирования адреса микрокоманды (СФАМК) и генератор тактовых синхронных импульсов (ГТИ).

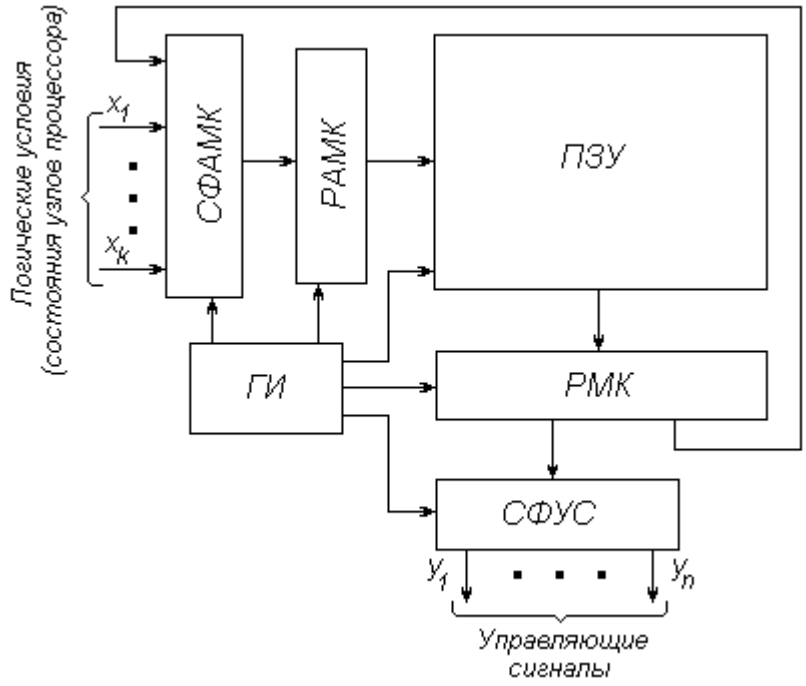


Рис. 3.12. Структурная схема микропрограммного устройства управления

В ПЗУ записаны все микрокоманды, которые могут вырабатываться конкретным микропрограммным устройством управления. Регистр адреса микрокоманды предназначен для хранения кода адреса микрокоманды во время выборки ее из ПЗУ. Регистр микрокоманды используется для хранения микрокоманды, выбранной из ПЗУ, на время ее выполнения. Схема формирования управляющих сигналов (СФУС) служит для выработки управляющих сигналов (сигналов микроопераций), задаваемых микрокомандой, находящейся на РМК. Схема формирования адреса микрокоманды вырабатывает адрес очередной подлежащей выполнению МК. Генератор тактовых синхроимпульсов вырабатывает синхронизирующие сигналы, обеспечивающие согласование функционирования всех узлов МПУУ в процессе выполнения микрокоманды.

Узлы рассмотренной структурной схемы фактически соответствуют первоначальной модели микропрограммного управления. Новым узлом является схема формирования управляющих сигналов, используемая для сокращения разрядности поля управляющих сигналов в микрокомандах.

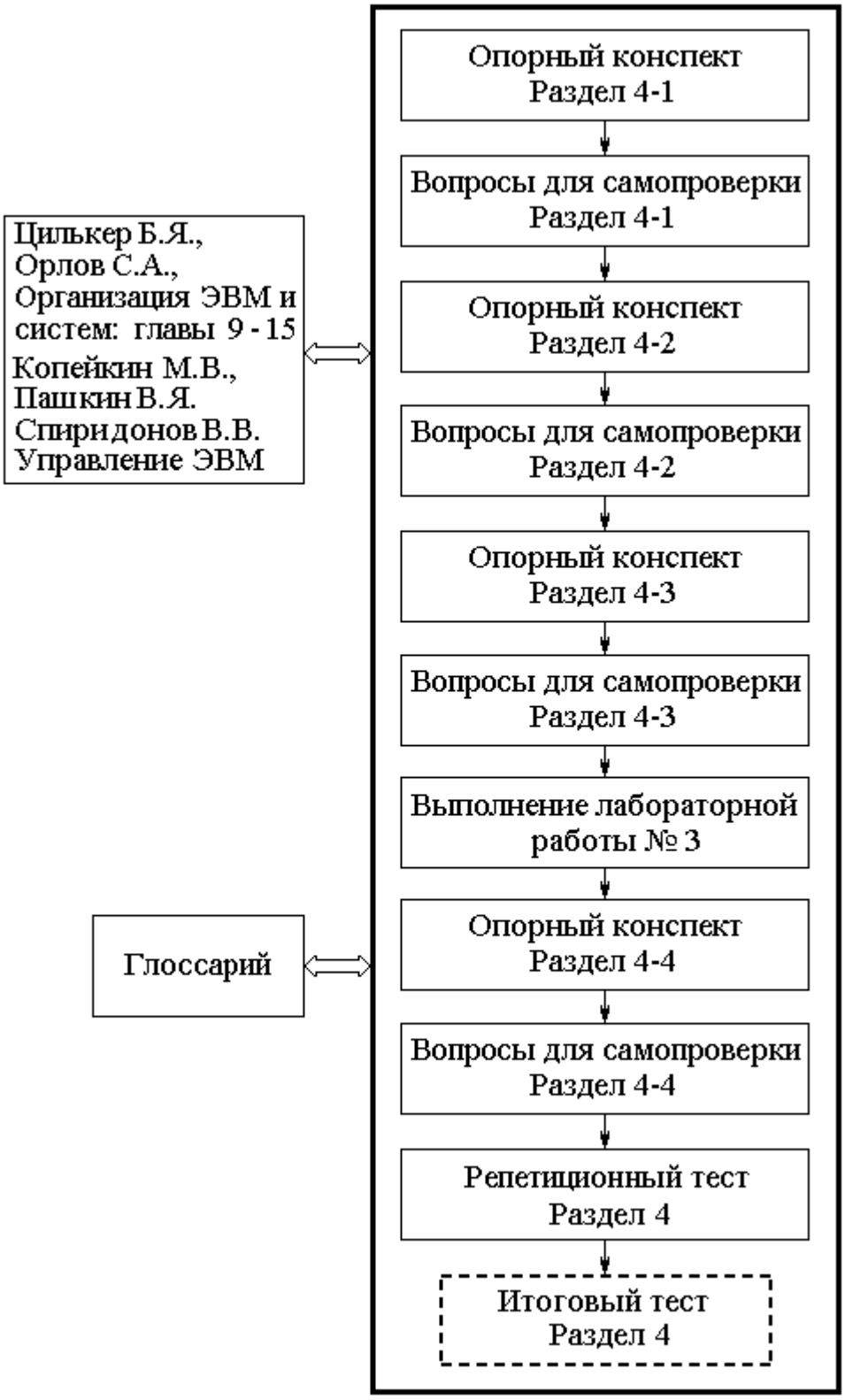
К достоинствам микропрограммного управления относят простоту проектирования, тестирования и эксплуатации; легкость внесения изменений в список команд; регулярность схемы, облегчающей ее изготовление методами технологии больших интегральных схем (БИС). А его недостаток – не очень высокое быстродействие, ограниченное временем цикла постоянной памяти.

Достоинством схемного управления является его высокое быстродействие. Поэтому в больших ЭВМ используется в основном схемное управление или его сочетание с микропрограммным. Однако трудоемкость проектирования, малая регулярность схем, сложность проверки и внесения изменений в схемные УУ усложняют их изготовление технологиями БИС.

Вопросы для самопроверки по теме 3.3

1. Каковы основные функции устройства управления?
2. Какие варианты реализации устройств управления существуют?
3. Что такое принудительный порядок следования команд?
4. Каковы основные этапы выполнения команды?
5. Чем различается выполнение операционных и управляющих команд?
6. Для чего используются различные способы адресации?
7. Как строятся схемные устройства управления?
8. К чему приводит зависимость между входными и управляющими сигналами при реализации разветвлений в микропрограмме?
9. Для чего служит постоянное ЗУ в микропрограммном устройстве управления?
10. Назовите достоинства и недостатки микропрограммных УУ.

Схема работы с разделом 4



Раздел 4. СИСТЕМНЫЕ СРЕДСТВА И АРХИТЕКТУРА ЭВМ

Четвертый раздел курса включает четыре темы: “***Системы прерывания программ и системы памяти ЭВМ”***, “***Организация ввода-вывода информации в ЭВМ”***, ***“Архитектура вычислительных систем”*** и “***Принципы по-***

строения аналоговых и гибридных ЭВМ”. После изучения каждой темы Вам следует ответить на вопросы для самопроверки.

В данном разделе выполняется лабораторная работа № 3. Рекомендации по ее выполнению приведены в методических указаниях к выполнению лабораторных работ в разделе 3.3.

Работа с разделом 4 завершается сдачей контрольного теста.

Для того, чтобы Вы смогли успешно ответить на вопросы контрольного теста, Вам предоставляется возможность поработать с репетиционным тестом. Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к учебнику [1].

4.1. Системы прерывания программ и системы памяти ЭВМ

При изучении данной темы Вы должны познакомиться с методологией и техническими средствами прерывания программ, используемыми для организации многопрограммного режима работы ЭВМ, а также организацией систем памяти ЭВМ в целом.

Для проверки изучения материала темы Вам предстоит ответить на вопросы для самопроверки.

Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к материалам файла Lect3.doc учебного сайта либо раздела сайта ord.com.ru/files/org_evm, к учебнику [1] или к материалам учебного пособия [7].

4.1.1. Системы прерывания программ

Многопрограммный режим работы ЭВМ требует при возникновении определенных условий прекращать работу над исполняемой программой и выполнять действия, обеспечивающие реакцию на возникшие условия. При этом предполагается возможность последующего возобновления прерванной программы и использования полученных до момента ее прекращения результатов.

Управление действиями по переключению программ возлагается на систему прерывания программ ЭВМ. Эта система представляет собой комплекс аппаратных и программных средств, обеспечивающих реакцию на события внутри ЭВМ или вне ее, моменты наступления которых имеют асинхронный характер либо трудно предсказуемы. События эти могут быть программно определяемыми (возникающими в результате выполнения команд программы) и программно независимыми.

Прерывание программы – это совокупность действий, производимых ЭВМ с целью временного прекращения выполнения текущей программы при возникновении определенных событий и передачи управления программам, предназначенным для обработки таких событий.

Сигнал о программно независимом событии, требующем прерывания, называют *запросом прерывания* (в англоязычной литературе – *Interrupt Request*, откуда и происходит используемое в ПЭВМ сокращение *IRQ*). Действия, вызываемые программно определяемыми событиями, часто имеют иные названия. Существуют и так называемые программные прерывания, которые представляют собой программный вызов системных функций.

Временну́ю диаграмму процесса прерывания для поступающего запроса можно представить так, как показано на рис. 4.1.

Рис. 4.1. Временна́я диаграмма прерывания (*Прог 1* – прерываемая программа, *ЗПр* – запрос прерывания, *Прог 2* – прерывающая программа, $t_р$ – время реакции на запрос, $t_з$ – время запоминания состояния , $t_в$ – время восстановления состояния)

Здесь обрабатываемая программа *Прог 1* прерывается поступившим запросом прерывания. Прекращение ее выполнения происходит с задержкой, связанной с завершением некоторых действий и выяснением возможности поступившего запроса прерывать исполняемую программу. Затрачиваемое на это время названо временем реакции, обозначенным через $t_р$. Источников запросов прерываний в ЭВМ, как правило, несколько, поэтому на данной фазе может возникнуть задача выбора для обработки одного из запросов.

После прекращения выполнения текущей программы начинается запоминание ее состояния, на которое затрачивается время запоминания состояния $t_з$, показанное на рис. 4.1 относительно большим. Это время может включать в себя как аппаратные действия, выполняемые системой, так и программные действия в начальной части программы обработки запроса, что происходит уже после передачи управления процессором программе *Прог 2* обработки ситуации, соответствующей запросу. По завершении программы обработки за время $t_в$ (которое может включать в себя и заключительные операции программы *Прог 2*) восстанавливается состояние ранее прерванной программы *Прог 1* и возобновляется ее выполнение.

Таким образом, основными функциями системы прерываний являются:

1. Прием запросов прерываний (с учетом их приоритетов при наличии нескольких запросов, требующих обработки).

1а. Выбор на обслуживание запроса с наивысшим приоритетом.

1б. Учет и управление приоритетом программ по отношению к запросам прерываний.

2. Прекращение выполнения текущей программы (с запоминанием ее состояния) и вход в прерывающую программу.

2а. Определение возможности прекращения выполнения текущей программы и выбор момента ее прекращения.

2б. Запоминание состояния прерываемой программы.

2в. Определение программы, которая должна быть вызвана для обслуживания запроса, и местоположения ее в памяти ЭВМ.

2г. Передача управления прерывающей программе и ее выполнение.

3. Возврат к прерываемой программе по завершении выполнения прерывающей.

3а. Восстановление состояния прерванной программы.

3б. Передача управления прерванной программе (на точку прерывания).

Для оценки системы прерываний используют характеристики:

1. *Время реакции* $t_р$ – время между появлением запроса прерывания и началом выполнения первой команды прерывающей программы.

2. *Время обслуживания прерывания* $t_{об}$ – время, затрачиваемое на запоминание и восстановление состояния прерываемой программы. В простейшем случае это время, затрачиваемое на переключение при переходе от прерываемой программы к прерывающей $t_{об} = t_з + t_в$.

3. *Количество типов (классов) прерываний* – количество различных по способу обработки или по характеру источников типов прерываний, имеющихся в системе.

4. *Количество входов запросов прерываний* – количество аппаратных линий (шин), на которые могут поступать запросы прерываний.

5. *Тип системы прерывания*, характеризующий способ, которым определяется источник запроса прерывания и местоположение сопоставленной ему программы обработки. Обычно по этому параметру различают опросные системы, векторные системы и системы с классами прерываний.

6. *Глубина прерывания* – максимальное количество программ, которые могут прерывать друг друга (вложенных прерываний).

Порядок выполнения прерываний

После выбора запроса прерывания при разрешении прерывания текущей программы определяется *допустимый момент ее прерывания*. Типовым вариантом является прерывание по окончании выполнения текущей команды. Однако если исполняемая команда пытается обратиться к не принадлежащей ей области памяти или выполнить какую-либо запрещенную для нее команду, то следует блокировать действия команды, не дожидаясь ее завершения.

После окончания выполнения текущей команды начинается этап *запоминания состояния* прерываемой программы. Запоминаемая информация должна полностью обеспечить возможность последующего возобновления выполнения прерываемой программы с той точки, в которой она прервана. К такой информации относится содержимое счетчика команд и регистра флагов (чтобы правильно выполнять переходы после возобновления программы).

Для корректного возобновления выполнения программы необходимо сохранить также содержимое регистров процессора, установленное прерываемой программой; служебную информацию, (маски, пороги), определяющую возможности прерывания данной программы; управляющую информацию, задающую особенности текущего режимы работы ЭВМ и др.

Состояние прерываемой программы в момент прерывания сохраняется в памяти ЭВМ, причем запоминание состояния организуют как аппаратными, так и программными средствами. Полностью аппаратный вариант более быстрый, но и более затратный. Место в памяти для запоминаемой информации о состоянии может определяться различными способами, но для обеспечения возмож-

ности “вложенности” прерываний (при глубине прерываний больше 1) удобно использовать стек.

Переход к прерывающей программе заключается в передаче ей управления загрузкой начального ее адреса в счетчик команд. Основной задачей на этом этапе является определение того, где в (оперативной) памяти располагается программа обработки того вида запросов прерываний, к которому относится выбранный запрос. Это называют *определением причины прерывания*, а выполняться оно может как программно, так и аппаратно.

При программном определении причины прерывания все запросы прерывания поступают на один общий вход, а для их обработки вызывается одна и та же обрабатывающая программа, поочередно опрашивающая все возможные источники (устройства) запросов прерывания. Такой способ прост в реализации, но медленен, особенно при большом количестве источников.

При аппаратном определении причины прерывания каждому входу, на который может поступать запрос прерывания, или однородной группе таких входов ставится в соответствие (жестко или перенастраиваемо) ячейка памяти или строка таблицы, в которой указан начальный адрес программы-обработчика соответствующего запроса прерываний.

После определения причины прерывания и передачи управления программе обработки запроса *выполняется обработка запроса*, по завершении которой осуществляется возврат к ранее прерванной программе.

Возврат производится посредством *восстановления состояния* этой программы, существовавшего на момент прерывания, и передачи управления той команде, перед (или *на*) которой она была прервана. Процедура восстановления состояния обратна запоминанию. Завершающие команды обработчика прерываний восстанавливают программно сохраненную информацию, а его последняя команда восстанавливает аппаратно сохраненную информацию и содержимое счетчика команд, существовавшее в момент прерывания.

Приоритетное обслуживание прерываний

При наличии в системе различных источников, способных формировать запросы прерываний, приходится решать задачи установления очередности (приоритета) приема и обработки этих запросов. Основных таких задач две:

- определять очередность приема запроса прерываний при одновременном наличии в системе нескольких запросов (поступивших одновременно или накопившихся в ожидании обслуживания);

- определять, могут ли принятые запросы прерывать выполнение текущей программы.

Этим задачам соответствуют два вида отношений приоритета: приоритет между запросами прерываний и приоритет между запросами прерываний и программами.

А) Способ задания приоритета и выбора запроса прерывания зависит от способа определения причины прерывания, рассмотренного выше.

Если причина прерывания определяется программно посредством опроса возможных источников запросов, то порядок опроса и задает приоритет, поскольку первым будет принят тот запрос, который выставило устройство, опрошенное ранее остальных, выставивших запросы.

При аппаратном определении причины прерывания часто используется схема приоритетного шифратора, содержащая цепочку блокировки менее приоритетных входов и шифратор номера запроса. Цепочка блокировки, называемая также “искателем левой единицы” или “дейзи-цепочкой”, показана на рис. 4.2.

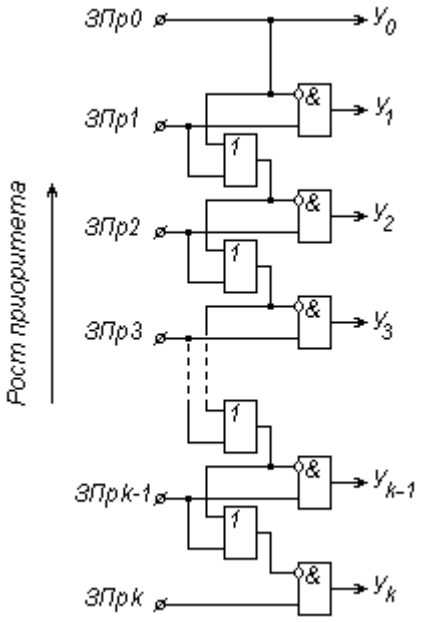


Рис. 4.2. Схема выбора наиболее приоритетного запроса

В этой схеме наиболее приоритетный запрос прерывания *ЗПр0* проходит на выход независимо от остальных запросов, т. е. $y_0 = ЗПр0$. Следующий по приоритету запрос *ЗПр1* пройдет на выход при своем появлении, только если отсутствует запрос *ЗПр0*, т.е., $y_1 = ЗПр1\ \neg ЗПр0$. Для прохождения следующего запроса требуется отсутствие первых двух, т. е. $y_2 = ЗПр2\ \neg ЗПр0\ \neg ЗПр1 = ЗПр2\ \neg(ЗПр0 \vee ЗПр1)$. В общем случае можно записать: $y_i = ЗПрi\ \neg(ЗПр0 \vee ЗПр1 \vee \ldots \vee ЗПрi\text{-}1)$. Эти соотношения и реализует схема, показанная на рис. 4.2.

Аппаратные схемы приоритетного выбора срабатывают быстро (при количестве входов запросов до нескольких десятков). Однако приоритет задан жестко и определяется порядком подключения запросов к входам.

Б) При задании приоритетов между запросами прерываний и программами используются механизм масок и механизм порогов.

Механизм масок позволяет каждой программе разрешать или запрещать отдельно для каждого запроса прерывания прерывать данную программу. Для этой цели используется двоичный код, называемый маской программы или маской прерывания, с разрядностью, соответствующей количеству входов запросов прерываний, для которых разрешаются или запрещаются прерывания. Каждому входу прерывания соответствует свой разряд маски программы (прерывания), установка которого в единицу (или, наоборот, в нуль) запрещает запросам прерываний, поступающим на этот вход, прерывать данную программу.

Код маски хранится в специальном регистре, в который он устанавливается либо специальными командами, либо общими командами ввода-вывода, которые могут адресовать данный регистр как устройство.

Механизм порогов устроен похожим образом. Каждой программе присваивается некоторое целое число – код уровня. Прервать программу могут только те запросы, номер которых меньше уровня программы.

Код маски и номер уровня программы относятся к информации о состоянии программы, сохраняемой при ее прерывании.

4.1.2. Система прерываний ПЭВМ

Система прерываний ПЭВМ реализует, в целом, типовой набор функций, используя для этого достаточно стандартные механизмы.

Выполнение аппаратно реализуемых функций прерываний в ПЭВМ производится контроллером (точнее, двумя контроллерами) прерываний. Прерывания в ПЭВМ разделяются на две большие группы: *аппаратные* и *программные*.

Аппаратные прерывания вызываются сигналами на специальных входах процессора. Причем в рамках аппаратных прерываний различают немаскируемые (*NMI* – non maskable interrupt) и маскируемые прерывания (точнее, запросы прерываний).

Программные прерывания разделяют на особые случаи и обращения к системным процедурам, например функциям BIOS или DOS, вызываемым командой процессора INT (для функций DOC – это INT 21h).

Особые случаи при выполнении программ, например деление на нуль; недействительный код операции; обращение к информации, отсутствующей в оперативной памяти, и т. п., получили название *исключений*. В зависимости от особенностей возникновения и необходимой реакции, исключения подразделяют на *отказы* (fault) и *ловушки* (trap), обрабатываемые соответственно до и после выполнения вызвавшей их команды, и *аварийные завершения* (abort) – серьезные ошибки, не позволяющие точно установить вызвавшую их команду.

Местоположение в памяти (адрес) программы обработки прерывания или исключения определяется по таблице (дескрипторов) прерываний с помощью указателя, или, иначе, вектора прерывания, задающего строку таблицы прерываний, в которой хранится адрес требуемой программы (обработчика прерывания). Этот вектор для программных прерываний задается командой, для исключений он генерируется внутри процессора, для немаскируемых аппаратных прерываний всегда фиксирован, а для маскируемых аппаратных прерываний формируется контроллером прерываний.

Позднее появился расширенный контроллер прерываний (APIC – advanced programmable interrupt controller), основным назначением которого является организация приема запросов прерываний в многопроцессорной системе. Но он используется и в однопроцессорных системах, обеспечивая в определенных режимах совместимость с обычным контроллером прерываний. Он имеет также дополнительные виды аппаратных прерываний, в частности межпроцессорные, от таймера мониторинга производительности и др.

4.1.3. Системы памяти ЭВМ

Классификация и характеристики систем памяти

Особенности построения систем памяти (СП) связаны с их структурой, принципами функционирования, логического взаимодействия и пр. К основным классификационным признакам систем памяти можно отнести следующие.

1. ***Количество уровней***, т. е. отличных по своему назначению или конструктивным характеристикам запоминающих устройств. По этому признаку можно разделять СП на *одноуровневые* и *многоуровневые*.

2. ***Характер связей между уровнями***. Связи между уровнями системы памяти, допускающие обмен информацией между ними, определяют допустимые потоки данных в системе и ее структуру. По характеру связей можно выделить:

- *централизованные* СП, в которых обмен информацией между ЗУ различных уровней осуществляется через какое-либо одно ЗУ, обычно через оперативную память;
- *линейные* СП, в которых обмен информацией возможен только между смежными уровнями системы (например, кэш – оперативная память – жесткие диски);
- *смешанные* СП, обладающие связями, характерными как для централизованных, так и для линейных СП (например, кэш – оперативная память – жесткий диск и CD ROM, имеющие одинаковые связи с оперативной памятью);
- СП *со структурой полного графа*, включающие в себя устройства, позволяющие устанавливать связи для обмена информацией между двумя любыми уровнями.

3. ***Тип разбиения адресного пространства памяти***. Обычно память разделяется на логические блоки для упрощения управления. По этому признаку различают системы памяти:

- *без разделения* поля памяти на блоки;
- *со страничной памятью*, адресное пространство которых разделено на участки одинакового размера, называемые страницами;
- *с сегментированием памяти*, в которых память разделяется на сегменты, размер которых жестко не задается;
- *с двухуровневым* (странично-сегментным) *разделением* поля памяти.

4. ***Количество обслуживаемых системой памяти процессоров*** – признак, по которому различают СП *однопроцессорных* и *многопроцессорных* ЭВМ и систем.

5. ***Порядок обслуживания обращений к ЗУ нижних уровней***. По этому признаку можно различать системы с обслуживанием обращений *в порядке поступления* и *с диспетчеризацией обращений*, т. е. обслуживанием их в порядке, позволяющем уменьшить среднее время обслуживания обращения.

С целью более полного учета характера функционирования и окружения СП при определении критерия ее оценки следует рассматривать эту систему как компоненту вычислительной машины (системы).

Критерий оценки СП должен включать основные характеристики оцениваемой системы, к которым в рассматриваемом случае относятся емкость системы памяти, среднее время обращения к ней, пропускная способность, стоимость и надежность. Ряд характеристик, например радиационная устойчивость, габариты, масса, энергопотребление, в типовых применениях может не учитываться. Хотя, если речь идет, например, о мобильных системах, последние три из названных характеристик имеют существенное значение.

4.1.4. Страничная и сегментная организация памяти. Защита памяти

Многоуровневая организация памяти ЭВМ требует управления размещением информации в системе памяти и ее передачей между уровнями памяти. Эти задачи решаются совместно аппаратными средствами ЭВМ и средствами операционной системы. К ним относятся:

- разделение памяти между несколькими программами, выполняемыми в ЭВМ, работающей в многопрограммном режиме;
- упрощение для прикладных программ (и программистов) использования многоуровневой памяти, вплоть до представления ее в виде логически однородного пространства памяти (“виртуальной памяти”);
- выделение места для данных и команд программы в верхних уровнях памяти (кэш, оперативная память);
- определение местонахождения требуемой информации в системе памяти;
- обмен информацией между уровнями памяти;
- защита информации различных программ от сбоев и несанкционированных воздействий.

Страничная и сегментная память

Общий подход к решению перечисленных задач – концепция виртуальной памяти, предложенная еще в конце 50-х годов прошлого столетия, – фактически явился основой для организации управления многоуровневой памятью всех современных ЭВМ.

Виртуальную память можно рассматривать как отображение некоторого линейно упорядоченного множества (множества логических адресов) на множества адресов, имеющихся в вычислительной системе запоминающих устройств, называемых физическими адресами. Более того, такое отображение может задаваться отдельно для каждой из выполняемых в системе программ. Кроме того, это отображение не является статическим (как отображение-функция в математике), а может изменяться в любой момент времени.

Основными механизмами, используемыми для реализации виртуальной памяти ЭВМ, служат сегментное и страничное разделение памяти.

Сегментное разделение предполагает деление памяти на логические блоки – сегменты произвольной (переменной) длины. Это разделение чаще связывается с логической структурой выполняемых программ и кроме общих задач, связанных с управлением памятью, решает проблему изоляции программ друг от друга.

Страничное разделение предполагает деление памяти на блоки постоянной длины – страницы. Это разделение, напротив, в большей степени ориентируется на физическую память и управление передачей информации.

В ЭВМ могут использоваться один или оба этих механизма.

Оба механизма основаны на том, что управлять блоками адресов памяти проще, чем каждым адресом в отдельности. При этом соответствие между логическими и физическими адресами задается с помощью *таблиц*, содержимое которых корректируется при каждой пересылке информации между ступенями памяти. Эти таблицы могут носить различные названия и иметь разную структуру, но задача у них одинакова: отображать существующее в текущий момент времени соответствие между логическими и физическими адресами.

Организацию сегментного и страничного механизма можно проиллюстрировать на примере архитектуры процессоров Intel IA-86, реализующей оба этих механизма совместно.

Здесь сегментация обеспечивает изоляцию отдельных модулей программ, данных и стека так, что несколько программ (или задач) могут выполняться на одном и том же процессоре без влияния друг на друга.

Страничный механизм обеспечивает реализацию традиционной виртуальной системы памяти со страничными запросами и главным образом предназначен для ускорения процедур обмена между оперативной и дисковой памятью. Последнее достигается за счет того, что при больших размерах логических сегментов обмен может производиться страницами, типовой размер которых составляет 4K байт, а не целыми сегментами. Страничный механизм также может использоваться и для изоляции различных задач.

Сегментный механизм преобразует логический адрес, сформированный исполняемой командой, в адрес в пространстве линейных адресов, разделенном на блоки – сегменты. Схема преобразования показана на рис. 4.3. В этой схеме логический адрес представляется в виде селектора, указывающего положение описателя (дескриптора) сегмента в таблице дескрипторов, и смещения. Дескриптор сегмента указывает на начало (базу) сегмента в пространстве линейных адресов. Линейный адрес получается посредством сложения базы сегмента со смещением, заданным второй частью логического адреса.

Дескриптор также содержит информацию о размере сегмента и его свойствах (типе, правах доступа, привилегии и др.). Таблицы дескрипторов располагаются в оперативной памяти.

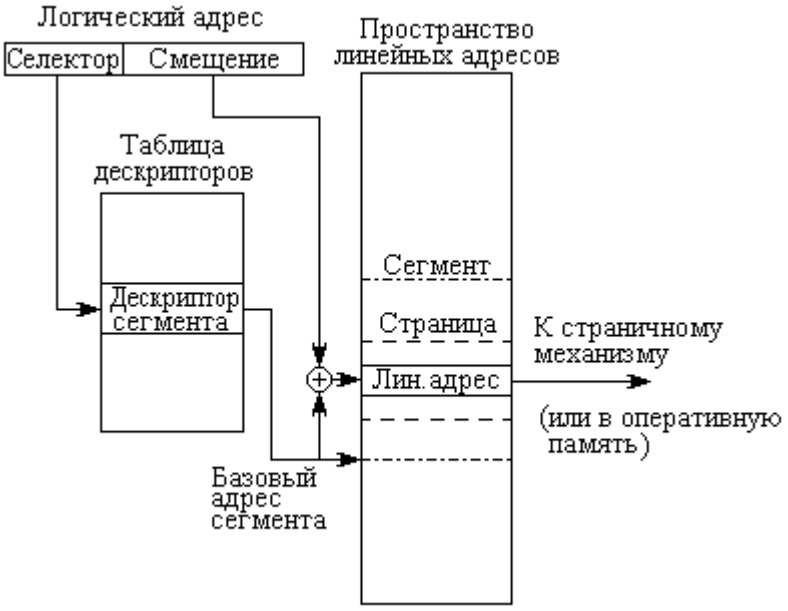


Рис. 4.3. Преобразование адреса в блоке сегментации

Если далее не используется страничный механизм, то полученный линейный адрес непосредственно является физическим адресом. В противном случае линейный адрес подвергается дальнейшему преобразованию в блоке страниц. Схема этого преобразования показана на рис. 4.4.

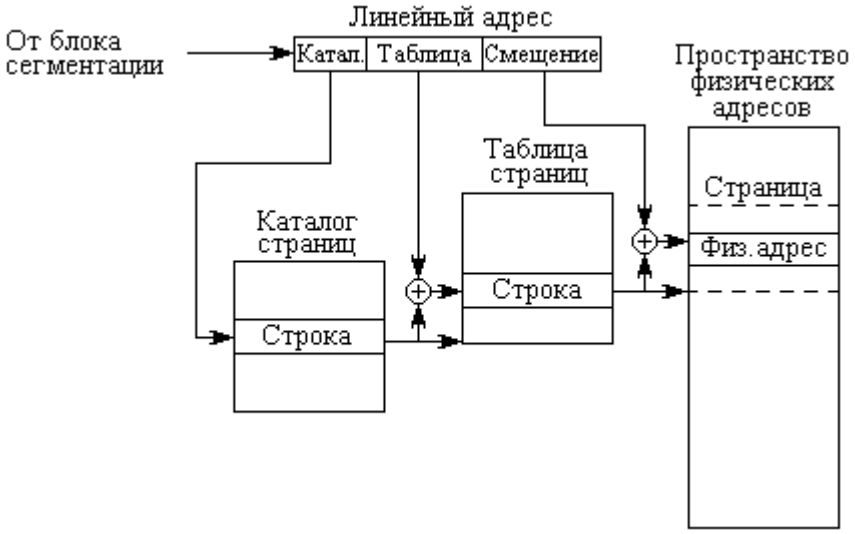


Рис. 4.4. Преобразование адреса в блоке страниц

Механизм страниц поддерживает окружение “виртуальной памяти”, в которой большое пространство линейных адресов моделируется с помощью меньшего количества физической памяти (оперативной или постоянной) и дисковой памяти. При использовании страничного механизма каждый сегмент разделяется на страницы (обычно по 4 Кбайт каждая), хранящиеся либо в оперативной памяти, либо на диске. Когда программа пытается обратиться по адресу в линейном пространстве адресов, процессор использует каталог страниц и таблицы страниц для преобразования (трансляции) линейного адреса в физические адреса, а затем производит запрошенную операцию (чтение или запись) с ячейкой памяти. Если страница, к которой производится доступ, отсутствует в фи-

зической памяти, то процессор прерывает выполнение программы (генерируя исключение страничного сбоя). Операционная система считывает страницу с диска в физическую память, а затем продолжает выполнение программы.

Двухуровневая организация таблиц, часто используемая в страничных механизмах, позволяет сопоставить каждому сегменту задачи свою таблицу страниц. Привязка таблиц к задачам производится с помощью специального регистра, указывающего адрес таблицы – каталога страниц. Все таблицы располагаются в оперативной памяти.

В схеме, приведенной на рис. 4.4, линейный адрес памяти, получаемый из блока сегментации, представляется в виде трех частей: номера строки каталога страниц, номера строки таблицы страниц и смещения физического адреса в странице, задаваемого в младших разрядах линейного адреса.

Первые две компоненты адреса используются для выборки информации из каталога (страниц) таблиц и таблицы страниц так, как показано на рис. 4.4. Номер строки каталога страниц добавляется к базовому адресу каталога, хранящемуся в управляющем регистре процессора. Этот адрес обязательно имеет нули в 12-ти младших разрядах, т. е. выровнен по границе 4K байт, которые и занимает каталог, содержащий 1024 строки по 4 байта. Адрес, полученный после сложения базового адреса каталога и десяти старших разрядов линейного адреса, указывает строку каталога страниц. Эта строка в старших разрядах содержит начальный адрес таблицы страниц, также выровненный по границе 4K байт – размеру таблицы страниц, имеющей такой же формат, как и каталог (таблиц) страниц. В младших разрядах строк каталога и таблиц страниц имеются несколько служебных разрядов.

Таблицы, используемые при формировании физического адреса, расположены в оперативной памяти. Чтение информации из них при каждом обращении втрое увеличивает его время, поэтому желательно обойтись без обращения к таблицам при каждом доступе к оперативной памяти. Для этого используется небольшая ассоциативная память, называемая буфером трансляции адресов – TLB (*translation lookahead buffer*), в котором хранят результаты последних преобразований адресов для нескольких страниц.

Защита памяти

В многопрограммном окружении при одновременном выполнении нескольких программ, разделяющих общую оперативную память, становится необходимым предотвращение ошибочного или намеренного доступа одной (прикладной) программы к данным или исполняемым кодам другой. Предотвращение такого доступа и называют защитой памяти.

Механизмы защиты памяти включают в себя аппаратные и программные средства и могут реализовываться по-разному.

Одним из первых таких механизмов была защита по границам (диапазону), в котором каждой программе сопоставлялись верхняя и нижняя границы – начальный и конечный адреса занимаемого программой участка оперативной памяти. Граничные адреса хранились в специальных регистрах и при обращении программы к памяти адрес, по которому программа собиралась обратиться

к памяти, сравнивался с этими границами. При выходе адреса за установленные в регистрах границы обращение не выполнялось и вырабатывался соответствующий сигнал. Значения границ заносились в регистры операционной системой при передаче управления программе.

Этот механизм имел существенный недостаток. Он требовал, чтобы программа со своими данными размещалась в одном непрерывном участке памяти. Конечно, можно было сделать несколько пар регистров границ, по числу занимаемых программой участков памяти, но это усложняло механизм защиты и увеличивало время, затрачиваемое на установку регистров и сравнение адреса обращения со всеми границами.

Другим вариантом является поблочная защита. В этом случае память разделяется на блоки равного размера, которыми она и выделяется исполняемой программе. Каждому блоку сопоставляется ячейка в специальной памяти ключей защиты, а программе присваивается некоторое значение ключа – ключ программы. При обращении программы к памяти значение ключа из ячейки памяти ключей защиты, соответствующей блоку, к которому относится запрашиваемый программой адрес, сравнивается с ключом программы. Обращение разрешается при совпадении ключей. В противном случае обращение блокируется и вырабатывается сигнал нарушения защиты памяти. Исключением является ключ, равный 0, с которым можно обращаться к любому блоку памяти, присваиваемый программам операционной системы.

Кроме собственно значения ключа в памяти ключей защиты могут храниться и дополнительные флаги, указывающие, какой вид доступа следует контролировать: чтение, запись или оба.

Механизмом защиты является и рассмотренная выше сегментация, которая разрешает программе доступ только либо к общим сегментам, либо к сегментам, дескрипторы которых занесены в локальную таблицу дескрипторов программы.

Механизм защиты должен быть сам защищен от возможности изменения содержимого регистров границ, ключей защиты или сегментных таблиц и регистров, на них указывающих. Для этого команды, которые изменяют содержимое названных компонент, должны быть разрешены к исполнению только системным программам и запрещены для прикладных программ.

Для разделения команд на группы, разрешенные или запрещенные к исполнению различным программам, обычно служит механизм привилегий. В этом случае каждой программе присваивается уровень привилегий (от двух до четырех уровней), каждому из которых соответствует свой набор команд, разрешенных к исполнению. Программы операционной системы имеют наивысший уровень привилегий и могут выполнять все команды. Прикладные программы имеют низший уровень привилегий, запрещающий выполнение команд управления системой, в частности, средствами защиты памяти. Попытка выполнить такую команду прикладной программой блокируется с формированием соответствующего сигнала, обычно вызывающего прерывание.

4.1.5. Управление обменом с внешней памятью. Дисковые массивы

Задачи управления обменом между ступенями многоуровневой памяти во многом являются общими для всех типов смежных уровней.

Алгоритмы замещения

Одной из основных общих задач управления обменом между уровнями памяти является задача освобождения места для ввода новой информации в верхний уровень, когда свободного места в нем недостаточно. Для решения этой задачи используются специальные правила, называемые алгоритмами замещения, с помощью которых и выбираются блоки, удаляемые из памяти при отсутствии в ней свободного места.

Логика алгоритмов замещения не зависит от того, к какой ступени памяти они применяются. Но для более быстрых ступеней время их срабатывания должно быть малым, что ограничивает их сложность. В случае обмена между дисковой и оперативной памятью требования к быстродействию алгоритмов не столь критичны, поэтому существуют различные по сложности схемы.

Идеальный алгоритм должен удалять из памяти тот блок (страницу), обращения к которому больше не будет или не будет дольше всего. Но так как наперед трасса обращений неизвестна, то такой алгоритм построить нельзя. Приходится строить алгоритмы, основанные на каких-либо предположениях.

Простейшим алгоритмом является случайный выбор удаляемых блоков. Но в этом случае можно удалить используемый блок или тот блок, который будет использоваться в ближайшее время. Более сложные алгоритмы связаны с построением списков-очередей, в которых блоки размещаются в соответствии с моментами или количеством обращений к ним. Один из распространенных алгоритмов – удаление наиболее давно использованного блока (НДИ). Другим примером может служить удаление наиболее редко используемого блока.

Диспетчеризация обменов с внешней памятью

Управление порядком обслуживания (диспетчеризация) обращений к запоминающим устройствам с механическими перемещениями носителей информации и механизмов чтения/записи является способом повышения производительности памяти. Такой подход используется, например, при обработке обращений к жесткому диску.

В обычном режиме обращения к диску обслуживаются в порядке их поступления. Но время перемещения блока головок чтения/записи зависит от количества цилиндров, на которое их требуется переместить. Поэтому и время обслуживания обращения зависит от взаимного расположения на диске запрошенного файла и файла предыдущего обслуженного обращения.

Следовательно, можно найти порядок обслуживания обращений, который уменьшает времея перемещения головок. Такую дисциплину обслуживания называют МВДП – “минимальное время доступа – первый”.

Однако было установлено, что алгоритм выбора на обслуживание обращений, адресующихся к ближайшим цилиндрам, приводит к “дискриминации” обращений, адресующихся к цилиндрам, более удаленным от средней дорожки

(т. е. расположенным на крайних и на центральных дорожках). Время ожидания в очереди для таких обращений существенно возрастает.

Поэтому используют и несколько иной порядок обслуживания, который не приводит к чрезмерным задержкам в обслуживании отдельных обращений. Такая дисциплина обслуживания называется дисциплиной сканирования (СКАН). В ней блок головок чтения/записи, обслуживая обращения, перемещается от края диска к центру и обратно, меняя направление движения лишь у первого и последнего из цилиндров, на которых расположены файлы обращений. Из очереди выбирается к обслуживанию обращение, адресованное к цилиндру (дорожке), находящемуся ближе всего к головкам по направлению их движения.

Дисциплины СКАН и МВДП могут быть реализованы как программно, так и аппаратно. Например, аппаратная поддержка МВДП некоторыми жесткими дисками получила названия NCQ (native command queue) и TCQ (tagged command queue). Известны и другие разновидности дисциплин обслуживания обращений.

Дисковые массивы

Еще одним способом повышения производительности и надежности дисковой памяти стало построение дисковых массивов. Технология RAID (*Redundant Array of Independent Disks* – избыточный массив независимых дисков) объединяет несколько недорогих жестких дисков для увеличения производительности, объема и надежности, по сравнению с одиночным диском. ЭВМ должна видеть такой массив дисков как один логический диск.

Если просто объединить несколько дисков в неизбыточный массив, то среднее время между отказами (СВМО) будет равно СВМО одного диска, деленному на количество дисков. Такой показатель слишком мал для приложений, критичных к аппаратным сбоям. Улучшить его можно применяя реализуемую различным образом избыточность при хранении информации.

В RAID системах для повышения надежности и производительности используются комбинации трех основных механизмов:

- организация “зеркальных” дисков, т. е. полное дублирование хранимой информации;
- подсчет контрольных кодов (четность, коды Хэмминга), позволяющих восстановить информацию при сбое;
- распределение информации по различным дискам массива, что повышает возможности параллельной работы дисков.

Определено несколько типов дисковых RAID массивов, различающихся по своим особенностям и производительности.

Вопросы для самопроверки по теме 4.1

1. Каковы основные функции системы прерывания программ?
2. Для чего необходимо приоритетное обслуживание прерываний?
3. Что такое вектор прерывания?
4. Сколько входов имеет контроллер прерываний ПЭВМ?
5. Перечислите основные характеристики системы памяти ЭВМ.

6. Чем различается сегментная и страничная организация памяти?
7. Что такое защита памяти?
8. Что такое уровень привилегии программы?
9. Что такое алгоритм замещения?
10. Для чего используются RAID массивы?

4.2. Организация ввода-вывода информации в ЭВМ

При изучении данной темы Вы должны познакомиться с принципами и средствами, используемыми при организации ввода-вывода в ЭВМ, а также организацией систем памяти ЭВМ в целом.

Для проверки изучения материала темы Вам предстоит ответить на вопросы для самопроверки.

Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к учебнику [1] или к материалам файла Lect3.doc учебного сайта либо раздела сайта ord.com.ru/files/org_evm.

4.2.1. Принципы организации ввода-вывода в ЭВМ

Операции ввода-вывода в ЭВМ представляют собой процедуры обмена информацией между оперативной памятью и внешними устройствами.

В ЭВМ первых поколений устройства ввода-вывода, как правило, непосредственно определялись архитектурой ЭВМ и не могли заменяться по типу. Эти устройства подключались по индивидуальным шинам, для каждого из них имелись собственные блоки управления, которые получали специальные команды процессора для конкретных устройств. Возможности варьировать состав или характеристики подключаемых устройств были очень ограничены.

Интерфейс

По мере увеличения числа областей применения ЭВМ, расширения спектра периферийных устройств и их характеристик возникла необходимость в выборе определенного набора периферийных устройств для конкретной ЭВМ и возможности последующей его модификации.

Это потребовало новых концепций и принципов организации ввода-вывода в ЭВМ. Ключевой идеей стала концепция *интерфейса*. Она предполагала построение единой системы шин для связи с различными периферийными устройствами. Для этой системы шин оговаривались: расположение и функциональное назначение шин, протоколы обмена, электрические и физические параметры линий и передаваемых по ним сигналов.

Такая унификация позволяла осуществлять стандартное подключение к ЭВМ любого периферийного устройства, поддерживающего параметры интерфейса. Кроме того, должны были быть унифицированы *форматы данных*, передаваемых по интерфейсу, а также в первоначальной концепции *команды процессора*, управляющие интерфейсом.

Конечно, в периферийные устройства стало необходимо встраивать управляющие устройства, которые обеспечивали подключение к интерфейсу.

Одной из первых массовых реализаций концепции такого интерфейса стал интерфейс системы IBM-360 (Единой системы ЭВМ).

Так был сделан шаг от специализированных интерфейсов к универсальным.

Появление универсального интерфейса способствовало дальнейшему расширению спектра периферийных устройств, которые теперь могли производиться компаниями, непосредственно не изготавливающими ЭВМ.

Иерархия управления

Скорости работы периферийных устройств, как правило, значительно ниже скорости работы процессора, и возлагать непосредственное управление операцией ввода-вывода на процессор нецелесообразно, хотя такие режимы работы и существуют. Поэтому управление операциями ввода-вывода реализуется с помощью иерархии управляющих блоков.

На верхнем уровне находится *процессор*, который определяет общий характер и основные параметры операции: направление передачи (ввод или вывод), количество передаваемой информации, место в оперативной памяти, куда или откуда будет передаваться информация. В случае обмена с внешней памятью указывается ее область, участвующая в обмене.

На следующем уровне находится устройство, которое преобразует команды процессора в последовательности управляющих сигналов, передаваемых по интерфейсу ввода-вывода, а после начала операции передает или получает и собственно передаваемую информацию. В качестве такого устройства в ЭВМ выступает *канал ввода-вывода* или *контроллер*, поддерживающий интерфейс, к которому подключаются периферийные устройства. Каналы или контроллеры можно рассматривать как специализированные процессоры, выполняющие свои собственные программы, реализующие управление операциями ввода-вывода на промежуточном (логическом) уровне.

Еще одним уровнем управления вводом-выводом служит *контроллер периферийного устройства*. В его задачи входит преобразование управляющих сигналов, получаемых по шинам интерфейса, в управляющие сигналы самого устройства; преобразование получаемой или передаваемой информации в/из форматов (и физических форм), используемых в самом устройстве, а также формирование сигналов оповещения о состоянии устройства, передаваемых на линии интерфейса. Эту схему управления операциями ввода-вывода можно представить в общем виде так, как показано на рис. 4.5.

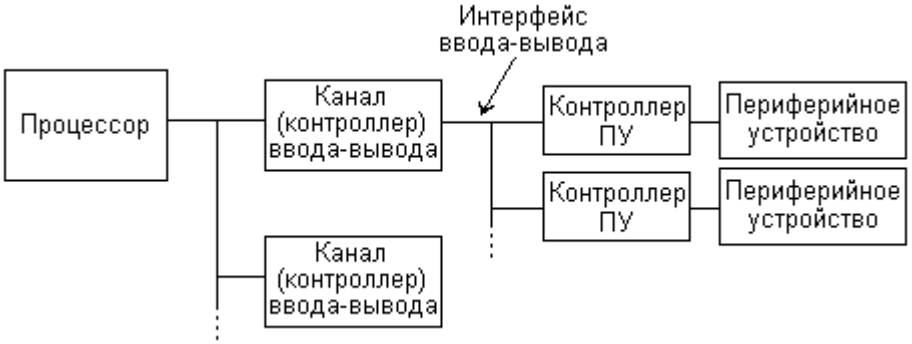


Рис. 4.5. Иерархия управления операциями ввода-вывода

Режимы управления вводом-выводом

Первым простейшим режимом управления вводом-выводом был режим *программно-управляемого ввода-вывода*. В этом режиме процессор непосредственно управлял опросом состояния и передачей данных между оперативной памятью и периферийным устройством, выдавая соответствующие команды. Общая блок схема такого управления показана на рис. 4.6, где *СчПД* – счетчик количества переданных данных.

Рис. 4.6. Программно-управляемый ввод-вывод

Такой режим прост в реализации и не требует дополнительных средств. Но синхронизация и собственно передача данных возлагаются на процессор. Это приводит к высокой его загрузке во время выполнения таких операций, вплоть до полной остановки исполнения других программ. Высокой скорости ввода-вывода в этом режиме при контроле состояния периферийного устройства достичь нельзя. Некоторое исключение составлял режим PIO (для старых компакт дисков), в котором процессор только передавал данные по командам “ввод/вывод строки”. Это обеспечивало скорость передачи до 16 Мбайт/с, но в синхронизации передачи участвовали аппаратные средства контроллера дискового интерфейса.

В режиме программно-управляемого ввода вывода иерархия, показанная на рис. 4.5, может не включать в себя уровень канала ввода-вывода, а также сводить к минимуму роль контроллера периферийного устройства.

С появлением средств прерывания стала возможной модификация программно-управляемого ввода-вывода, при которой процессор не опрашивал командами состояние периферийного устройства, а напротив, устройство выставляло сигнал прерывания при готовности к передаче. Такой режим иногда называют *программно-управляемый ввод-вывод с прерываниями*. Он позволяет получить более высокую скорость обмена и разгрузить процессор, но большое количество прерываний не позволяет значительно улучшить эти показатели.

Более высокую скорость обмена позволяет получить режим, в котором процессор используется только на стадии запуска и завершения операции ввода-вывода и освобождается от непосредственного управления ею. Такой режим предполагает непосредственную передачу информации между периферийным устройством и оперативной памятью, что и обусловило его название – *прямой доступ к памяти* (ПДП, или *DMA – Direct Memory Access*).

Непосредственное управление режимом прямого доступа к памяти осуществляется каналом (процессором) ввода-вывода или контроллером прямого доступа (используются и варианты управления шинами интерфейса контроллером периферийного устройства). На начальном этапе операции процессор посылает в канал или контроллер: *начальный адрес* области оперативной памяти, с которой будет производиться обмен (ввод или вывод); *значение счетчика*, равное количеству байтов, которое требуется передать; дополнительные *параметры операции* (в числе которых и направление передачи – ввод или вывод).

После этого процессор переходит к другим задачам, а управление операцией до ее завершения продолжает канал или контроллер ПДП.

Общий порядок циклов обмена с прямым доступом примерно одинаков и сводится к следующим шагам.

Получив от устройства, управление обменом данными которого с памятью осуществляет контроллер, информацию о готовности устройства принять или передать данные, контроллер запрашивает управление шиной данных памяти. Причем приоритет контроллера выше приоритета процессора.

Получив управление шиной памяти, контроллер выставляет на шину адрес памяти, по которому следует прочитать или записать данные. Далее при вводе данных он получает данные от устройства, выставляет их на шину данных памяти и запускает цикл записи в память. При выводе данных контроллер запускает цикл чтения памяти по установленному им адресу, по окончании которого считывает данные с шины данных памяти и передает их в устройство, осуществляющее обмен.

Если устройство передает данные с высокой скоростью, то контроллер запрашивает управление шиной сразу на несколько циклов, организуя передачу целого пакета данных. (Работу контроллера ПДП могут также называть захватом цикла и управлением шиной – bus mastering соответственно; иногда о ПДП говорят как о частном случае bus mastering.)

Скорость передачи данных по каналу ПДП (DMA) не слишком высокая. Первоначально она составляла 2 Мб/с, специальные модификации в пакетных режимах позволили поднять ее до 100 Мб/с и выше (Ultra DMA).

Позднее режим прямого доступа перерос в более общие способы управления интерфейсными шинами и связью между устройствами ЭВМ.

Специализация интерфейсов

Внешние устройства разного вида значительно различаются по скорости передачи данных и по характеру режимов передачи (равномерный, пакетный). Подключать к одинаковому интерфейсу низко- и высокоскоростные устройства

экономически нецелесообразно. Существовали также специализированные интерфейсы, используемые в ряде применений, отказ от которых неоправдан.

Поэтому развитие интерфейсов привело к сосуществованию, как минимум, двух уровней интерфейсов: системного и периферийного. К этим уровням можно добавить и внутренние интерфейсы (шины), связывающие процессор с системными контроллерами, а также сами эти контроллеры между собой.

Первый уровень включает в себя интерфейсы рассмотренного выше вида. К ним можно отнести единый интерфейс ЕС ЭВМ, общую шину СМ ЭВМ, AT-шину ранних ПЭВМ, интерфейсы ISA, PCI (с ее модификациями), AGP, а также с некоторыми оговорками IDE (ATA), LPC, широко распространенные в современных ЭВМ. Эти интерфейсы используются для внутреннего или стационарного подключения контроллеров типовых периферийных устройств, а также некоторых внутрисистемных устройств.

Интерфейсы второго из названных уровней используются для внешнего коммутируемого подключения периферийных устройств. К ним относятся шины USB, IEEE-1384 (FireWire), COM, LPT и Game порты. К данному уровню можно отнести и интерфейсы внешний SATA и SCSI, а также интерфейсы для подключения мониторов, клавиатуры и мыши. Эти интерфейсы реализуются специальными контроллерами, подключаемыми к системному интерфейсу, т. е. с помощью еще одного уровня управления операциями ввода-вывода.

В последние годы получил распространение и несколько иной принцип построения интерфейсов: не набор шин, к которым подключаются параллельно различные устройства (точнее, их контроллеры), а попарные соединения (точка-точка) устройств с мостами и коммутаторами или даже друг с другом.

Так был сделан следующий шаг в развитии архитектуры средств ввода-вывода – теперь от универсальных интерфейсов к специализированным.

4.2.2. Периферийные устройства ЭВМ

К периферийным устройствам ЭВМ относится очень широкий спектр оборудования, используемого для обмена информацией с ЭВМ. В зависимости от функционального назначения их можно разделить на большие группы устройств, используемых для хранения информации, для ввода-вывода информации и для связи с различными объектами и другими системами. К периферийным устройствам иногда относят и модемы, используемые для передачи информации по различным каналам.

Устройства ввода

Устройства ввода принято подразделять на ручные и автоматические.

К ручным устройствам ввода относят клавиатуры, манипуляторы (мыши, трэкболы, сенсорные панели, джойстики, руль, педали), планшеты, сенсорные дисплеи (это одновременно и ввод, и вывод) и т. п. К автоматическим устройствам ввода относятся различные устройства для восприятия в основном не цифровых объектов.

Клавиатура может встраиваться в специальные пульты управления, непосредственно в саму ЭВМ либо представлять собой отдельную панель, как кла-

виатура ПЭВМ. Варианты клавиатур различаются набором клавиш, расположением и размером основных управляющих клавиш. Также имеют место отличия в реализации контактных групп клавиш, типе связи (проводная, беспроводная) и соединительных разъемах [3].

Семейство манипуляторов-указателей, включающее в себя ряд разновидностей мышей и их заменители (трекболы и сенсорные панели, а также игровые джойстики и планшеты), используется для управления графическими указателями, имеющимися в различных приложениях, и для некоторых дополнительных функций, связанных с графическим вводом.

Описание особенностей и разновидностей ряда манипуляторов дано в [3].

Сенсорные панели, планшеты (иногда их называют дигитайзерами) и сенсорные дисплеи позволяют вводить координаты точек, в которых к ним прикасается палец или специальное перо. Обычно они используются в различных мобильных применениях или терминалах информационных систем.

Автоматические устройства ввода в настоящее время используются как средства для ввода изображений, речи, звуков и различных сигналов. Наиболее распространенными устройствами ввода статических изображений являются сканеры, а ввода динамических изображений – web-камеры. Аналого-цифровые преобразователи обычно используются для организации связи с различными объектами.

Устройства вывода

Основными устройствами вывода для ЭВМ являются устройства отображения текста и графики на бумаге (пленках, дисках и пр.) и экранах. Существуют также устройства на основе цифроаналоговых преобразователей и других схем, используемые в управляющих системах для вывода сигналов в объекты управления. Цифроаналоговые преобразователи используются и в аудиотехнике, подключаемой к ЭВМ.

Устройствами вывода информации на бумагу или иные “твердые” носители являются принтеры и плоттеры (графопостроители).

Наиболее распространены струйные и лазерные принтеры. Струйные принтеры печатают, выбрасывая на бумагу микроскопические капли специальных чернил. По способу образования и выброса капель они бывают электростатические, пьезоэлектрические и пузырьковые. Их различают и по обеспечиваемому разрешению, возможности цветной печати, количеству используемых цветов чернил, скорости печати, размеру и типу бумаги.

Лазерные принтеры (а также светодиодные) действуют по принципу, аналогичному используемому в копировальных аппаратах. В них изображение проецируется на покрытый селеном барабан, поверхность которого несет электрический заряд. Засвеченные участки разряжаются лазерным лучом (подсветкой светодиодами), а к заряженным притягивается красящий порошок, который переносится на бумагу и фиксируется подогревом.

Лазерные принтеры различаются обеспечиваемым разрешением, скоростью печати, форматом бумаги и конструктивными параметрами – способами

подачи бумаги, размерами и расположением лотков для нее, устройством механизмов протяжки, подогрева, фокусировки и т.д.

Существуют также матричные и сублимационные принтеры. Информация об этих и названных выше принтерах приведена в [3].

Плоттеры используются для отображения графических материалов большого формата (до A0) при конструкторских, архитектурных, оформительских работах. Плоттеры различаются принципом отображения (печать, рисование, вырезание), способом перемещения бумаги, форматом, разрешением и производительностью.

Кроме упомянутых устройств, для вывода информации могут использоваться проекторы, цифроаналоговые преобразователи и некоторые другие устройства.

Вопросы для самопроверки по теме 4.2

1. Каковы основные требования к системе ввода-вывода информации ЭВМ?
2. Что такое интерфейс ввода-вывода?
3. Чем различаются основные режимы ввода-вывода?
4. Что такое прямой доступ к памяти?
5. Каковы основные функции каналов ввода-вывода?
6. Для чего нужен контроллер периферийного устройства?
7. Что такое мастер шины?
8. Где используются аналого-цифровые и цифроаналоговые преобразователи?
9. Назовите основные устройства ввода

4.3. Архитектура ЭВМ и вычислительных систем

При изучении данной темы Вы должны познакомиться с основными особенностями архитектуры ЭВМ различных классов и вычислительных систем в целом.

Для проверки изучения материала темы Вам предстоит ответить на вопросы для самопроверки.

Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к учебнику [1] или к материалам файла Lect3.doc учебного сайта либо раздела сайта ord.com.ru/files/org_evm.

4.3.1. Понятие архитектуры ЭВМ и вычислительных систем

Повышение производительности ЭВМ может идти двумя основными путями: повышением быстродействия элементов ЭВМ (рабочих частот) и совершенствованием их структурной организации.

Для второго направления было установлено, что последовательный характер выполнения отдельных шагов программы и отдельных этапов выполнения команд является существенным ограничением для повышения производительности ЭВМ. Хотя последовательный характер функционирования ЭВМ во многом определяется последовательной природой самого алгоритма, пришлось искать пути преодоления этого ограничения.

Это привело к реализации параллелизма в работе ЭВМ на двух главных уровнях:

- на уровне исполнения команд, т. е. внутри самого процессора;
- на уровне организации совместной работы процессоров или ЭВМ.

С последним связано появление многопроцессорных и многомашинных вычислительных систем. При этом зависимость между аппаратными затратами и производительностью систем с параллельными вычислениями оказывается не линейной, а определяется соотношением в решаемых задачах участков, допускающих параллельное или только последовательное исполнение. Это нашло отражение в законах Амдаля и Густавсона [1].

Существует много определений понятия архитектуры ЭВМ и вычислительных систем, значительно отличающихся друг от друга. Поэтому здесь не вводится специальное определение этого термина. Важно лишь то, что архитектура – это некоторое абстрактное представление о функциональном устройстве вычислительной системы, составе и связях ее компонент.

Данное представление рассматривают на нескольких уровнях. Так, говорят об архитектуре вычислительной системы в целом, архитектуре ЭВМ, архитектуре ее программного обеспечения, архитектуре системы команд, архитектуре процессора (микроархитектуре в терминах Intel), архитектуре ввода-вывода и некоторых других уровнях. В соответствии с задачами настоящего учебного курса в нем рассматриваются в основном вопросы архитектур вычислительных систем, ЭВМ и процессоров.

4.3.2. Архитектура вычислительных систем

К вычислительным системам (ВС) относится широкий спектр вычислительных средств, различающихся своим назначением, составом, структурой, характером функционирования и другими особенностями. ВС обычно разделяют по основным признакам на следующие классы.

По *составу* различают однопроцессорные (одно- и многоядерные), многопроцессорные и многомашинные ВС. В рамках многопроцессорных и многомашинных систем рассматривают также однородные и разнородные их разновидности в зависимости от использования в системе одинаковых или различных процессоров либо ЭВМ.

По *функциональной ориентации* (уровню параллелизма):

- ВС для решения независимых задач,
- ВС для задач, состоящих из параллельных процессов,
- ВС для задач с одинаковой параллельной обработкой массивов данных (векторов, матриц).

По *форме параллелизма* (классификация Флинна):

- с одиночным потоком команд и одиночным потоком данных (ОКОД или SISD – Single Instruction – Single Data stream);
- с одиночным потоком команд и множественным потоком данных (ОКМД или SIMD – Single Instruction – Multiple Data stream);
- с множественным потоком команд и одиночным потоком данных (МКОД или MISD – Multiple Instruction – Single Data stream);

- с множественным потоком команд и множественным потоком данных (МКМД или MIMD – Multiple Instruction – Multiple Data stream).

В основу этого разделения положен характер вычислений, производимых системой.

По *виду связей* ВС различаются в значительной степени, причем среди них можно особо выделить виды связей, соединяющих процессоры:

- способ связи: непосредственный, через память (общую), через магистраль (шину), через коммутаторы, через каналы в/в;
- характер связи: статические, динамические;
- топология связей: линейные (кольцо), матричные, древовидные, полные, n-мерные (куб, гиперкуб и пр.), систолические, коммутируемые.

По *типу используемых процессоров* различают ВС обычные, конвейерные, векторные, ассоциативные, поточные (управляемые потоком данных), транспьютерные.

По *пространственному расположению* выделяют сосредоточенные и распределенные ВС. В последнем случае чаще говорят о вычислительных сетях.

Ключевыми вопросами при организации вычислительных систем, помимо вопросов обеспечения надежности их функционирования, являются:

- обеспечение быстрого доступа ко всем блокам памяти системы и поддержание согласованности информации в различных узлах системы
- обеспечение быстрой связи между вычислительными блоками
- эффективность операционных систем и программного обеспечения при разделении задач по вычислительным блокам системы и их решении.

Организация доступа к памяти в ВС

Поскольку обрабатывающие блоки в ВС в общем случае выполняют некоторым образом связанную обработку данных, то основной задачей обычно является организация доступа всех таких блоков к информации, имеющейся в системе. Иначе говоря, необходимо обеспечить возможность различным процессорам обращаться к различным запоминающим устройствам системы. Эта задача решается принципиально различными способами в многопроцессорных и многомашинных ВС.

В *многопроцессорных* ВС обычно организуется непосредственный доступ ко всем блокам оперативной памяти, т. е. в них организовано совместное использование памяти. В таких системах имеется несколько основных различных архитектур:

- с однородным доступом к памяти (UMA – Uniform Memory Architecture);
- с неоднородным доступом к памяти (NUMA – NonUniform Memory Architecture);
- с доступом только к кэш-памяти (COMA – Cache Only Memory Access).

В первом случае время доступа к различным модулям для процессоров памяти одинаковое. Если доступ к какому-то модулю более быстрый, то он

специально замедляется, что важно для программирования. В системе NUMA доступ к модулям памяти, более близким к процессору (или его собственным), выполняется существенно быстрее. В системе COMA также имеет место неоднородный доступ.

Организация связи процессоров с модулями памяти выполняется посредством общих шин (например, распространенная схема SMP – симметричный мультипроцессор), с помощью координатных коммутаторов либо с помощью многоступенчатых сетей.

В первом случае связи простые и быстрые, но при увеличении количества процессоров возрастает число конфликтов на шинах и скорости обмена по ним падают. Во втором случае скорости можно получить высокими и для большого числа процессоров, но тогда этот вариант оказывается очень дорогим.

Третий случай дает промежуточные значения, поэтому часто находит применение. Однако в нем возникают вопросы, связанные с выбором топологии таких сетей, алгоритмов маршрутизации передач по ним, а также с используемыми методами коммутации (каналов с промежуточным хранением, или буферизацией, с приостановкой передач). Приходится выбирать и тип связи в канале – последовательный или параллельный.

Наиболее серьезной проблемой при совместном использовании памяти является проблема согласованности информации в различных блоках памяти и кэш-памяти. В перечисленных выше архитектурах памяти используются, в целом, подобные приемы, связанные с отслеживанием (snooping) изменений в копиях одной и той же информации, хранящейся в различных ЗУ. Однако конкретные механизмы в различных реализациях могут заметно различаться.

Сами принципы согласования также могут быть различными [2]: строгая согласованность, согласованность по последовательности, процессорная согласованность, слабая согласованность и свободная согласованность. Широко распространенным протоколом согласования является протокол MESI (Modified–Exclusive–Shared–Invalid - изменено, монопольно, разделяемо, недействительно), отслеживающий записи в кэш любым процессором.

В *многомашинных* ВС непосредственный доступ ко всем блокам оперативной памяти всех ЭВМ, входящих в систему, в принципе, невозможен, поэтому в них обмен информацией с памятью, расположенной в другой ЭВМ, происходит с помощью передач по коммуникационным связям между ними. Эти связи должны быть высокоскоростными и могут иметь различную топологию и протоколы обмена.

Основные архитектуры многомашинных вычислительных систем – это процессоры с массовым параллелизмом, которые строятся на типовых процессорах, объединяемых высокоскоростными сетями передачи сообщений, либо кластеры рабочих станций (например, обычных ПЭВМ), связываемых посредством стандартных сетей.

Топологии соединений ВС

Существенный вклад в производительность, стоимость и надежность ВС вносит система соединений между компонентами (узлами) системы, для которой в соответствии с [1, 2] будет использоваться термин сеть межсоединений (СМС). Особенности коммутации узлов многопроцессорных и многомашинных вычислительных систем несколько различаются, но между ними имеется много общего.

Одним из определяющих моментов в сети межсоединений между узлами системы является топология ее связей. Такую сеть можно рассматривать как некоторый граф, вершинами которого (узлами сети) являются соединяемые устройства, а дугами – линии (каналы, шины) связи. Соединяемыми устройствами, в свою очередь, могут быть процессоры, модули памяти (в т.ч. внешней), специальные переключатели – коммутаторы, а также собственно интерфейсы и каналы ввода-вывода.

Узлы такой СМС обмениваются между собой информацией. Во многих случаях обмен производится посредством сообщений. Передача сообщений, особенно между удаленными узлами, в большинстве случаев осуществляется пакетами, имеющими определенный формат.

Узлы ВС могут являться как источником, так и приемником сообщений. Кроме того, большинство из них (кроме памяти) могут выступать и в роли коммутаторов, передающих информацию с входа на выход.

При построении сети межсоединений ключевыми вопросами являются *топология* связей, способы *коммутации и связи* и *выбор маршрутов* для передач между узлами.

Для оценки топологии соединений, используют определенные характеристики. Поскольку сеть межсоединений может рассматриваться как граф, то, помимо общих значений (количества узлов и связей), ее оценивают следующими показателями:

- *диаметр сети* – количество связей, соединяющих два наиболее удаленных (по связям) друг от друга узла сети (чем меньше диаметр, тем больше быстродействие сети);

- *порядок (степень) узла*, или *коэффициент разветвления*, – количество каналов, подключенных к узлу (чем больше степень, тем больше может быть маршрутов и выше отказоустойчивость, но тем сложней коммутация в узле);

- *пропускная способность* сети – количество данных, которое может передаваться по сети в единицу времени.

Для определения пропускной способности более важным параметром считают *бисекционную пропускную способность*. Под ней понимают количество данных, которое можно передать между двумя равными по числу узлов непересекающимися частями сети.

Рассматриваются и другие характеристики сети межсоединений. К ним относят *задержки* прохождения сообщений через сеть, *размерность* сети, определяемую как количество направлений движения из узлов сети и др.

Топологии межсоединений могут быть *статическими* и *динамическими*.

В статических топологиях связи между узлами ВС жестко определены (или настроены заранее), а сами узлы такой сети, как правило, не являются коммутаторами. Путь между парой узлов либо единственный, либо может в случае необходимости в некоторых топологиях (обычно резервный путь) создаваться с помощью процессорных и интерфейсных модулей, выполняющих в таком варианте роль коммутаторов. Основными вариантами таких топологий являются [1]: шинная, звезда, линейная (или кольцо), древовидная, решетчатая (матричная), полносвязная, кубическая и гиперкубическая.

В сетях межсоединений с динамическими топологиями узлами являются коммутаторы, а устройства, производящие обмен сообщениями, подключаются к входам и выходам сети (коммутаторов). Коммутаторы часто позволяют установить различные пути передачи. А это создает основу для задания правил выбора маршрутов (путей) – алгоритмов маршрутизации.

Коммутаторы также могут иметь схемы входных или выходных буферов, позволяющие осуществлять коммутацию с промежуточным хранением, что бывает необходимо при невозможности выполнить требуемую передачу, так как этому мешает уже передаваемое сообщение. В последнем случае говорят о возникновении блокировки в сети.

Варианты топологий многоступенчатых сетей рассматриваются в [1].

4.3.3. Архитектура ЭВМ

Архитектуру ЭВМ можно рассматривать как частный случай архитектуры вычислительной системы, состоящей из одной (строго говоря, однопроцессорной и одноядерной) ЭВМ. Основные архитектуры ЭВМ можно рассмотреть на примере микроЭВМ, систем с общей шиной и больших ЭВМ.

ЭВМ Единой системы (точнее, их прототипы IBM System 360) и несколько семейств малых машин других производителей сыграли в развитии вычислительных машин очень большую роль. В 60-70-х годах XX столетия эти машины доминировали на рынке ЭВМ общего назначения.

ЕС ЭВМ представляли собой семейство ЭВМ различной производительности, совместимых программно и функционально. Младшие и старшие модели отличались производительностью, объемом оперативной памяти, количеством подключаемых устройств. В общем виде структура ЭВМ Единой системы представлена на рис. 4.7, где ОП – оперативная память, БСОП – блок связи с оперативной памятью, АЛУ – арифметико-логическое устройство, БЦУ – блок центрального управления, БУП – блок управления пультом, БСС – блок системной связи, МК – мультиплексный канал, СК – селекторные каналы, КУВВ – контроллеры УВВ.

В этой структуре процессор является центральным узлом, связывающим все остальные блоки и коммутирующим информационные потоки. Каналы, через которые с помощью контроллеров (в ЕС они назывались блоками управления) подключаются внешние устройства, имеют доступ к оперативной памяти, используя для этого блок связи с ОП, размещенный в процессоре.

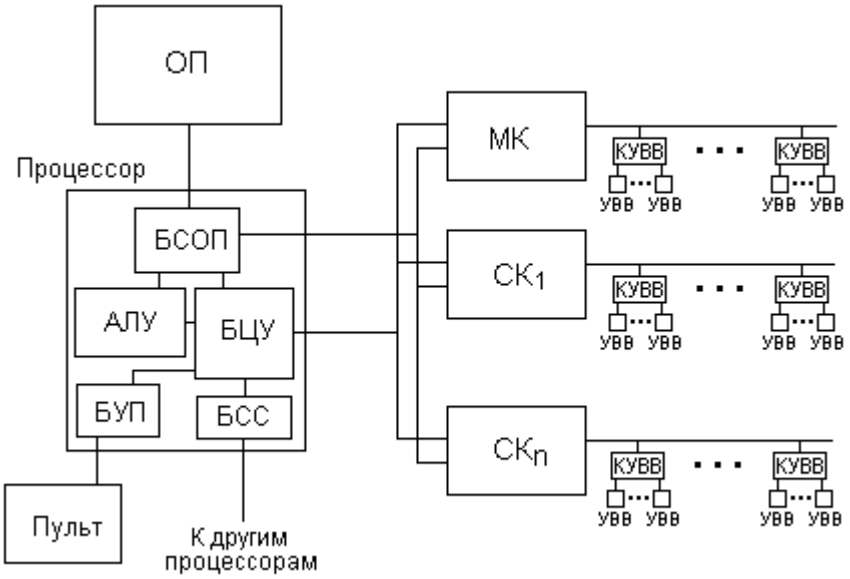


Рис 4.7. Общая структура ЕС ЭВМ

В противовес такой структуре в малых машинах “центром” является общая шина (UNIBUS), к которой подключаются все блоки ЭВМ. Такая архитектура была предложена (и не сразу принята) разработчиками знаменитой фирмы DEC, а ее ЭВМ PDP-11 продавалась в общей сложности около 25 лет (с 1970 по 1996 г.). Позднее эта архитектура была реализована рядом других фирм, а также в отечественных СМ ЭВМ (системе малых ЭВМ).

Структура ЭВМ с общей шиной показана на рис. 4.8. Все блоки и устройства подключаются к общей шине, по которой и выполняются все необходимые передачи данных, адресов и управляющих сигналов. Такая структура, обладавшая регулярностью и простотой, широко использовалась во многих ЭВМ, в том числе в микроЭВМ и первых персональных ЭВМ.

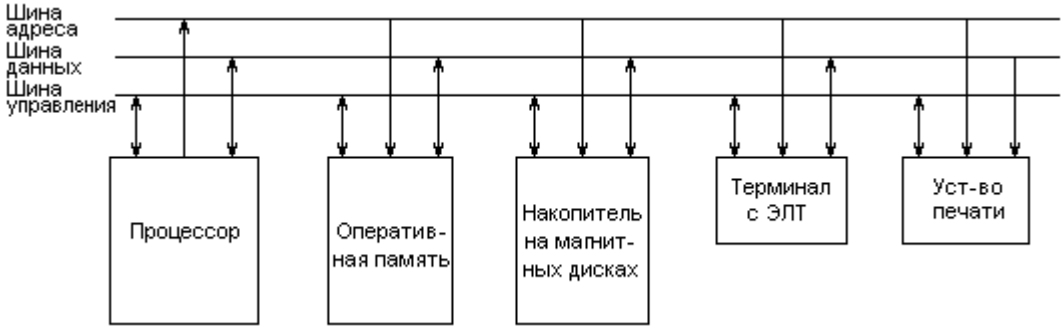


Рис 4.8. Структура ЭВМ с общей шиной

Соединение процессора с оперативной памятью через общую шину, разделяемую всеми устройствами, существенно ограничивает производительность системы. Поэтому в последующей модификации PDP-11 – системе VAX-11 была введена отдельная шина для связи ОП с процессором.

Позднее похожий прием использовали и в процессорах Intel, в которые, начиная с Pentium Pro, ввели шину связи процессора с кэшем второго уровня.

Создание микропроцессоров и развитие интегральных технологий обусловило и появление в конце 70-х – начале 80-х годов XX века класса микро-

ЭВМ. Основным их отличием от существовавших в то время ЭВМ была реализация их на базе микропроцессоров и больших интегральных схем. (В такой трактовке все современные ЭВМ можно было бы называть микроЭВМ.) Технологии того периода позволяли создавать лишь несложные по структуре ЭВМ, обладающие небольшими функциональными возможностями. Их архитектура во многом копировала малые ЭВМ, а разработчиком одной из первых и широко известной микроЭВМ – LSI-11 (отечественный аналог – “Электроника-60”) была уже упоминавшаяся фирма DEC. Эти ЭВМ имели общую шину, к которой подключались все основные устройства. Поддержка общей шины могла организовываться самим процессором. Общий вид архитектуры микроЭВМ показан на рис. 4.9. Такая структура практически совпадает со структурой первых персональных ЭВМ.

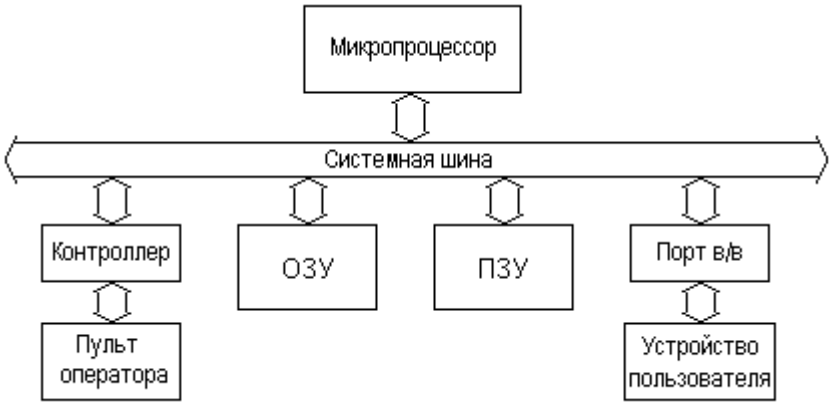


Рис 4.9. Общая структура микроЭВМ

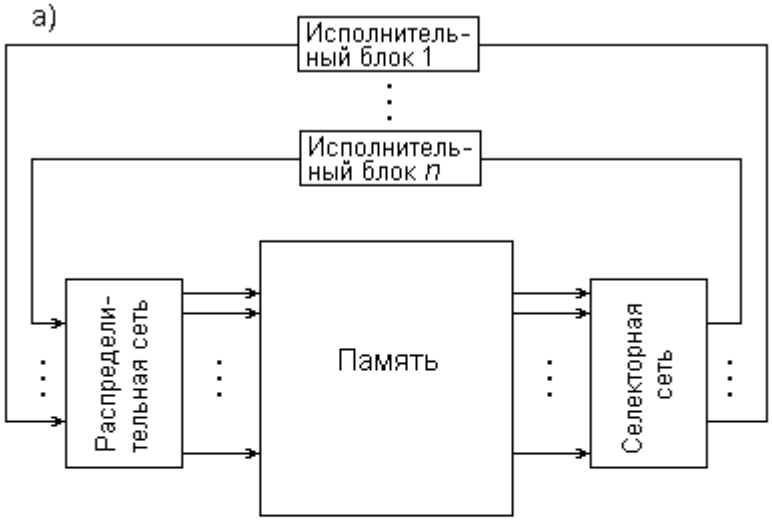
Совершенствование интегральных технологий привело к увеличению возможностей микроЭВМ, появлению персональных ЭВМ и переводу большинства больших ЭВМ и вычислительных систем на типовые (микро) процессоры, потерявшие постепенно в названии составляющую “микро”.

Помимо классических архитектур, развивались средства автоматического распараллеливания выполнения команд программы. Их суть сводилась к выполнению операций программы не строго в порядке, предписанном алгоритмом, а при наличии свободных исполнительных ресурсов и готовности данных или необходимости получения результата операции для выполнения дальнейших действий. Это дает три механизма управления запуском команд:

- традиционный (control flow driven) – выполнение последовательности команд в соответствии с программой;
- управление потоком данных (data flow driven) – выполнение команды при готовности всех ее операндов и наличии свободного операционного блока;
- рекурсивный вызов или вызов по запросу (recursive или demand driven) – постановка команды в очередь на выполнение, если ее результат нужен другой команде.

Архитектура ЭВМ, управляемой потоком данных, схематически может быть представлена так, как это показано на рис. 4.10, а. В памяти этой ЭВМ

хранятся команды программы, формат которых показан на рис. 4.10, б. Этот формат отличен от классических команд. Помимо кода операции, он содержит:



б)

Код операции	Адреса назначения		
Тип условия	Флаг условия	Флаг данных	Данные
Тип условия	Флаг условия	Флаг данных	Данные

Рис. 4.10. Упрощенное представление структуры ЭВМ, управляемой потоком данных (а), и формат ее команды (б)

- поле адресов назначения, по которым должен рассылаться результат выполнения команды;
- поля данных, куда записываются значения операндов команды;
- флаги данных, которые взводятся при готовности операндов;
- поля типов условий, определяющие порядок приема данных в соответствующее поле (безусловный, при значении **.t.** флага условия, при значении **.f.** флага условия, константа);
- поля значений флагов условий (не поступил, **.t.**, **.f.**).

При работе этой ЭВМ селекторная сеть просматривает содержимое памяти и находит команды, у которых готовы данные (взведены флаги данных). При наличии свободных операционных блоков такие данные и коды операций таких команд пересылаются в исполнительные блоки и выполняются. После этого распределительная сеть рассылает результат выполнения команды в поля данных по адресам, указанным в поле адресов назначения, с проверкой значений соответствующих флагов и типов условий и взводит соответствующие флаги данных.

Такая структура позволяет выполнять параллельно команды программы без нарушения логики алгоритма, автоматически выявляя возможность такого выполнения. Известны несколько разновидностей таких ЭВМ [1].

В ЭВМ с управлением потоком данных готовые к выполнению команды исполняются при наличии свободных исполнительных блоков вне зависимости

от того, потребуется ли это при выполнении алгоритма. Стремление исключить выполнение ненужных команд положено в основу организации рекурсивных ЭВМ с выполнением команд по запросу их результата. (Такие ЭВМ иногда называют редукционными.) Анализ программ в этом случае происходит в обратном порядке – от конечных операций, поэтому для таких ЭВМ применяют специальные языка программирования, называемые функциональными.

Существуют также некоторые другие специальные архитектуры ЭВМ, различающихся своим функциональным назначением, например машины баз данных, ЭВМ с систолическими структурами, ЭВМ на основе транспьютеров.

4.3.4. Архитектура процессоров (микроархитектура)

Основные пути повышения производительности процессоров базируются на увеличении степени параллелизма в процессе выполнения команд. Это достигается за счет:

- конвейеризации выполнения различных этапов команд программы;
- использования нескольких исполнительных блоков для выполнения операции и формирования адресов;
- оптимизации порядка обращений к памяти.

Принципы конвейерной обработки

Конвейерная обработка команд используется для повышения производительности ЭВМ. Принцип такой обработки, авторство которого относят академику С.А. Лебедеву, заключается в совмещении во времени выполнения различных этапов последовательных команд программы.

В выполнении каждой команды можно выделить несколько этапов. Сделать это можно различными способами, например выделив четыре этапа: выборка команды из памяти, декодирование команды и выборка операндов из памяти, выполнение заданной в команде операции и запись результата в память. Выполнение таких команд в конвейере показано на рис. 4.11.

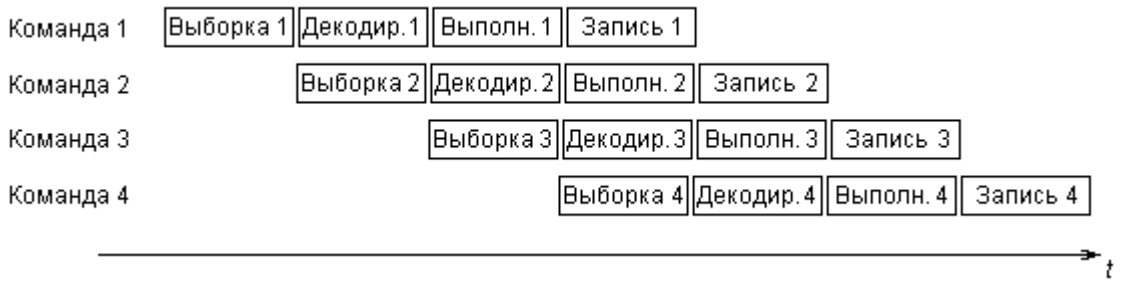


Рис 4.11. Конвейерное выполнение команд программы в ЭВМ

Выполнение действий на каждой ступени конвейера должно по возможности занимать одинаковое время. И, чем больше этапов выделено в исполнении команды, тем короче и проще могут быть эти этапы и тем больше команд одновременно находится в обработке. А это дает и повышение производительности, и возможность поднять рабочую частоту ядра процессора.

Но увеличение количества этапов (длины конвейера) не всегда приводит к повышению производительности. Выполнение команд по указанной схеме

лучше всего производится при одинаковом времени осуществления отдельных этапов. Однако это далеко не всегда имеет место по следующим причинам:

- обращение к памяти, как правило, занимает значительно больше времени, чем собственно выполнение операций;

-в качестве данных для следующей команды могут использоваться результаты выполнения предыдущей команды;

- выполнение некоторых операций, например деления, требует больше времени, чем для большинства других операций;

- адрес следующей команды может зависеть от результатов выполнения предшествующей команды (условные переходы).

Для учета влияния этих факторов используются следующие методы.

Команды и данные хранятся в кэш-памяти, к которой и обращается процессор. Обращения к оперативной памяти производятся только при отсутствии информации в кэш-памяти, и задержки в работе конвейера из-за большого времени обращения к памяти имеют место только в таких случаях.

Задержки из-за зависимости (конфликтов) по данным, возникающие при использовании командой результата выполнения предшествующей команды, исключаются (при операциях, выполняемых за один такт) за счет непосредственной передачи результата на обработку этой команде одновременно с началом записи результата (в регистр).

Задержки в запуске выборки команды из-за зависимости адреса следующей команды (направления перехода) от результата исполнения предыдущей можно уменьшить посредством начала выборки команды наиболее вероятной ветви перехода до завершения выполнения предыдущей. Однако задержка будет исключена, только если переход пойдет именно по выбранному направлению. В противном случае все выполненные, начиная с выборки команды, следующей за переходом, действия придется игнорировать и начать выполнение нужной ветви.

Для сокращения задержек, обусловленных выборкой команды из точки перехода, помимо различных вариантов буферизации [1] в случае условных переходов используется один из следующих вариантов:

- игнорирование факта перехода и продвижение по программе как в его отсутствие;

- формирование обеих ветвей перехода до того момента, когда значение условия перехода станет вычисленным;

- выбор наиболее вероятного направления перехода.

Первый путь прост, но неэффективен. Второй – малоприменим на участках программ со следующими друг за другом переходами. Поэтому часто используется третий путь, в котором выбор направления перехода осуществляет так называемый механизм *предсказания переходов*.

Алгоритмы предсказания переходов делятся на две большие группы: статические и динамические [1]. Хорошие алгоритмы (обычно динамические) могут правильно предсказывать до 95-97 % переходов.

Суперскалярная архитектура

Развитие конвейерной архитектуры связано с использованием нескольких исполнительных блоков (АЛУ), позволяющих выполнять одновременно несколько операций. Такие системы получили название *суперскалярной конвейерной архитектуры*. Для лучшей загрузки исполнительных блоков в них применяется механизм изменения порядка выполнения команд.

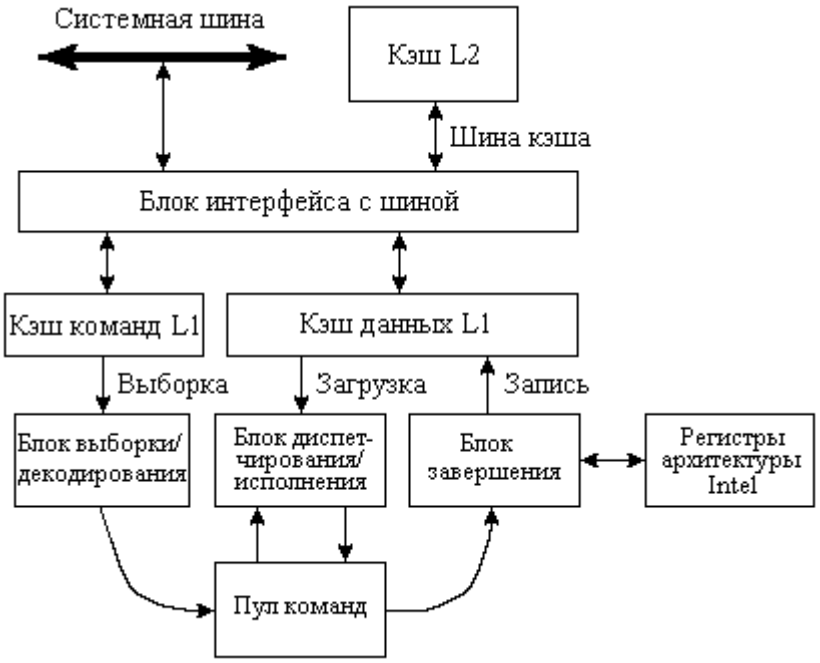


Рис 4.12. Функциональные блоки процессоров Intel семейства P6

Суперскалярная конвейерная архитектура показана на рис. 4.12 на примере процессоров семейства P6 фирмы Intel, которые могли в среднем декодировать, обработать и завершить выполнение трех команд за один рабочий цикл. В них использовался 12-ступенчатый конвейер с *неупорядоченным* (out-of order) *исполнением* команд, оптимизирующим обращения к памяти и использование операционных блоков. Эта архитектура появилась уже в первых процессорах Pentium.

Конвейер включает 4 блока: блок выборки/декодирования, блок диспетчирования/выполнения, блок завершения и пул команд. Для обеспечения непрерывной подачи команд и данных в конвейер обработки команд, архитектура процессоров семейства P6 включает два уровня кэша. Кэш первого уровня (L1) состоит из кэша команд и кэша данных.

Механизм неупорядоченного выполнения команд, названный “динамическим выполнением”, использует *глубокое предсказание переходов, анализ динамических потоков данных* и *предварительное выполнение команд*.

Алгоритм предсказания переходов определяет направления переходов при многоуровневых ветвлениях, вызовах процедур и возвратах из них.

Анализ динамических потоков данных в реальном масштабе времени определяет зависимости между данными и регистрами для выявления возможностей неупорядоченного выполнения команд. Блок диспетчирования / исполнения процессоров семейства P6 может одновременно просматривать много

команд и выполнять их в порядке, оптимизирующем использование исполнительных блоков процессора, поддерживая целостность данных.

Предварительное исполнение означает возможность выполнять команды прежде, чем на них укажет счетчик команд, но в конечном счете формировать результаты в соответствии с исходным порядком потока команд. Блок диспетчирования/исполнения анализирует возможность выполнения всех команд из пула команд и сохраняет результаты во временных регистрах. Затем блок завершения просматривает пул команд в поисках команд, не имеющих более зависимостей по данным с другими командами или необработанных предсказаний переходов, записывает результаты выполнения таких команд в память и/или в регистры процессора в том порядке, в котором они должны выполняться по программе, и удаляет их из пула команд.

В процессорах Pentium 4 конвейер получил 20 ступеней, что позволило повысить частоту работы процессора. Кэш команд, преобразованный в кэш трассы, мог хранить до 126 декодированных команд. Был улучшен алгоритм предсказания переходов, что необходимо для длинного конвейера, и буфер предсказания переходов мог обрабатывать до 12 вложенных разветвлений.

Позднее в архитектуре Core отказались от более длинного конвейера.

Гиперпоточная архитектура

Гиперпоточная архитектура Intel позволяет одному физическому процессору одновременно исполнять два или большее количество раздельных потоков команд (threads).

Процессор с гиперпоточной технологией состоит из двух или более логических процессоров, каждый из которых имеет свой собственный набор регистров обычной архитектуры, называемый блоком состояния процессора (в оригинале архитектурным состоянием – AS).

Этот блок содержит состояние регистров (общего назначения, управляющих, прерывания, служебных). Каждый БСП плюс единственное физическое ядро (блоки предсказания ветвлений, АЛУ, блок операций с плавающей запятой, SIMD-блоки и пр.) представляет собой отдельный логический процессор (ЛП). У каждого ЛП есть свой собственный контроллер прерываний и набор регистров. Для корректного использования регистров логическими процессорами существует специальная таблица – RAT (Register Alias Table – таблица имен регистров), по которой можно установить соответствие между регистрами общего назначения физического CPU. Эта таблица у каждого ЛП своя. В результате на одном и том же ядре могут выполняться столько независимых фрагментов кода, сколько имеется логических процессоров.

Программное обеспечение воспринимает такой процессор как несколько независимых процессоров (по числу БСП), что позволяет программному обеспечению, разработанному для исполнения в многопроцессорных системах, без изменений выполняться на одном физическом процессоре с гиперпоточной архитектурой, распределяя потоки между его логическими процессорами.

Архитектура ЭВМ со сверхбольшой длиной командного слова

Еще один подход к реализации параллелизма уровня команд предполагает анализ программ на выявление возможностей параллельного выполнения их команд до выполнения программы уже на этапе компиляции. В одном из основных вариантов своей реализации такой подход связывался со сверхдлинными командными словами. Компилятор по возможности объединяет в одну сверхдлинную команду (*Very Large Instruction Word - VLIW*) несколько простых команд, которые могут быть выполнены параллельно различными блоками.

Сверхдлинные команды первых компьютеров имели длину 256 битов и содержали несколько полей, задающих от 6 до 8 операций для разных функциональных блоков. В одной из ЭВМ устройства команды можно было комбинировать так, что получались команды длиной до 1024 битов.

К VLIW-архитектурам относились и отечественные разработки, например ЭВМ М10 и М13 М.А. Карцева; а также «Эльбрус-3» Б.А. Бабаяна.

Intel заложила некоторые принципы VLIW в процессоры Itanium. Но значительного повышения производительности в системах на базе этих процессоров по отношению к суперскалярным архитектурам можно добиться лишь путем упрощения аппаратного обеспечения при переносе ряда его функций в компилятор. Впоследствии Intel совместно с Hewlett-Pakkard развили эту архитектуру, назвав ее EPIC (Explicitly Parallel Instruction Computing). Коммерческой реализацией EPIC стала архитектура IA-64.

Процессоры с RISC-архитектурой

В классических ЭВМ развитие системы команд шло по пути усложнения в стремлении приблизить внутренний язык (команд) машины к языкам программирования высокого уровня. Но доля сложных команд в программах не столь высока, чтобы полностью оправдывать усложнение аппаратуры для их поддержки. Кроме того, операторы языков высокого уровня часто реализуются в исполнительном модуле подпрограммами. Время, затрачиваемое на вызов подпрограмм, передачу им параметров, возврат результатов и возврат из подпрограмм, может достигать до 40 % от общего времени вычислений.

Эти и некоторые другие результаты анализа программ привели к созданию процессоров с упрощенным набором команд – RISC (Reduced Instruction Set Computers – компьютеры с упрощенным, или сокращенным, набором команд). Основные концепции RISC-архитектуры следующие:

- в набор команд процессора должны входить простые наиболее часто используемые команды одинаковой длины, время выполнения которых примерно одинаково;

- обращение к памяти, время которого достаточно велико, выполняется только командами чтение и запись, а остальные имеют формат регистр-регистр;

- простые команды генерируют много обращений к переменным, поэтому количество регистров, доступных командам, должно быть большим (более 32).

Таким образом, RISC-процессоры имеют простую и небольшую систему команд (около 100 команд одинаковой длины, трех-четырех форматов и малое

количество способов адресации), быстрое устройство управления с жесткой логикой, значительное количество регистров общего назначения.

RISC-процессоры имеют более высокие рабочие частоты, меньше стоят и более надежны. Но простота системы команд приводит к более длинным программам. Большое количество регистров увеличивает время доступа к ним из-за усложнения схемы декодирования. Короткие команды и простые схемы адресации усложняют доступ к памяти достаточной емкости.

Вопросы для самопроверки по теме 4.3

1. Каковы достоинства и недостатки классификации вычислительных систем Флинна?
2. Перечислите способы параллельной обработки данных.
3. Что дает применение распределенной и совместно используемой памяти?
4. Какой из протоколов слежения наиболее популярен и почему?
5. Что такое коммутатор сети межсоединений?
6. В чем состоят различия между конвейерными и векторно-конвейерными вычислительными системами?
7. Какие проблемы решает кластерная организация вычислительной системы?
8. В чем заключается управление от потока данных?
9. Каковы основные особенности VLIW-архитектуры?
10. С чем связано появление RISC-процессоров?

4.4. Принципы построения аналоговых и гибридных ЭВМ

При изучении данной темы Вы должны познакомиться с основами устройства и принципами построения аналоговых и гибридных ЭВМ.

Для проверки изучения материала темы Вам предстоит ответить на вопросы для самопроверки.

Если Вы испытываете затруднения в ответе на какой-либо вопрос, обратитесь к материалам файла Lect5_ac.doc учебного сайта или раздела сайта ord.com.ru/files/org_evm.

4.4.1. Представление информации в АВМ

АВМ иначе называют моделирующими установками, или электронными моделями. Решение задач, или моделирование, на АВМ сводится к составлению и решению уравнения (или системы уравнений) для моделируемого процесса.

АВМ в основном строятся на электрических и электронных элементах постоянного тока. Переменные (функции) представляются в АВМ с помощью напряжений постоянного тока. Хотя для построения модели может использоваться любой другой физический процесс, описываемый теми же математическими зависимостями, что и моделируемый процесс.

4.4.2. Организация АВМ

В общем виде структурная схема АВМ показана на рис. 4.13. *Решающие блоки* АВМ являются ее основными блоками, выполняющими математические операции. *Измерительные приборы* используются для измерения вводимых величин и результатов решения. *Система питания* обеспечивает все необходи-

мые напряжения для решающих блоков и других устройств машины. *Система управления* обеспечивает управление остальными блоками машины.

Решающие блоки в зависимости от вида реализуемых математических операций подразделяются на линейные блоки и блоки нелинейности.

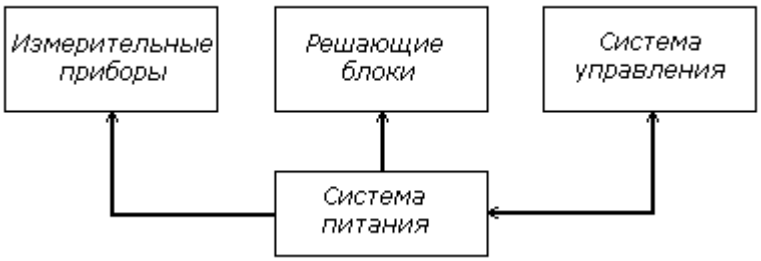


Рис 4.13. Структурная схема АВМ

Линейные блоки АВМ

Базовым линейным решающим блоком электронных АВМ является операционный усилитель. Он представляет собой усилитель постоянного тока с большим коэффициентом усиления, охваченный глубокой отрицательной обратной связью. С помощью различных комбинаций электрических емкостей и сопротивлений на входе усилителя и в цепи его обратной связи реализуются следующие математические операции: умножение на постоянный коэффициент, изменение знака (инвертирование в аналоговом смысле), алгебраическое сложение, интегрирование и дифференцирование (во времени).

Количество операционных усилителей в АВМ составляло от 15 до 200.

Реализация нелинейных зависимостей в АВМ

Реализовать нелинейные зависимости в АВМ можно: на основе типовых линейных блоков или специальных блоков нелинейностей.

Первый путь предполагает два основных подхода:

- решение определяющих дифференциальных уравнений;
- кусочно-линейную (или кусочно-постоянную) аппроксимацию воспроизводимых функций.

Определяющим дифференциальным уравнением для некоторой функции $f(t)$ называется такое дифференциальное уравнение, решением которого является сама эта функция. Например, для функции $e^{-\alpha t}$ таким уравнением будет $dy/dt = -\alpha y$, в чем легко убедиться подстановкой функции в уравнение.

При использовании кусочно-линейной аппроксимации реализуемая функция представляется суммой отрезков прямых. Каждый отрезок может быть воспроизведен одним-двумя линейными блоками. Этот способ применяется и для построения аппаратных блоков нелинейностей.

Аппаратные *блоки нелинейностей* могут строиться с использованием “точного” (непрерывного) способа либо кусочно-линейной (или кусочно-постоянной) аппроксимации.

При применении непрерывного способа блоки нелинейностей строятся на основе использования нелинейных характеристик каких-либо элементов, например, нелинейного участка вольт-амперной характеристики диода.

При кусочно-линейной (или кусочно-постоянной) аппроксимации блок нелинейностей включает в себя некоторое количество узлов, каждый из которых может быть настроен на реализацию одного линейного участка, представляющего функцию в некотором диапазоне значений аргумента.

Реализация операций умножения и деления в АВМ

В качестве основной операции в АВМ реализуют умножение, а деление выполняют с помощью множительного блока, как обратное к умножению.

По способу реализации операции различают устройства умножения прямого и косвенного действия. В первых из них умножение выполняется непосредственно по формуле $Z = k\,X*Y$, а в устройствах косвенного действия – в результате выполнения других математических действий, например,

$$Z = X*Y = \frac{1}{4}\,[(X + Y)^2 - (X - Y)^2] \quad \text{или} \quad Z = X*Y = a^{(lg_a X + lg_a Y)}.$$

Наиболее распространены устройства косвенного типа на основе элементов с квадратичными и логарифмическими характеристиками, так как они имеют приемлемую точность и быстродействие, а логарифмические и квадратичные функции реализуются проще других. Погрешность выполнения операции умножения в подобных блоках составляет порядка 0,5 %.

Реализация операции деления производится на основе множительного блока, включаемого в обратную связь операционного усилителя.

Вопросы для самопроверки по теме 4.4

1. Какой вид моделирования используется в АВМ?
2. Какой блок является основным решающим блоком АВМ? Какие операции он выполняет?
3. Как реализуются нелинейные операции в АВМ?
4. Что такое определяющее уравнение?
5. Какие операции может выполнять аналоговая ЭВМ и не может цифровая?

ЗАКЛЮЧЕНИЕ

Вы закончили изучение дисциплины “Организация ЭВМ и систем”.

В процессе изучения курса Вы познакомились с организацией и принципами построения современных ЭВМ и систем. Усвоенный материал должен послужить основой для изучения последующих дисциплин курса, определяя ключевые моменты, на которые надо в первую очередь обратить внимание в дисциплинах, связанных с системным программным обеспечением, организаций интерфейсов ЭВМ, сетями ЭВМ, схемотехникой, защитой информации и моделированием систем.

Кроме того, в связи с непрерывным и стремительным совершенствованием средств вычислительной и информационной техники изученные принципы должны позволить Вам самостоятельно осваивать все новые средства, устройства и системы, появляющиеся на рынке. Тем более, что при наличии Интернета получение

информации соответствующей степени детальности не является чем-то недостижимым.

Из рассмотренных законов развития информационно-вычислительной техники известно, что многие ее ключевые параметры изменяются в разы за полтора-два года. Однако, отслеживая основные параметрические изменения в ЭВМ, необходимо также помнить, что не менее важными являются и структурные их изменения, изменения в принципах организации.

Глоссарий

ATA – AT-attachment (подключение к шине AT) – интерфейс, использующийся для подключения жестких и оптических дисков к ЭВМ

DIMM – dual in line memory module (модуль памяти с двумя рядами краевых контактов) – один из вариантов конструктивного оформления модулей оперативной памяти, использующийся в основном для синхронной динамической оперативной памяти

DMA – direct memory access (прямой доступ к памяти) – см. *доступ к памяти, прямой*

ICH – input-output (иногда, integrated) control hub (хаб управления вводом-выводом) одна из микросхем системного набора, функционально эквивалентная *южному мосту*, также отличающаяся главным образом средствам связи между соединяемыми компонентами

IDE – integrated device electronics (интегрированная в устройстве электроника) – принцип построения и интерфейс связи с жесткими дисками, при котором большинство функций управления, определяющихся спецификой конкретного жесткого диска, возложено на контроллер, находящийся непосредственно на (или в) корпусе жесткого диска

MCH – memory control hub (хаб управления памятью) – одна из микросхем системного набора, функционально эквивалентная *северному мосту*, но отличающаяся, главным образом, средствами связи между соединяемыми компонентами

PCI – peripheral component interconnect (подключение периферийных компонент) шина расширения, использующаяся для подключения стандартным способом дополнительных устройств к ЭВМ

PCI-Express – развитие шины PCI, обеспечивающее более высокие скорости передачи данных; отличается использованием последовательной передачи данных и связыванием устройств только попарно (точка – точка)

RAID – redundant array of inexpensive disks (избыточный массив недорогих дисков) – технология построения памяти на жестких дисках, использующая несколько дисков для повышения производительности и надежности системы дисковой памяти

RAM – random access memory (память с произвольным доступом) – см. *память с произвольным доступом*; англоязычный термин, часто используемый как синоним оперативного ЗУ

SATA – Serial ATA (последовательный ATA) – интерфейс, использующийся для подключения жестких и оптических дисков к ЭВМ, пришедший на смену обычному ATA (который часто стали называть параллельным), допускающий более высокую скорость передачи и длину соединительного кабеля

SCSI – small computer system interface (интерфейс малых вычислительных систем) – высокоскоростной интерфейс для обмена информацией с несколькими устройствами; часто используется для подключения жестких дисков в серверах

SIMM – single in line memory module (модуль памяти с одним рядом краевых контактов) – один из вариантов конструктивного оформления модулей оперативной памяти, использовавшийся как для асинхронной, так и для синхронной динамической оперативной памяти

Адресации, способ – способ преобразования кода, записанного в адресном поле команды, в виртуальный или физический адрес памяти

АЛУ – см. *арифметико-логическое устройство*

Арифметико-логическое устройство – устройство, обычно часть процессора, непосредственно производящее выполнение операций по переработке информации

Банк памяти – часть запоминающего устройства, имеющая собственные схемы адресации и управления, допускающая возможность обслуживания обращений, адресованных к элементам памяти этой части независимо от остальных частей устройства

Время доступа – одна из основных характеристик запоминающих устройств, показывающая время, необходимое для извлечения информации из ЗУ или записи информации в него; определяется по-разному для различных типов запоминающих устройств

Вычислительная машина, аналоговая – вычислительная машина, использующая аналоговую (модельную) форму представления информации и схемную организацию вычислительного процесса

Вычислительная машина, цифровая – вычислительная машина, использующая алфавитную (дискретную) форму представления информации и программную организацию вычислительного процесса

Доступ, адресный – способ доступа к запоминающему устройству, при котором местоположение извлекаемой или записываемой в ЗУ информации определяется ее адресом (обычно некоторым положительным целым числом)

Доступ, ассоциативный – способ доступа к запоминающему устройству, при котором местоположение извлекаемой или записываемой в ЗУ информации определяется ее значением

Доступ к памяти, прямой – способ организации управления вводом-выводом, при котором внешнее устройство обменивается информацией с оперативной памятью без непосредственного участия процессора

ЗУ – см. *запоминающее устройство*

Запоминающее устройство (ЗУ) – устройство, реализующее функцию хранения информации

Запоминающее устройство, оперативное (ОЗУ) – запоминающее устройство, непосредственно доступное процессору, хранящее программы, выполняемые или готовые к исполнению, и их данные, а также основные программы операционной системы; в англоязычной литературе часто называется памятью с произвольным доступом – *RAM*

Запоминающее устройство, перепрограммируемое (ППЗУ) – запоминающее устройство, как правило, сохраняющее информацию при выключении питания, информация в которое может записываться многократно, причем время записи заметно превосходит время считывания

Запоминающее устройство, постоянное (ПЗУ) – запоминающее устройство, информация в которое заносится однократно (при изготовлении или пользователем) и впоследствии не изменяется, а также сохраняется при выключении питания

Интерфейс – совокупность средств, форматов, правил и протоколов логического и/или физического уровня, определяющих способ взаимодействия между устройствами, программами, функциями, пользователем и информационной системой и т. п.

Канал ввода-вывода – специализированный процессор, предназначенный для управления операциями ввода-вывода

Комплекс, вычислительный – совокупность ЭВМ, каналов связи, периферийного оборудования и других технических средств, а также программного обеспечения и обслуживающего персонала, обычно предназначенный для решения определенного класса задач

Конвейер команд – совокупность основных блоков процессора и памяти, рассматриваемых как последовательность узлов, обслуживающих выполнение команды процессора

Контроллер – универсальное или специализированное устройство управления

Коммутация (в *сети межсоединений*) – способ установления связей и управления передачей информационных пакетов между узлами вычислительной системы или сети

Кэш-память – быстродействующая память (одного или более уровней), располагающаяся между процессором и оперативной памятью; буферная память различных устройств, например жесткого диска

Маршрутизация – выбор пути передачи информации (при наличии нескольких путей) между узлами сети межсоединений вычислительной системы

Массового обслуживания, теория – раздел теории вероятностей, используемый для оценки производительности ЭВМ и систем

Межсоединений, сеть – совокупность средств соединения узлов вычислительной системы

Микропрограммное устройство управления – см. *устройство управления, микропрограммное*

Мост – связь или устройство, использующееся для соединения двух (или более) компонент ЭВМ, системы или сети

Мост, северный - компонент системного набора микросхем для построения ЭВМ, обеспечивающий взаимодействие между процессором, оперативной памятью, видеосистемой и южным мостом

Мост, южный – компонент системного набора микросхем для построения ЭВМ, обеспечивающий взаимодействие с жесткими дисками, средне- и низкоскоростными периферийными устройствами, поддержку шин расширения и включающий в себя ряд контроллеров, например контроллер прерываний, сетевой контроллер, контроллер прямого доступа и др.

фон Неймана, Джона принципы – основные положения, выдвинутые Дж. фон Нейманом, в отношении организации цифровых вычислительных машин

Неймановская архитектура – традиционная архитектура ЭВМ, включающая в себя основные устройства ЭВМ и отвечающая принципам Дж. фон Неймана

Неупорядоченное исполнение команд – механизм исполнения команд программы в порядке, отличном от их следования в программе, обеспечивающий в результате правильную логическую последовательность выполнения программы

Организация памяти, сегментная – механизм представления памяти ЭВМ в виде совокупности логических блоков различной длины и отображения этой совокупности на запоминающие устройства, входящие в состав ЭВМ

Организация памяти, страничная – механизм разбиения памяти ЭВМ на блоки постоянной длины, использующиеся для обмена между ступенями памяти; служит (часто вместе с *сегментной организацией*) основой для построения *виртуальной памяти*

Памяти, тайминги – временн***ы***е параметры оперативных запоминающих устройств ЭВМ, характеризующие время выполнения отдельных этапов цикла обращения

Памяти, уровень – совокупность запоминающих устройств ЭВМ или системы, объединенных общим функциональным назначением и обладающих близкими характеристиками

Память, асинхронная – запоминающее устройство, сигналы цикла обращения к которому не синхронизируются

Память, виртуальная – совокупность запоминающих устройств ЭВМ, рассматриваемая логически как некоторое логически однородное пространство адресов памяти (или средства реализации такого представления)

Память, оперативная – см. *запоминающее устройство, оперативное*

Память, сверхоперативная – быстродействующее запоминающее устройство, применяющееся для ускорения обмена информацией между процессором и оперативной памятью; в настоящее время для обозначения таких устройств чаще используется термин *кэш-память*

Память, синхронная – запоминающее устройство, сигналы цикла обращения к которому синхронизируются специальной последовательностью сигналов синхронизации

Память с произвольным доступом – запоминающее устройство с адресным доступом, выбор элементов которого при записи или чтении производится посредством системы внутренних линий (обычно линий строк и столбцов), выбираемых дешифратором в соответствии с заданным адресом

Переходов предсказание – (предположительный) выбор ветви программы в разветвлении по условию до определения значения условия

Периферийное устройство – устройство, включенное в состав ЭВМ, комплекса или системы, но не входящее непосредственно в состав центрального или периферийных процессоров, системного набора микросхем и каналов ввода-вывода

Прерывание – процедура прекращения выполнения текущей программы при возникновении определенных ситуаций в ЭВМ или системе, сопровождаемая переходом к программе обработки возникшей ситуации и предполагающая возможность последующего возврата к прерванной программе

Прерываний, контроллер – контроллер, обеспечивающий прием сигналов запросов прерываний, их предварительную обработку (маскирование, приоритетный выбор) и передачу в процессор

Производительность ЭВМ – характеристика ЭВМ, отражающая затраты времени ЭВМ на решение задач; может измеряться количеством задач, решаемых в единицу времени (пропускная способность)

Процессор – центральный блок ЭВМ, осуществляющий основные действия по переработке информации, а также управлению ЭВМ в целом

Процессор, векторный – процессор, позволяющий выполнять одновременную обработку нескольких скалярных величин (обычно компонент вектора)

Процессор, суперскалярный – процессор, имеющий несколько исполнительных блоков и как правило, более одного конвейера команд

Процессора, ядро – часть процессора, обеспечивающая обработку последовательности команд, обычно включающая в себя исполнительные блоки, управляющую часть и регистры; процессор может иметь одно или несколько ядер

Прямой доступ к памяти – см. *доступ к памяти, прямой*

Расслоение обращений к памяти – способ назначения адресов ячеек (байтов) в многоблочной памяти, при котором последовательные адреса размещаются в смежных блоках памяти

Регенерация информации в динамических ЗУ – периодически выполняемая процедура восстановления записанной в ЗУ информации, необходимость которой вызвана самопроизвольным разрядом запоминающих конденсаторов

Режим работы ЭВМ – порядок выполнения программ в ЭВМ и особенности его взаимодействия с пользователями; различают однопрограммный и многопрограммные режимы, режимы коллективного доступа, реального времени, пакетной обработки

Северный мост – см. *мост, северный*

Сервер – ЭВМ, имеющая специальное функциональное назначение или структуру, ориентированную на обеспечение специальных требований

Сеть, вычислительная – совокупность вычислительных машин, соединенных между собой каналами связи, как правило, разнесенных территориально

Сеть, межсоединений – см. *межсоединений, сеть*

Система, вычислительная – совокупность устройств, содержащих один или более процессоров (или ЭВМ), запоминающих и периферийных устройств, объединенных шинами расширения или каналами связи

Способ организации вычислительного процесса – способ задания последовательности действий, обеспечивающих решение задачи на ЭВМ; различают алгоритмическую (программную) и схемную организацию вычислительного процесса

Суперскалярный процессор – см. *процессор, суперскалярный*

Схемное устройство управления – см. *устройство управления, схемное*

Устройство управления (УУ) – устройство, вырабатывающее последовательность управляющих сигналов для операционных блоков в соответствии с заданным алгоритмом управления

Устройство управления микропрограммное – устройство управления, последовательность микрокоманд в котором записана в постоянной памяти

Устройство управления, схемное – устройство управления, последовательность микрокоманд в котором формируется с помощью управляющего автомата, состоящего из памяти состояний и комбинационной схемы

Форма представления информации – вид отображения информации, используемый в вычислительной машине; две основные формы: алфавитная (цифровая) и аналоговая

Шина расширения – внутренняя шина ЭВМ, используемая для подключения к ней дополнительных устройств, например шина *PCI*

Эмуляция – воспроизведение программными или аппаратными средствами одной ЭВМ или устройства функционального поведения другой ЭВМ или устройства

Эффективность ЭВМ – критерий, характеризующий степень соответствия ЭВМ своему назначению, или в простом случае соотношение затрат и производительности ЭВМ

Южный мост – см. *мост, южный*

Ядро процессора – см. *процессора, ядро*

3.3. Методические указания к выполнению лабораторных работ

По дисциплине предусмотрено выполнение лабораторного практикума. Тематика лабораторных работ охватывает вопросы исследования структурной организации и процессов функционирования арифметических устройств, устройств управления и ЭВМ в целом. В зависимости от формы обучения и объема занятий, выделенных на проведение лабораторных работ, содержание программы работ может варьироваться.

При выполнении работ № 1 и № 2 используются моделирующие программы ALU_V16.EXE (или ALU3.EXE) и MUP.EXE, расположенные на учебном сайте СЗТУ, а также в разделе сайта www.ord.com.ru/files/org_evm. Варианты заданий к этим работам следует получить у преподавателя. Примеры заданий можно найти по той же ссылке www.ord.com.ru/files/org_evm.

Лабораторная работа № 1

ИССЛЕДОВАНИЕ СТРУКТУРЫ И ПРИНЦИПА ДЕЙСТВИЯ ДВОИЧНОГО АРИФМЕТИЧЕСКОГО УСТРОЙСТВА

1. Цель работы

Исследование работы двоичного арифметического устройства, выполняющего операции над двоичными числами с фиксированной запятой.

2. Основные теоретические положения

Арифметические устройства (АУ) предназначаются для переработки информации в ЭВМ. В состав АУ входят сумматор, регистры и ряд вспомогательных элементов и узлов.

В зависимости от типа используемого сумматора (комбинационного или накапливающего) структура АУ и алгоритмы выполнения операций в нем имеют некоторые различия. Так, в АУ на основе накапливающего сумматора обычно имеются еще два регистра. Микрооперации сдвига выполняются непосредственно в сумматоре и регистрах, преобразования кодов (из прямого в обратный и наоборот) могут осуществляться непосредственно в сумматоре и при передаче операндов из регистров в сумматор. Значения знаковых и других разрядов операндов, определяющих ход выполнения операции (например, значения разрядов множителя), анализируются непосредственно по месту хранения операндов.

В состав АУ на основе комбинационного сумматора, помимо собственно сумматора, входят еще три-четыре регистра. Микрооперации сдвига, как правило, осуществляются в специальном узле – сдвигателе, преобразования кодов выполняются только при передаче чисел из регистров в сумматор. Значения знаковых и других, необходимых по ходу выполнения операции разрядов анализируются обычно лишь на выходных шинах сумматора или вспомогательных регистров, временно сохраняющих значения выходных сигналов сумматора. Для этого операнды подаются на один из выходов сумматора при нулевом коде, поданном на его второй вход (передача транзитом). Типичная структура АУ на основе комбинационного сумматора представлена на рис. Л1.1, где *P1*, *P2*, *P3* –

регистры; *Сдв* – сдвигатель; *ВР* – выходной (промежуточный) регистр; *М1* и *М2* – мультиплексоры входов 1 и 2 сумматора, позволяющие также осуществлять преобразование кодов (один или оба); *ДР* – дополнительный разряд (разряды) для хранения информации, выходящей при сдвигах за пределы разрядной сетки.

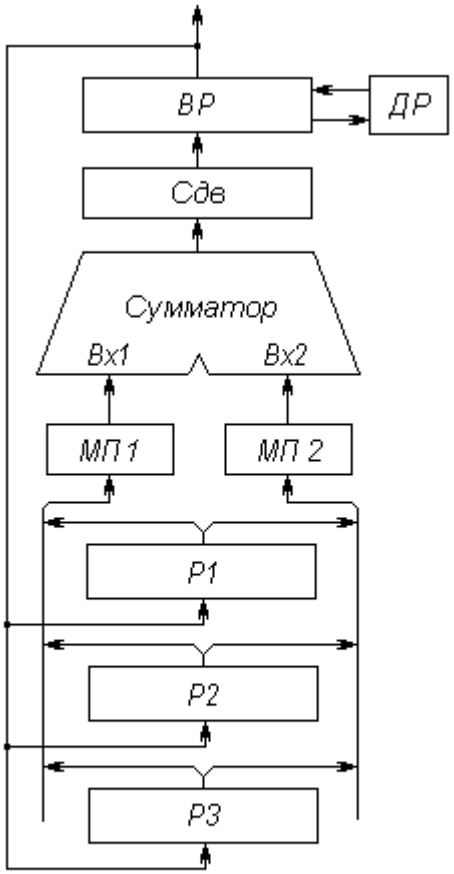


Рис. Л1.1. Структура АЛУ на основе комбинационного сумматора

Двоичные числа с фиксированной запятой могут храниться как в прямом, так и в дополнительном коде памяти ЭВМ. Обработка чисел непосредственно в дополнительном коде позволяет исключать этапы предварительного преобразования кодов операндов и преобразования кода результата, в общем случае приводя к сокращению времени выполнения операции. Это сокращение оказывается особенно существенным для операций сложения и вычитания.

Рассмотрим особенности выполнения операций с помощью лабораторного макета АУ, построенного на базе комбинационного сумматора (рис. Л1.2). Алгоритм сложения двоичных чисел с фиксированной запятой, представленных в прямом коде, включает следующие шаги:

- анализ знака первого слагаемого и преобразование его в обратный (дополнительный) код, если слагаемое отрицательное;
- анализ знака и преобразование кода второго слагаемого;
- суммирование преобразованных кодов;
- анализ результата на переполнение разрядной сетки;
- анализ знака результата и преобразование обратного (дополнительного) кода отрицательного результата в прямой.

При использовании обратного кода необходимо сигнал переноса из знакового разряда подавать в младший разряд сумматора (циклический перенос). При использовании дополнительного кода этого не требуется.

Если необходимо выполнить операцию вычитания, то знак вычитаемого изменяется на обратный.

Сложение чисел, представленных в дополнительном коде, включает лишь третий и четвертый шаги, причем коды слагаемых преобразованиям не подвергаются.

При вычитании код вычитаемого преобразуется в дополнительный вне зависимости от знака и преобразование охватывает все разряды, включая знаковые. Это преобразование производится передачей на вход сумматора инверс-

ных значений разрядов уменьшаемого и добавлением 1 к младшему разряду сумматора, что выполняется одновременно с подачей уменьшаемого на второй вход сумматора.

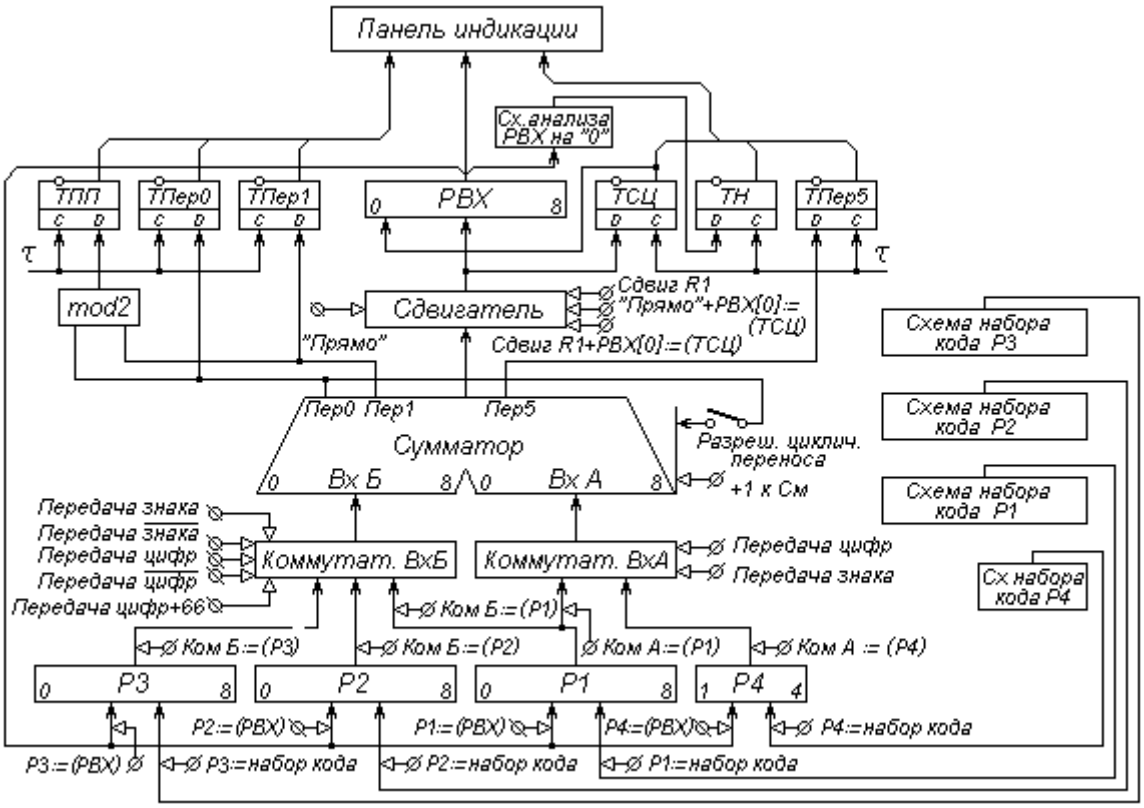


Рис. Л1.2. Структурная схема макета АЛУ

Признаком переполнения разрядной сетки при наличии только одного знакового разряда в сумматоре может служить несовпадение сигналов переноса из старшего цифрового и знакового разрядов.

При умножении наиболее часто используется алгоритм умножения со сдвигом вправо суммы частичных произведений, начиная с младших разрядов множителя. Возможность реализации такого алгоритма умножения и предусматривается лабораторным макетом.

В АУ на основе комбинационного сумматора множимое, множитель и произведение (сумма частичных произведений) хранятся на отдельных регистрах. Алгоритм умножения чисел, представленных в прямом коде, включает в себя следующие шаги:

- анализ младшего разряда множителя и добавление множимого к сумме частичных произведений при единичном значении анализируемого разряда;
- сдвиг множителя и суммы частичных произведений вправо;
- определение знака произведения.

Первый и второй шаги повторяются столько раз, сколько разрядов имеет множитель. Во время их выполнения знаковый разряд множимого не принимается во внимание и не участвует в преобразовании суммы частичных произве-

дений. Знак произведения определяется как сумма по mod2 знаков сомножителей.

Сдвиг суммы частичных произведений и множителя осуществляется поочередно посредством подачи их на один из входов сумматора с последующей передачей сигналов с его выхода через сдвигатель со сдвигом вправо на регистр, в котором хранится множитель или сумма частичных произведений.

В качестве счетчика количества повторений цикла умножения может использоваться обычный регистр, информация из которого передается через сумматор обратно в регистр, причем в момент передачи на второй вход сумматора подается обратный код единицы.

При умножении чисел, представленных в дополнительном коде, без перевода сомножителей в прямой код учитываются следующие особенности. При отрицательном множимом и положительном множителе отличие от умножения в прямых кодах состоит лишь в использовании модифицированного (с сохранением значения знакового разряда) сдвига суммы частичных произведений. Если множитель отрицателен, то независимо от знака множимого при нулевом значении анализируемого разряда множителя к сумме частичных произведений добавляется дополнение кода множимого. Перед началом операции к нулевому значению суммы частичных произведений следует добавить дополнение кода множимого, т. е. вычесть множимое. При положительных сомножителях их дополнительные коды совпадают с прямыми, поэтому отличий от умножения чисел, представленных прямыми кодами, не имеется.

3. Описание лабораторной установки

Используемый в работе лабораторный макет изготовлен на микросхемах серии 155. Структурная схема макета приведена на рис. Л1.2 и включает в себя следующие узлы:

- девятиразрядный двоичный комбинационный сумматор (один знаковый и восемь цифровых разрядов), предназначенный для суммирования двух двоичных кодов, подаваемых на его входы;
- три девятиразрядных регистра *Р1, Р2, Р3*, предназначенные для хранения кодов слагаемых, сомножителей и результатов операции, и один четырехразрядный регистр *Р4*, используемый для подсчета числа повторений циклов при выполнении операции умножения;
- регистр временного хранения *РВХ*, используемый для приема сигналов с выходных шин сумматора и хранения их до передачи на один из вышеназванных регистров;
- сдвигатель, позволяющий осуществить занесение сигналов с выходных шин сумматора на *РВХ* со сдвигом на один разряд вправо или без сдвига;
- коммутаторы входов *А* и *Б* сумматора, позволяющие подавать на вход *Б* сумматора содержимое *Р1, Р2* или *Р3* прямым или обратным кодом, а на вход *А* – содержимое *Р1* или *Р4*;
- блок индикации на светодиодах, позволяющий индицировать состояние триггеров регистра *Р3* и ряда вспомогательных триггеров.

Кроме того, в состав макета входят также некоторые вспомогательные триггеры и схемы, в том числе:

- схема анализа содержимого *РВХ* на нуль, чей выход связан с триггером нулевого результата *ТН*; состояние *ТН* индицируется на панели индикации, причем единичное состояние *ТН* (зажженный светодиод) соответствует установке на *РВХ* нулевого кода;
- схема суммирования по mod2 переносов из старшего цифрового и знакового разрядов, выход которой связан с триггером переполнения *ТПП*; схема используется для выявления случаев переполнения, для чего состояние триггера *ТПП* индицируется на панели индикации;
- триггер сдвинутой цифры *ТСЦ*, предназначенный для запоминания значения выходного сигнала с шин младшего разряда сумматора при подаче его содержимого в *РВХ* со сдвигом вправо;
- три триггера переносов *ТПер0*, *ТПер1* и *ТПер5*, предназначенные для фиксации сигналов переноса из знакового, первого и пятого цифровых разрядов, что необходимо для выполнения операций десятичной арифметики.

Исходная информация в регистры *Р1* и *Р4* заносится с помощью тумблерного набора. Кроме того, информация в эти регистры может передаваться из регистра *РВХ*.

Управление операциями передачи информации между регистрами преобразования кодов, сдвига и другими осуществляется при помощи выведенных на лицевую панель макета тумблеров и кнопок с соответствующими обозначениями. Информация в триггеры *ТПП*, *ТН*, *ТСЦ*, *ТПер0*, *ТПер1*, *ТПер5* заносится автоматически при передаче в *РВХ* сигналов выхода сумматора через сдвигатель.

В макете предусматривается возможность следующего набора микроопераций:

y_1 – *Р1[0:8]*:= *набор кода* (занесение кода с тумблерного набора на *Р1*);

y_2 – *Р2[0:8]*:= *набор кода*;

y_3 – *Р3[0:8]*:= *набор кода*;

y_4 – *Р4[1:4]*:= *набор кода*;

y_5 – *Уст."0"* (начальная установка: *Р1* := *Р2* := *Р3* := *Р4* := *РВХ* := *ТСЦ* := *ТПП* := *ТПер0* := *ТПер1* := *ТПер5* := 0);

y_6 – *Р1* := (*РВХ*) – передача содержимого из *РВХ* в *Р1*;

y_7 – *Р2* := (*РВХ*) – передача содержимого из *РВХ* в *Р2*;

y_8 – *Р3* := (*РВХ*) – передача содержимого из *РВХ* в *Р3*;

y_9 – *Р4[1...4]* := (*РВХ[5...8]*);

y_{10} – *КомА* := (*Р1*) – подача содержимого регистра *Р1* на вход коммутатора входа *А* сумматора;

y_{11} – *КомА* := (*Р4*);

y_{12} – *КомБ* := (*Р1*);

y_{13} – *КомБ* := (*Р2*);

y_{14} – *КомБ* := (*Р3*);

y_{15} – "*Передача знака КомБ*" – *Вход Б[0]* := (*КомБ[0]*) – передача в знаковый разряд входа Б сумматора знака кода, поданного на вход коммутатора Б;

y_{16} – "*Передача знака КомБ*" – *Вход Б[0]* := (*КомБ[0]*);

y_{17} – "*Передача цифр КомБ*" – *Вход Б[1...8]* := (*КомБ[1...8]*) – передача на входные шины цифровых разрядов входа *Б* сумматора цифровых разрядов кода, поданного на коммутатор *Б*;

y_{18} – "*Передача цифр КомБ*" – *Вход Б[1...8]* := ($\overline{КомБ[1...8]}$);

y_{19} – "*Передача цифр КомБ*" – *Вход Б[1...8]* := (*КомБ[1...8]*)+66 – передача на цифровые разряды входа *Б* сумматора, увеличенного на 66 значений числа, поданного на коммутатор *Б*;

y_{20} – "*Передача знака КомА*" – *Вход А[0]* := (*КомА[0]*);

y_{21} – "*Передача цифр Ком.А*" – *Вход А[1...8]* := (*Ком.А[1...8]*);

y_{22} – "*+1 к См*" – добавление единицы в младший разряд сумматора;

y_{23} – "*Разрешение циклического переноса*" – замыкание цепи связи сигнала переноса из знакового разряда сумматора со входом переноса младшего разряда сумматора;

y_{24} – "*Прямо*" – *РВХ[0...8]* := (*См[0...8]*) – занесение кода с выходных шин сумматора в *РВХ*;

y_{25} – "*Прямо + РВХ[0]* := *ТСЦ*" – *РВХ[1...8]* := *См[1...8]*, *РВХ[0]* := (*ТСЦ*);

y_{26} – "*Сдвиг R1*" – *РВХ[1...8]* := (*См[0...7]*), *ТСЦ* := (*См[8]*) – занесение кода с выходных шин сумматора в *РВХ* со сдвигом на один разряд вправо, а младшего разряда кода – в *ТСЦ*;

y_{27} – "*Сдвиг Р1 + РВХ[0]* := *ТСЦ*" – *РВХ[1...8]* := (*См[0...7]*), *РВХ[0]* := (*ТСЦ*), *ТСЦ* := (*См[8]*).

Питание макета осуществляется от сети напряжением ~220 В. Включение питания производится тумблером *"Вкл."*, расположенным на лицевой панели макета. О подаче напряжения сигнализирует зажигание зеленого светодиода на панели индикации.

Функционирование макета воспроизводит моделирующая программа, ALU_V16.EXE, позволяющая имитировать выполнение всех микроопераций макета и дополнительно микрооперации левого сдвига, что позволяет реализовать микропрограмму операции деления.

Для реализации условных переходов в программе предусматривается возможность проверки логических условий, к основным из которых относятся:

x1 – состояние *ТПП*;
x2 – состояние *ТПер0*;
x3 – состояние *ТПер1*;
x4 – состояние *ТПер5*;
x5 – состояние *ТСЦ*;
x6 – состояние *ТН*;

x7 – состояние *ТЗН*;

...

x19 – значение *PBX[0]*.

Микропрограмма вводится в моделирующую программу в следующем формате:

- оператор выполнения микроопераций (микрокоманда)

NOп	Y	NMO1	NMO2	. . .	NMO10

- оператор условного перехода

NOп	X	NУсл	NOп1	NOп2

- оператор безусловного перехода

NOп	Z	NOп1

где Nоп – номер оператора;
X, Y, Z – указатели типа оператора;
NMO*i* – номер микрооперации;
NУсл – номер условия, определяющего ветвь перехода в операторе условного перехода;
NOп1 – номер оператора, которому передается управление, в операторе безусловного перехода и в операторе условного перехода при ложном значении условия;
NOп2 – номер оператора, которому передается управление, в операторе условного перехода при истинном значении условия.

Составленная микропрограмма может быть сохранена в файле и впоследствии загружена для исполнения.

4. Порядок выполнения работы

1. Ознакомиться с особенностями алгоритмов выполнения операций сложения, вычитания и умножения двоичных чисел с фиксированной запятой, представленных в прямом и дополнительном кодах по настоящему описанию.

2. Изучить схему лабораторного макета и назначение его основных узлов.

3. Составить алгоритмы сложения (вычитания) двоичных чисел с фиксированной запятой, представленных в прямом (дополнительном) коде, реализуемые на базе имеющихся в макете блоков.

4. Ввести составленную микропрограмму в моделирующую программу и выполнить действия для кодов операндов, заданных преподавателем.

5. Составить алгоритмы умножения (деления) двоичных чисел с фиксированной запятой, представленных в прямом (дополнительном) коде, ориентированные на реализацию с использованием имеющихся в макете узлов.

6. Ввести составленную микропрограмму в моделирующую программу и выполнить действия для кодов сомножителей, заданных преподавателем.

В соответствии с указанным порядком перед началом работы необходимо изучить особенности выполнения операций сложения, вычитания и умножения чисел с фиксированной запятой, представленных в прямом и дополнительном кодах. Следует рассмотреть особенности лабораторного макета и знать назначение основных его узлов и органов управления.

При составлении алгоритмов сложения и вычитания чисел, представленных в прямом коде, необходимо предусмотреть анализ знаков операндов по состоянию триггера знака. Для этого операнды поочередно передаются транзитом через сумматор на *РВХ*, в результате чего триггер знака устанавливается в соответствующее состояние, что индицируется на панели индикации.

Составляя алгоритмы и микропрограммы, целесообразно начать их представление с наиболее общего уровня, а затем поэтапно раскрывать их, доводя до уровня микроопераций.

Например, если задание на лабораторную работу предусматривает составление микропрограммы сложения двоичных чисел, представленных в прямых кодах, которые изначально расположены в регистрах Р2 и Р3, с получением результата в регистре Р3 (Р2 + Р3 → Р3), то общий вид алгоритма будет выглядеть так, как показано на рис. Л1.3.

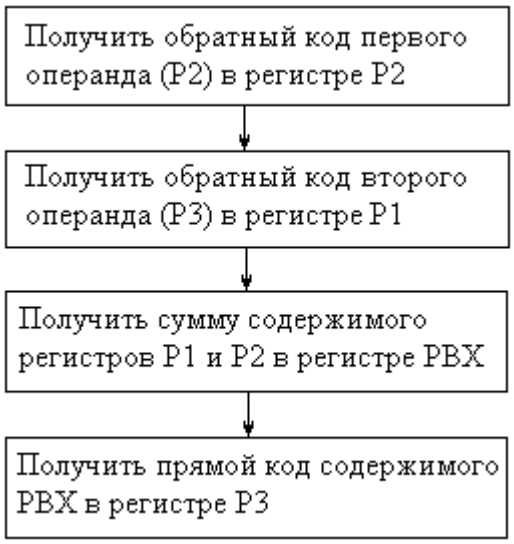


Рис. Л1.3. Общий вид алгоритма выполнения операции сложения

Здесь следует обратить внимание на то, что во втором блоке обратный код содержимого регистра Р3 помещается в регистр Р1. Это необходимо потому, что в структуре макета АЛУ содержимое регистров Р2 и Р3 можно подать только на вход Б сумматора, что не позволяет получить их сумму. В то время как регистры Р1 и Р2 (или Р1 и Р3) можно подать на два входа сумматора одновременно.

На следующем шаге необходимо изобразить алгоритм на уровне межрегистровых передач между узлами макета. Учитывая, что получить инверсию цифр слагаемых можно только при передаче через коммутатор входа Б, а проверку знака числа можно выполнить только в регистре РВХ, то следует сперва передать содержимое обрабатываемого регистра в РВХ, проанализировать знак

РВХ и, если число отрицательно, инвертировать его цифры, не изменяя знак. Эти действия для первого из рассмотренных блоков будут выглядеть так, как показано на рис. Л1.4.

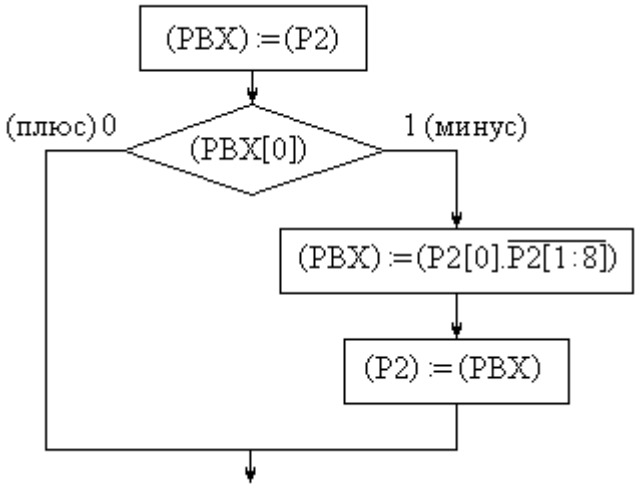


Рис. Л1.4. Представление первого блока алгоритма (рис. Л1.3) на уровне передач между узлами моделируемого АЛУ

Далее каждый из операторных блоков этого уровня представления алгоритма (микропрограммы) заменяется группой микроопераций, обеспечивающих его исполнение. Соответствующий фрагмент граф-схемы, использующий микрооперации моделируемого АЛУ, показан на рис. Л1.5.

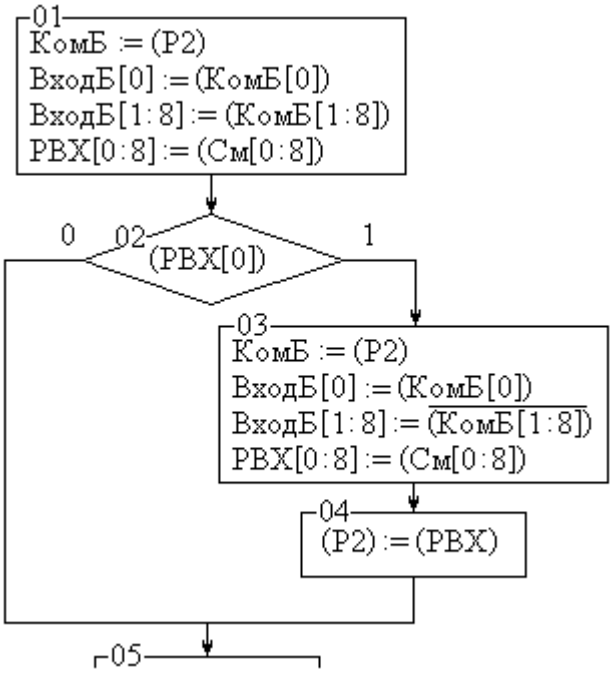


Рис. Л1.5. Представление первого блока алгоритма (рис. Л1.3) на уровне микроопераций моделируемого АЛУ

Последний этап подготовки микропрограммы заданной операции сводится к записи ее в формате представления операторов микропрограммы в моделирующей программе. В приводимом примере с учетом номеров микроопераций этот фрагмент микропрограммы будет выглядеть так, как показано ниже:

01 Y 13 15 17 24
02 X 19 05 03
03 Y 13 15 18 24
04 Y 07.

Задание номеров операторов в операторах условных переходов в зависимости от версии моделирующей программы может требовать указания одноразрядных номеров двумя цифрами с первым нулем либо просто одной цифрой.

При составлении алгоритмов умножения (деления) в качестве счетчика циклов следует использовать регистр *P4*, устанавливая в нем в исходном состоянии код числа разрядов множителя. Для вычитания единицы из содержимого *P4* необходимо подать код, набранный на *P4*, на вход сумматора. На вход *Б* сумматора подать 11...1 (дополнительный код "−1"), для этого включить тумблер *"Передача знака"*. Во время последнего вычитания единицы из содержимого *P4* при получении нулевого результата триггер *ТН* устанавливается в единичное состояние, что будет индицироваться зажиганием соответствующего светодиода, сигнализируя о выполнении требуемого количества сигналов суммирования и сдвига при умножении.

Анализ значения очередного разряда множителя следует производить по состоянию триггера *ТСЦ*, которое соответствует значению очередного разряда при сдвиге множителя вправо, выполняемом посредством передачи его транзитом через сумматор со сдвигом вправо при занесении в *РВХ* и последующей пересылкой его в *РВХ* обратно в регистр множителя.

При умножении полноразрядных сомножителей (8 цифровых разрядов) для передачи младших разрядов суммы частичных произведений в освобождающиеся при сдвиге множителя разряды регистра множителя также следует воспользоваться триггером *ТСЦ*. Для этого попадающая в него при сдвиге суммы частичных произведений младшая цифра данной суммы при последующем выполнении сдвига множителя заносится в нулевой разряд *РВХ* и далее передается в регистр множителя.

Определение знака произведения целесообразно производить в начале операции, сложив на сумматоре знаковые разряды сомножителей и занеся результат в регистр множимого. Это можно осуществить подачей множимого на один из входов сумматора и одновременной подачей на второй его вход только знакового разряда множителя, после чего занести результат в регистр множимого.

Особенности работы с макетом

При выполнении заданных операций следует включить питание, затем нажать кнопку *"Уст.0"*, устанавливающую все узлы макета в исходное состояние, набрать на тумблерном наборе заданные преподавателем коды и занести их в соответствующие составленным алгоритмам выполнения операций регистры, нажав и отпустив кнопки с занесением кодов с тумблерного набора *P := "Набор кода"*. После этого необходимо выполнить все микрооперации, определяемые составленными алгоритмами.

Для обеспечения одновременной подачи управляющих сигналов различных микроопераций одни сигналы формируются при нажатии соответствующих кнопок на лицевой панели макета, а другие – при установке тумблеров в верхнее положение, оставаясь при этом поданными на все время нажатия кнопки или включения тумблера. Тумблеры используются для управления режимами передачи коммутаторов входов сумматора, цепью циклического переноса и для добавления единицы в младший разряд сумматора. Все остальные сигналы вырабатываются при нажатии кнопок. Это позволяет предварительно установить соответствующий режим передачи, например передачу обратного кода, а затем осуществить собственно передачу информации нажатием нужной кнопки.

При выполнении действий, связанных с суммированием, сдвигом и с передачей данных через сумматор транзитом, следует обратить внимание на последовательность нажатия и отпускания кнопок. Для правильного функционирования схемы, например, при суммировании, необходимо сперва нажать и удержать в этом положении кнопки передачи информации с регистров на коммутаторы входов сумматора, затем нажать и отпустить кнопку занесения сигналов с выходных шин сумматора на *PBX* через сдвигатель и только после этого можно отпустить кнопки подачи информации на коммутаторы.

Для индикации результата, не находящегося в *P3* по окончании операции, его следует передать в этот регистр, так как с индикацией связан только регистр *P3*.

5. Содержание отчета

1. Структурная схема лабораторного макета и описание назначения его основных узлов.

2. Алгоритмы выполнения операций сложения (вычитания) и умножения (деления), составленные с учетом возможности реализации их на макете, а также микропрограммы этих операций с указанием используемых микроопераций в форме, пригодной для ввода в моделирующую программу.

3. Примеры выполнения операций по составленным алгоритмам с указанием содержимого регистров, участвующих в операции, в последовательные моменты времени.

Литература: [1], с. 337…372; [5], с. 53…67

Лабораторная работа № 2

ИССЛЕДОВАНИЕ УСТРОЙСТВА МИКРОПРОГРАММНОГО УПРАВЛЕНИЯ

1. Цель работы

Цель работы состоит в приобретении практических навыков составления микропрограмм и размещения их в управляющей памяти (памяти микрокоманд) на примере простого макета устройства микропрограммного управления.

2. Основные теоретические положения

Микропрограммные устройства управления (МПУУ) предназначаются для управления процессами переработки информации или формирования управляющих воздействий в различных средствах цифровой техники – от простых контроллеров до ЭВМ средней производительности. Эти устройства в ряде случаев более эффективны, чем устройства управления с жесткой логикой.

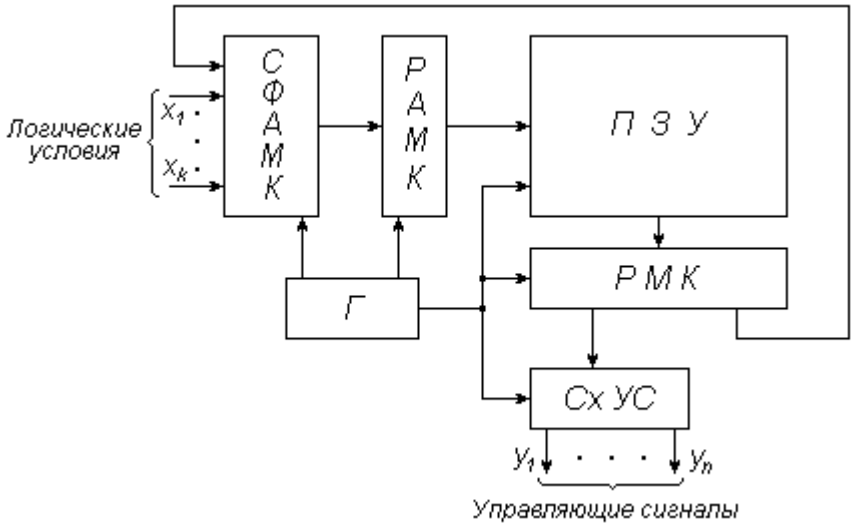


Рис. Л2.1. Структура микропрограммного УУ

В состав МПУУ, структура которого представлена на рис. Л2.1, входят: постоянное запоминающее устройство *ПЗУ*, называемое также ЗУ микрокоманд, регистр микрокоманд *РМК*, регистр адреса микрокоманды *РАМК*, схема формирования управляющих сигналов *СхУС*, схема формирования адреса следующей микрокоманды *СФАМК* и задающий генератор *Г*.

Постоянное *ЗУ* служит для хранения микропрограмм (МП), представляющих собой последовательность микрокоманд (МК), которые обеспечивают требуемую очередность подачи управляющих сигналов в операционную часть управляемой системы.

Регистр МК предназначен для хранения выполняемой в данный момент микрокоманды. Регистр адреса МК используется для хранения адреса МК во время выборки ее из *ПЗУ*.

СхУС служит для выработки управляющих сигналов (сигналов микроопераций), задаваемых микрокомандой, находящейся на *РМК*. Необходимость такой схемы обусловлена тем, что не во всех МПУУ разряды МК однозначно соответствуют управляющим сигналам.

СФАМК вырабатывает адрес следующей микрокоманды в соответствии с информацией, записанной в адресной части текущей МК, и значениями логических условий, поступающих в *СФАМК*.

Генератор служит для выработки синхросигналов, обеспечивающих согласование взаимодействия всех узлов МПУУ при выполнении МК.

Особенности построения и функционирования МПУУ рассматриваются в литературе [1], [6].

3. Описание лабораторного макета

Используемый в работе лабораторный макет изготовлен на микросхемах серии 155. Структурная схема макета приведена на рис. Л2.2. В его состав входят:

а) запоминающее устройство микрокоманд (*ЗУМК*) емкостью 64 12-разрядных слова, используемое для записи в него составляемых микропрограмм. В отличие от реального МПУУ в макете вместо постоянного ЗУ использовано оперативное, что позволяет заносить в *ЗУМК* составляемые МП;

б) 12-разрядный регистр микрокоманды (*РМК*), назначение которого такое же, как в рассмотренной выше схеме МПУУ;

в) 6-разрядный регистр адреса МК (*РАМК*) в макете используется для хранения адреса МК при выборке МК из *ЗУМК* и, в отличие от обычного МПУУ, при записи МК в *ЗУМК*;

г) блок дешифрирования и индикации управляющих сигналов (*БДИУС*), используемый в макете для наглядного представления содержимого поля управляющих сигналов МК (рис. Л2.3). Разряды этого поля высвечиваются на индикации двумя восьмеричными цифрами, первая из которых отображает содержимое разрядов 0...2 микрокоманды, а вторая - разрядов 3 и 4. Такая индикация соответствует случаю кодирования полей совместимых микроопераций [6], причем при нулевом содержимом названных разрядов на индикации не высвечивается никаких цифр, так как этот случай обычно указывает на отсутствие управляющих сигналов в соответствующих полях;

д) схема формирования адреса следующей микрокоманды (*СФАМК*), с помощью которой адрес очередной микрокоманды формируется в *РАМК*: старший разряд регистра *РАМК[0]* устанавливается всегда тумблером А0 набора адреса МК. В разряды *РАМК[1...4]* засылается содержимое разрядов *РМК[7...10]*. В разряд *РАМК[5]*, т. е. в младший разряд *РАМК*, схема *СФАМК* передает:

- содержимое *РМК[11]*, если в разрядах *РМК[5,6]* (см. рис. Л2.3, поле номера условия) содержатся нули (этот случай соответствует безусловному переходу в МП);

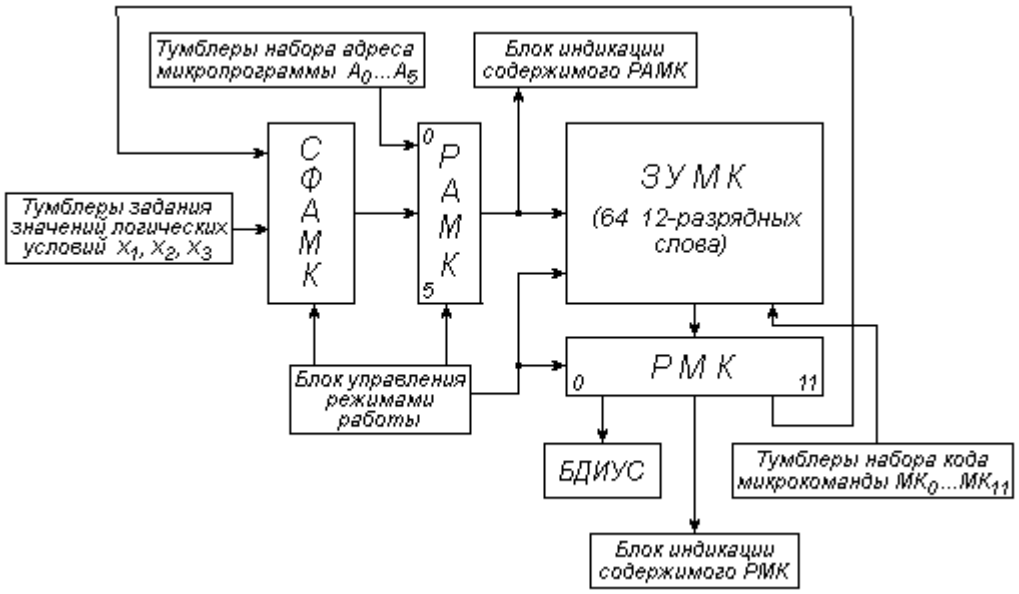


Рис. Л2.2. Структурная схема лабораторного макета МПУУ

- значение одного из логических условий x_1, x_2 или x_3, номер которого соответствует коду *РМК[5,6]*, если содержимое разрядов *РМК[5,6]* отлично от нуля. Этот случай является случаем выполнения МК с условным переходом, причем значение условий задается тумблерами задания условий;

е) блоки индикации содержимого *РАМК* и *РМК*, позволяющие индицировать информацию, находящуюся в этих регистрах на лицевой панели макета в двоичном коде;

ж) тумблеры набора.

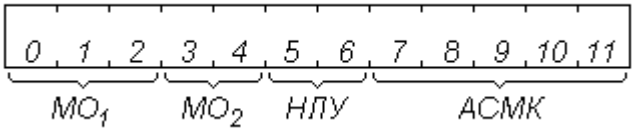


Рис. Л2.3. Формат микрокоманды

Питание макета осуществляется от сети напряжением 220 В. Включение питания производится тумблером *"Вкл."*, расположенным на лицевой панели макета. О подаче напряжения сигнализирует подсветка зеленой лампочки *"+5 В"* на лицевой панели макета.

4. Порядок выполнения работы

1. Ознакомиться со структурой устройств микропрограммного управления, особенностями взаимодействия их блоков в процессе выполнения микрокоманд.

2. Ознакомиться со схемой лабораторного макета и назначением его основных узлов.

3. Составить по заданной преподавателем граф-схеме микропрограммы линейного типа микропрограмму в виде последовательности микрокоманд, формат которых принят в макете.

4. Записать составленную микропрограмму в моделирующую программу MUP.EXE и, проследив за ходом ее выполнения, убедиться в правильности выработки последовательности управляющих сигналов.

5. Составить для заданной преподавателем граф-схемы циклической микропрограммы с разветвлениями микропрограмму в пригодном для записи в ЗУМК виде.

6. Записать составленную микропрограмму в моделирующую программу MUP.EXE и убедиться в правильности ее выполнения при различных значениях условий, определяющих переходы в микропрограмме.

7. Дать оценку времени выполнения (в тактах) циклической микропрограммы с разветвлениями, учитывая особенности выполнения переходов в макете.

5. Указания к выполнению работы

При выполнении п.п. 1 и 2 необходимо изучить структуру МПУУ и порядок взаимодействия его блоков, уяснить отличия лабораторного макета от реального МПУУ и причины этих отличий, ознакомиться с назначением основных узлов управления и индикации, имеющихся в макете.

При выполнении п. 3 каждой операторной вершине граф-схемы линейной микропрограммы требуется сопоставить одну ячейку *ЗУМК*. Для этого удобнее всего последовательно пронумеровать операторные вершины граф-схемы МП, начиная с первой, которой присваивается номер нуль, и далее рассматривать эти номера как адреса ячеек *ЗУМК*, в которые записываются соответствующие вершинам граф-схемы микрокоманды. Конечно, возможна и любая иная нумерация вершин (без повторений).

Для более наглядного представления МП целесообразно записывать размещение микропрограммы в ЗУМК в таблицу по форме 1.

Форма 1

Адрес МК	**Код МК**			
A_0 A_1 A_2 A_3 A_4 A_5	$МО_1$	$МО_2$	НЛУ	АСМК
. . .	. . .	. . .	. . .	. . .

При выполнении п. 4 необходимо с разрешения преподавателя включить питание макета. Для записи составленной микропрограммы в ЗУМК следует перевести макет в режим *РУЧН*, поставив в соответствующее положение переключатель режима работы "*РУЧН-АВТ*" (выключенное положение переключателя, кнопка отпущена), а также перевести переключатель режима обращения к *ЗУМК* "*ЗП-ЧТ*" в положение *ЗП*.

После этого в *ЗУМК* можно последовательно записать микрокоманды составленной микропрограммы. Для записи каждой МК необходимо:

- набрать на тумблерах набора адреса микрокоманды $A_0...A_5$ код адреса ячейки *ЗУМК*, в которую производится запись МК;

- набрать на тумблерах набора микрокоманды $МК_0...МК_{11}$ код записываемой в данную ячейку МК;

- нажать и отпустить кнопку "*ПУСК*", что приведет к записи МК в *ЗУМК* и индикации на лицевой панели макета содержимого *РМК* и *РАМК*, соответствующего записываемой МК и адресу *ЗУМК*, по которому эта МК записана.

Синхронизация макета построена таким образом, что при нажатии кнопки "*ПУСК*" код адреса МК с тумблерного набора заносится в *РАМК*, а при ее отпускании – код микрокоманды записывается в *ЗУМК* и заносится в *РМК*.

Подобным образом в *ЗУМК* поочередно записываются все МК составленной МП. Причем сброс *РАМК* и *РМК* в приведенной последовательности операций необходим, так как при нажатии кнопки "*ПУСК*" в *РАМК* формируется дизъюнкция кода, набранного на тумблерах набора адреса, и содержимого разрядов 7...11 регистра *РМК*.

Для проверки правильности выполнения составленной МП следует перевести переключатель "*ЗП-ЧТ*" в положение *ЧТ*, набрать на тумблерах $A_0...A_5$ адрес ячейки *ЗУМК*, в которой записана первая МК проверяемой МП, нажать и отпустить кнопку сброса "*НУ*".

После этого при нажатии и отпускании кнопки "*ПУСК*" считывается из *ЗУМК* и передается на *РМК* та МК, адрес которой установлен на тумблерах набора адреса. На лицевой панели считанная микрокоманда будет индицироваться как содержимое *РМК*, а ее первые пять разрядов, соответствующие полям управляющих сигналов $МО_1$ и $МО_2$ (рис. Л2.3), индицируются также цифровыми индикаторами.

При нажатии кнопки "*ПУСК*" в следующий раз будет сформирован адрес очередной МК, выполнено чтение ее из *ЗУМК* и передача на *РМК*. Следует учитывать, что для правильного формирования адреса второй и последующих микрокоманд необходимо после чтения первой МК переключить тумблеры набора адреса в нулевое положение. В противном случае, как указано выше, адрес очередной МК будет равен дизъюнкции кода, набранного на тумблерах набора адреса, и адресной части текущей МК. Поэтому удобно размещать микропрограммы в *ЗУМК*, начиная с адресов 000000 и 100000.

Поочередно нажимая кнопку "*ПУСК*", следует убедиться, что последовательность управляющих сигналов, появляющихся в операционных частях считываемых друг за другом МК, соответствует заданной граф-схеме исходной микропрограммы.

П. 5 выполняется аналогично п. 3. Однако здесь следует учитывать, что МК, в которых выполняется проверка условия перехода, передают управление в зависимости от значения проверяемого условия только на такие МК, адреса которых в *ЗУМК* различаются лишь значением младшего разряда. Это несколько усложняет задачу размещения микропрограмм в *ЗУМК*.

Поэтому при размещении МП с разветвлениями в *ЗУМК* целесообразно начать назначение адресов операторным вершинам с тех вершин, которые сле-

дуют непосредственно за условными. Причем двум операторным вершинам, связанным с различными выходами одной и той же условной вершины, присваиваются адреса, все разряды которых, кроме младшего, совпадают. Младшие же разряды адресов полагаются равными значениям условия, проставленным у соответствующих выходов условной вершины. Пример такого назначения адресов представлен на рис. Л2.4, где вершинам 2 и 4 фрагмента микропрограммы сопоставлены адреса *ЗУМК* $a_0a_1a_2a_3a_4$1 и $a_0a_1a_2a_3a_4$0 соответственно, где $a_0...a_4$ – двоичные цифры кода адреса, совпадающие в одноименных разрядах обоих адресов.

Если такое размещение не удается выполнить из-за перекрестных переходов в микропрограмме, то следует продублировать в МП те операторы, к которым есть переходы от различных условных вершин.

Если в исходной МП имеются участки, содержащие проверку двух логических условий подряд, то для их реализации в макете необходимо использовать две микрокоманды с условным переходом, так как в каждой МК можно проверить лишь одно условие. В этом случае вторая МК не содержит управляющих сигналов в операционной части и выполняет только проверку второго из условий.

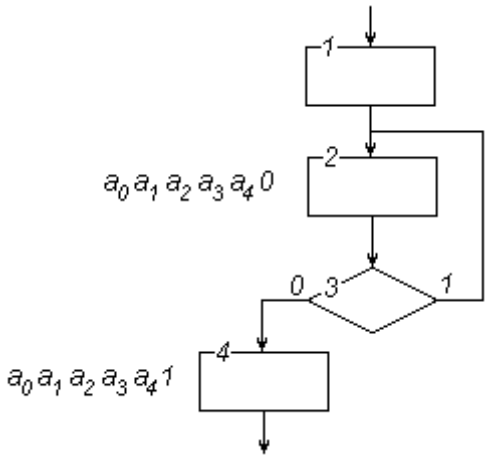


Рис. Л2.4. Фрагмент граф-схемы микропрограммы

Таким образом, в данном случае целесообразно придерживаться приводимых ниже рекомендаций.

1. Адреса микрокоманд удобно (но не обязательно) начинать с нулевого.

2. Первоначально каждая микрокоманда должна содержать не более одной проверки одного условия.

3. Назначать адреса микрокомандам можно произвольно за исключением микрокоманд, следующих за проверкой условия.

3а. Для пары микрокоманд, следующих за проверкой условия, адреса должны совпадать во всех разрядах, кроме младшего. Младший разряд адреса должен быть равен значению логического условия, записанному на том выходе условной вершины, с которым связана данная микрокоманда.

4. При наличии в формате микрокоманды нескольких полей условий, следует использовать самое правое из них (так как именно оно управляет последним разрядом адреса следующей микрокоманды).

5. При наличии нескольких входов в одну и ту же вершину (операторную или условную) микропрограммы из разных условных вершин эту вершину приходится дублировать, полностью копируя ее содержимое.

6. Последнюю микрокоманду удобно “зациклить” на начало микропрограммы.

П. 6 выполняется аналогично п. 4. Отличие состоит в том, что при выполнении п. 6 следует проверить правильность формирования последовательности управляющих сигналов при различных значениях логических условий, определяющих переходы в заданной МП. Для этого, устанавливая поочередно тумблерами X_1, X_2, X_3 различные значения условий, необходимо каждый раз проследить полностью и зафиксировать для отчета последовательность вызова МК при установленном сочетании значений условий.

Выполнение циклической микропрограммы можно проследить и в автоматическом режиме, нажав кнопку переключателя "*АВТ-РУЧН*", что будет соответствовать режиму *АВТ*, т. е. автоматическому выбору МК из *ЗУМК*.

При выполнении лабораторной работы с помощью моделирующей программы ROM_BIOS.EXE или MUP.EXE используется более мощный формат МК, позволяющий проверять два или три (в зависимости от версии) условия в одной микрокоманде и имеющей три поля для кодирования микроопераций. В этом случае при записи микропрограмм следует использовать третье поле условия "ЛУ3" (или "НУ2"), а сигналы микроопераций заносить в два поля: "МО1" и "МО2". Кодировка всех полей в программе трехразрядная.

6. Содержание отчета

1. Структурная схема лабораторного макета с описанием назначения его основных узлов.

2. Граф-схема заданной линейной микропрограммы и эта же МП, представленная в таблице по форме 1.

3. Граф-схема циклической микропрограммы и ее запись в таблице по форме 1.

4. Последовательность управляющих сигналов, вырабатываемых при различных значениях логических условий, и оценка среднего времени выполнения микропрограммы в тактах.

Вопросы для самопроверки

1. Опишите назначение микропрограммных УУ и принцип их действия.

2. Поясните назначение основных блоков лабораторного макета.

3. Какое максимальное количество МК может содержать представимая в макете МП?

4. Какое максимальное количество МО может содержать МК?
Операционное устройство, управляемое данным УУ?

5. Что происходит при нажатии кнопки *"ПУСК"* в режиме *"РУЧН"*? При отпускании?

Литература: [1], с. 302...324

Лабораторная работа № 3

ОПРЕДЕЛЕНИЕ КОНФИГУРАЦИИ И ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ ПЭВМ

1. Цель работы

Цель работы состоит в практическом ознакомлении со структурой персональной ЭВМ и методами определения ее конфигурации и параметров, а также оценки производительности и тестирования персональных ЭВМ и их компонент.

2. Основные сведения

Диагностические и тестовые программы используются для определения состава и характеристик ПЭВМ и отдельных их устройств, проверки работоспособности и производительности ПЭВМ и ее компонент.

Эти программы строятся с использованием непосредственного обращения к аппаратным ресурсам ПЭВМ и в основном исполняются под управлением DOS. Запуск многих из таких программ под Windows может привести к получению искаженных результатов. Но существуют и специальные Windows версии тестовых программ.

Оценка производительности, как правило, производится в диагностических программах либо в относительном виде (проценты, индекс), либо в виде некоторого численного показателя, определяемого для конкретной смеси заданий (например, WhetStone, DhryStone, WinBench99 и пр.). Результаты таких измерений часто имеют вероятностный характер и для получения усредненного значения должны производиться многократно.

Персональные ЭВМ типа IBM PC совместимых имеют явно выраженную модульную структуру, что позволяет собирать из базовых модулей конфигурацию, соответствующую потребностям и возможностям пользователя.

Состав ПЭВМ, различаясь для конкретных экземпляров, имеет базовые компоненты, обязательные для любой модификации. Обычно в любой ПЭВМ имеются следующие узлы:

- процессор;
- материнская (системная) плата;
- оперативная память;
- видеоадаптер;
- жесткий диск;
- гибкий диск;

- корпус системного блока с блоком питания;
- монитор;
- клавиатура;
- мышь.

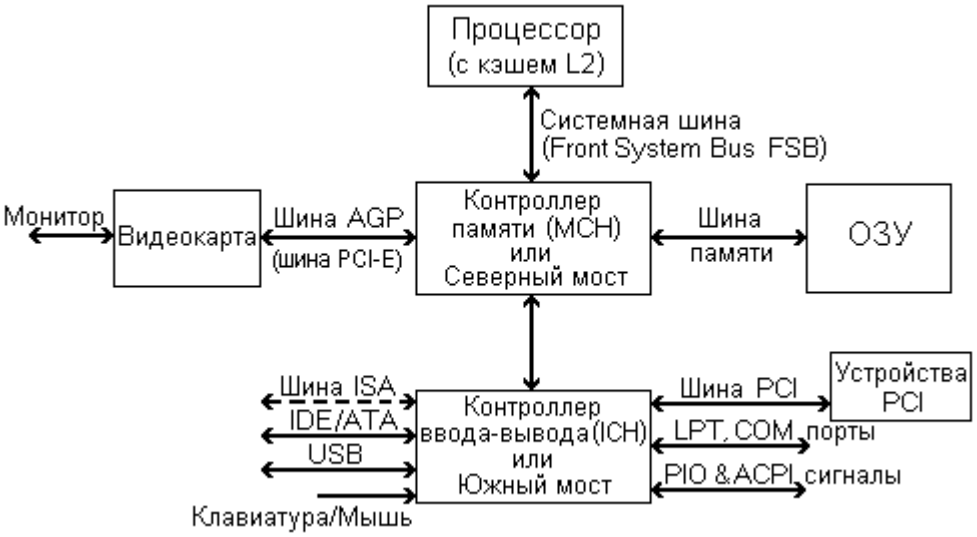


Рис. Л3.1. Обобщенная структурная схема персональной ЭВМ

В самом общем виде структура ПЭВМ может быть представлена так, как показано на рис. Л3.1, где ОЗУ – оперативная память, а сокращения MCH и ICH у контроллеров памяти и ввода-вывода означают *Memory Control Hub* и *Input-Output Control Hub* соответственно.

Такое представление, конечно, скрывает особенности организации системной шины и способов подключения основных устройств. Данные способы связаны с организацией системной шины и дополнительных интерфейсов и могут различаться в зависимости от поколения ПЭВМ, типа процессора и комплекта микросхем (так называемого чипсета) материнской платы.

Эти различия, главным образом, связаны с теми системными интерфейсами, которые поддерживает материнская плата. Известны следующие виды локальных шин ЭВМ, использующихся для подключения внешних устройств ПЭВМ:

Название шины	Разрядность данных (биты)	Частота шины (МГц)	Пропускная способность (Мб/с)	Разрядность адреса (биты)
ISA	8	8	8	20
EISA	32	8,33	33,3	32
VLB	32/64	33-50	132	32
PCI	32/64	33/66	132/528	32
AGP	32	66 (x1, x2, x4)	256/512/1024	32
PCI-E	1 (передача) 8/16 (вх/вых)	2500	500	-

Шина ISA (Industry Standard Architecture) использовалась в ПЭВМ, начиная с моделей с процессором 8086 и до Pentium II, в последующих моделях она уже исключена. Шина EISA представляет расширенную модификацию ISA. Шина VLB (VESA Local Bus) использовалась только в процессорах 486. Шина PCI (*Peripheral Component Interconnect*) является наиболее распространенным вариантом, иногда сочетаемым с шиной ISA для аппаратной совместимости с более ранними устройствами. Для подключения видеоадаптера используют и более быстрый вариант – AGP (*Advanced Graphical Port*), а в более новых ПЭВМ – и шину PCI-Express.

Наличие этих шин и интерфейсов в ПЭВМ обеспечивают контроллеры, называемые также мостами или хабами (hub), связывающие системную шину с соответствующей локальной шиной (например, системная шина – шина PCI), или различные интерфейсы (например, шина PCI – шина ISA).

Эти контроллеры могут интегрировать в себе узлы, которые обеспечивают и подключение жестких дисков. Двумя наиболее распространенными вариантами управления жестким диском являются интерфейсы ATA (*AT Attachment for Disk Drives* – подключение дисководов к PC AT), точнее, его модификации ATAPI (*ATA Package Interface*) и *Serial ATA*, а также SCSI (*Small Computer System Interface*).

С интерфейсом ATA связаны еще два названия: IDE (*Integrated Device Electronics*), указывающее на особенности организации контроллера жесткого диска, и DMA (*Direct Memory Access*) или его более новая модификация Ultra DMA, определяющие режим обмена данными с оперативной памятью. (Другим режимом обмена, используемым в этом интерфейсе, является программный ввод-вывод – PIO, чаще использовавшийся для подключения CD ROM.) Интерфейс ATA первоначально предназначался для подключения жестких дисков к шине ISA. Однако, при наличии шины PCI организуется связь IDE диска с данной шиной. Интерфейс ATA обеспечивает передачу данных со скоростью до 133 Мбайт/с, а его последовательный вариант – до 150 Мбайт/с

Интерфейс SCSI обеспечивает скорости передачи данных до 160 Мбайт/с, и также может использоваться не только для подключения жестких дисков, но и других устройств. Однако он требует дополнительных контроллеров и является более дорогим вариантом.

Внешние устройства типа клавиатуры, принтера, гибкого диска, мышки также требуют контроллеров для своего подключения. Обычно эти контроллеры интегрированы в единый узел, который и обеспечивает их связь с системной шиной.

Собственно системный контроллер организует связь процессора с оперативной памятью, внешним кэшем (если таковой имеется), шиной PCI, выполняет функции контроллера оперативной памяти, контроллера кэша и контроллера прерываний.

Определить конфигурацию ПЭВМ в общем виде можно в момент загрузки после включения или при перезагрузке по данным, высвечиваемым на

экране в процессе тестирования машины базовой системой ввода-вывода BIOS. BIOS начинает исполняться при включении или перезагрузке ПЭВМ и после вывода своего типа и версии, а также типа материнской платы обычно высвечивает следующую информацию:

тип процессора (CPU *Type*),

наличие сопроцессора (*Co-Processor*),

частоту (ядра) процессора (CPU *Clock*),

объем основной части оперативной памяти (*Base Memory*) – во всех ПЭВМ сейчас это 640 Кбайт,

объем расширенной оперативной памяти (*Extended Memory*),

объем кэш-памяти (*Cache Memory*),

параметры гибкого дисковода A (*Diskette Drive*),

параметры гибкого дисковода B (*Diskette Drive*),

параметры основного диска 1-го канала IDE (*Primary Master*),

параметры вспомогательного диска 1-го канала IDE (*Primary Slave*),

параметры основного диска 2-го канала IDE (*Secondary Master*),

параметры вспомогательного диска 2-го канала IDE (*Secondary Slave*),

а также вид дисплея, установки портов ввода-вывода, тип оперативной памяти и кэш-памяти.

Методы оценки производительности ЭВМ подразделяются в зависимости от ряда факторов (задач и целей оценки, готовности оцениваемой ЭВМ и пр.) на экспериментальные и теоретические.

Первая группа предполагает непосредственное измерение времени выполнения на ЭВМ каких-либо вычислений. Результаты этих измерений представляются, как правило, каким-либо индексом, соотносящим производительность оцениваемой системы с производительностью некоторой базовой модели. Такие индексы обычно используются для оценки систем с близкой или однотипной архитектурой (например, 32-разрядной архитектурой процессоров Intel) и достаточно часто меняются в связи с быстрыми изменениями технологии и ростом производительности новых процессоров. Например, известная программа CheckIt измеряет производительность относительно базовой модели ПЭВМ PC XT на процессоре Intel 8086, давно вышедшей из употребления.

Теоретические методы предполагают применение той или иной математической модели ЭВМ.

3. Программа работы

1. Включить ЭВМ и зафиксировать основные особенности конфигурации и параметры ЭВМ, высвечиваемые при загрузке:

процессор: тип, частота, изготовитель;

память: емкость, тип;

кэш: емкость, тип;

жесткие диски: марка, объем, серийный номер;

количество и адреса портов ввода-вывода.

Внимание! Пункты 2 – 7 и 15 выполняются по указанию преподавателя.

2. Загрузить ПЭВМ в режиме MS-DOS.

3. Вызвать на исполнение программу *PCConfig* (*Informer*). Определить с ее помощью и записать в таблицы, формы которых приведены в указаниях к выполнению работы:

а) общий состав и основные характеристики аппаратных средств ПЭВМ (первое окно программы);

б) основной состав системного программного обеспечения (второе окно);

в) характеристики производительности процессора, видеотракта и дисков;

г) чипсет системной платы;

д) состав резидентных программ;

е) загруженные драйверы (устройства);

ж) распределение адресного пространства по различным видам памяти и относительные скорости этих устройств;

з) закрепление векторов прерывания за аппаратными запросами прерываний IRQ и модули программного обеспечения, обслуживающие эти прерывания;

и) логическую структуру дисковой памяти.

Подпункты б), д) и е) выполняются по указанию преподавателя.

4. Зафиксировать результаты анализа в таблицах, формы которых приведены ниже, в указаниях к выполнению работы.

5. Запустить программу *System Speed Test* 4.61, получить с ее помощью оценки рейтинга процессора и параметров диска и оперативной памяти. Зафиксировать результаты средствами программы (в текстовой форме) и сравнить их с результатами, полученными с помощью *PCConfig*.

6. Запустить на исполнение программу *CheckIt*. Просмотреть первые три пункта главного меню (*SysInfo*, *Tests* и *BenchMark*). Сравнить полученные результаты с данными построенной таблицы. Записать отклонения.

7. Вызвать программу *PartInfo*. Сравнить данные о CHS-геометрии жесткого диска, полученные при выполнении предыдущих пунктов, с данными программы *PartInfo*. Сопоставить данные о разделах диска с их описанием в BOOT-секторе диска.

8. С помощью DOS-программы DEBUG прочитать содержимое RTC CMOS памяти и записать форму представления даты, времени, параметров жесткого диска, памяти и место их расположения в CMOS.

9. Загрузить ПЭВМ в MS-Windows (если ранее была загрузка DOS).

10. Вызвать на исполнение программу *DrHardware*. Выбрать из меню тестов пункты, соответствующие данным, полученным с помощью других программ, и определить их значения с помощью *DrHardware*. (Пробную версию программы можно загрузить на сайте http://www.dr-hardware.com/).

11. Вызвать на исполнение программу SANDRA фирмы *SiSoft* и выполнить пункты (если не произойдет "зависание"):

- общие сведения о системе: *System Summary*;
- информация по системной плате: *Mainboard Information*;
- процессор и BIOS: CPU&BIOS;

- оценка производительности памяти: *Memory Benchmark*;

- оценка производительности процессора: CPU *Benchmark*;

- сравнить содержимое CMOS RTS, полученное при выполнении п. 8 с дампом CMOS-памяти: CMOS *Dump*;

- мастер настройки производительности: *Performance Tune Up Wizard*.

(Пробную версию программы можно загрузить в разделе загрузки сайта www.sisoftware.net).

12. Сравнить результаты с полученными ранее и дополнить их сведениями об особенностях параметров памяти. Ознакомиться с рекомендациями мастера настройки производительности.

13. Вызвать на исполнение программу Aida (или Everest) заполнить таблицы по формам 2, 4 и 5. Сравнить результаты с ранее полученными.

(Пробную версию программы можно загрузить в разделе загрузки сайта www.lavalys.com).

14. Ознакомиться с информацией, предоставляемой программами WCPUID (идентификация типа процессора).

15. Перезагрузить машину и войти в утилиту BIOS *Setup*. Поочередно войти в основные разделы BIOS и, **НЕ ИЗМЕНЯЯ** установленных значений параметров, зафиксировать основные настроечные параметры BIOS.

16. Используя системную утилиту DEBUG, найти и высветить первые команды обработчика прерывания, заданного преподавателем.

17. Составить отчет о проделанной работе.

4. Указания к выполнению работы

При выполнении программ следует помнить, что на некоторых процессорах и системных платах прогон программ может привести к "зависанию" системы.

При определении характеристик процессора обычно выясняются:

- изготовитель – в большинстве случаев это *Intel Corporation* – строка идентификации *GenuinIntel*, или *Advanced Micro Device Corporation* (AMD) – строка идентификации *AuthenticAMD*;
- тип, семейство, модель, степпинг – числовые и символьные коды, возвращаемые при выполнении команды идентификации процессора CPUID (в моделях, начиная с 486 процессоров и выше), указывающие на модификацию процессора:

  тип – код типа (00-11), вообще говоря, не относится к определению класса процессора и подразделяет их на версии OEM (значение 00), Overdrive (01), Dual (10);

  семейство – код разновидности процессоров 486, Pentium, Pentium II и т. д., код имеет значение 0100 – для 486 процессоров, 0101 – для Pentium, 0110 для Pentium Pro, Pentium II, III, Core, Core2, Core i7, но 1111 для Pentium 4 (начиная с последних моделей Pentium III Intel ввело код Brand ID, уточняющий идентификацию процессора);

модель – четырехразрядный двоичный код, определяющий модель в каждом семействе (например, DX2, DX4, DX5 для 486, P54, P55, P55C и др. для Pentium и т. д.);

степпинг – модификация процессора в рамках одного семейства и модели.

Диагностические программы не всегда сообщают собственно значения кодов идентификации процессора, чаще предоставляя уже выделенные из них сведения о семействе и модели процессора;

- частота – рабочая частота (ядра) процессора;
- индексы производительности – диагностические программы приводят численные значения некоторых индексов, определяющих относительную производительность ПЭВМ, причем довольно часто для сравнения даются еще и значения этих же индексов для ряда других моделей.

Ряд сведений о процессорах можно найти на сайте www.x86.org.

Данные о процессоре, полученные при выполнении лабораторной работы, рекомендуется свести в таблицу по форме 2.

Форма 2

Характеристика процессора	**Значение характеристики**		
	Aida (Everest)	DrHardware	SANDRA
Изготовитель			
Тип			
Семейство			
Модель			
Степпинг			
Частота ядра			
Частота шины			
Чипсет системной платы			
Индексы производительности			

При определении характеристик памяти следует учитывать, что:

а) из адресного пространства оперативной памяти выделяется ряд областей адресов, которые передаются другим видам памяти, главным образом BIOS, памяти видеоадаптеров, видео BIOS, а также расширенной памяти конфигурационных данных системы (ESCD – *Extended System Configuration Data*), схемам удаленной загрузки адаптеров локальных сетей, некоторым адаптерам шины ISA и др.;

б) определенные области (адресов) оперативной памяти могут выделяться для выполнения специфических функций:

- доступа к адресам за пределами первого мегабайта в реальном режиме;
- непосредственного доступа к первому сегменту за адресом 100000H для размещения в нем системных и других резидентных программ;

- косвенного доступа DOS-программ реального режима ко всей памяти через область (до четырех страниц по 16 Кбайт), используемую драйвером EMS для поочередного отображения содержимого старших адресов памяти (EMS или *expanded memory* – отображаемая память);
- размещения электронного диска;
- размещения теневых копий (кэширования в оперативной памяти) более медленных ЗУ BIOS и видео BIOS и т. п.

Обычно выделяемые для перечисленных функций области имеют более или менее фиксированное назначение. Общая "архитектура" адресного пространства оперативной памяти выглядит следующим образом.

Адреса	**Область**	**Размер**
00000000h-0009FFFFh	Стандартная (базовая) память (Conventional (Base) Memory)	640 Кбайт
000A0000h-000FFFFFh включая 000F0000h-000FFFFFh	Верхняя память (Upper Memory Area - UMA) BIOS	384 Кбайт 64 Кбайт
00100000h-(F)FFFFFFFFh включая 000F0000h-000FFFFFh	Дополнительная (расширенная) память (Extended Memory) Дополнительная верхняя память (High Memory Area)	до 4 Гбайт (Pentium II и выше – 64 Гбайт) 64 Кбайт – 16 байт

Стандартная память непосредственно доступна DOS и программам реального режима.

Верхняя память используется для системных целей: в ней размещаются области буферной памяти видеоадаптеров (видеопамять) и постоянная память (BIOS с расширениями).

Дополнительная память непосредственно доступна только в защищенном режиме. Однако в ней имеется небольшая область (см. выше) дополнительной верхней памяти, доступной и в реальном режиме при открытом вентиле A20 (21-й разряд шины адреса). Эту область драйвер HYMEM.SYS делает доступной для размещения ядра DOS с целью экономии стандартной (*conventional*) памяти.

В самых старших адресах памяти размещается область ПЗУ BIOS: для ПЭВМ с 24-разрядной шиной адреса – это область FE0000h-FFFFFFh размером 128 Кбайт, а для ПЭВМ (386, 486, Pentium) с 32-разрядным адресом – FFFE0000h-FFFFFFFFh и с 36-разрядным адресом (Pentium II, III) – FFFFE0000-FFFFFFFFFh.

Стандартная память имеет несколько фиксированных областей:

- 00000h - 003FFh – область векторов прерывания (256 двойных слов);
- 00400h - 004FFh – область данных BIOS (256 байтов);
- 00500h - 00xxxh – область данных DOS (до 2800 байтов),

остальная память предоставляется пользователю.

Верхняя память (000A0000h-000FFFFFh) стандартно распределяется следующим образом:

- A0000h-BFFFFh – видеопамять (128 Кбайт - Video RAM);
- C0000h-DFFFFh – резерв для BIOS различных адаптеров

  (Дополнительный BIOS видеоадаптера имеет фиксированный адрес C0000h и инициализируется на шаге инициализации видеоадаптера. Платы адаптеров, установленные в слоты системной платы, могут иметь свои ПЗУ для программной поддержки – дополнительные модули ROM BIOS (additional ROM BIOS). Их используют некоторые контроллеры жестких дисков, сетевые адаптеры с удаленной загрузкой и другие периферийные устройства. Для этих модулей в адресном пространстве зарезервирована область C8000h-F4000h. При загрузке ПЭВМ эта область сканируется с шагом 2 Кбайт в поисках дополнительных модулей BIOS);
- E0000h-EFFFFh – свободная область 64 Кбайт, иногда занятая под системный BIOS;
- F0000h-FFFFFh – системный BIOS: 64 Кбайт ПЗУ (или флэш) область на системной плате.

При наличии системы *Plug&Play* в области системного BIOS адреса FD000h-FDFFFh отданы энергонезависимой конфигурационной памяти (*Extended System Configuration Data* – ESCD).

Доступ к стандартной памяти конфигурации оборудования и часам реального времени CMOS RTC осуществляется через порты ввода-вывода с адресами 70h (смещение-адрес) и 71h (данные).

Данные о памяти, полученные при выполнении лабораторной работы, рекомендуется занести в карту адресного пространства памяти по форме 3, а характеристики кэш-памяти и оперативной памяти таблицу по форме 4.

В карте адресного пространства памяти не следует повторять данные, полученные с помощью разных программ, а привести общие значения, которые должны совпасть.

Форма 3

Область памяти	Диапазон адресов области памяти	Объем области памяти	Относительная производительность

При определении характеристик жесткого диска его тип, объем и производителя можно найти непосредственно по данным, выводимым на экран тестом POST BIOS при загрузке машины.

CHS-организация (иначе говорят, CHS-геометрия) определяет количество цилиндров (*Cylinders*), головок (*Heads*) и секторов (*Sectors*) на дорожках (*Tracks*) диска. Емкость диска определяется как произведение общего количества секторов (C*H*S) на размер сектора, который для жестких дисков равен 512 байтам.

При этом следует учитывать, что логическая адресация диска может не совпадать непосредственно с его физической геометрией. Эта ситуация возникла при появлении дисков большого объема из-за необходимости согласования форматов адресов, воспринимаемых контроллерами жестких дисков и функциями DOS (BIOS), обслуживающими обращения к жестким дискам.

Форма 4

Характеристика оперативной или кэш-памяти	**Значение характеристики**		
	Aida (Everest)	DrHardware	SANDRA
Объем кэша L1			
Скорость передачи кэша L1			
Объем кэша L2			
Скорость передачи кэша L2			
Объем ОП			
Тип ОП			
Скорость передачи ОП			

Так, функция дискового сервиса BIOS INT 13h использует для обращения к диску:

- 10-разрядный номер цилиндра (биты [7:6] регистра CL и восемь битов регистра CH),
- 8-разрядный номер головки (регистр DH),
- 6-разрядный номер сектора (биты [5:0] регистра CL).

Это позволяет адресовать диски объемом до 2 в 24-й степени секторов, или до 8 Гбайт.

Однако контроллер интерфейса ATA (IDE) для адресации диска позволяет использовать только 4-разрядный номер головки, хотя и имеет:

- 16-разрядный номер цилиндра (два однобайтовых регистра контроллера),
- 4-разрядный номер головки (половина регистра номера головки/номера диска)
- 6-разрядный номер сектора.

Это позволяет адресовать диски объемом до 2 в 26-й степени секторов, или до 31 Гбайт. Но результирующая емкость диска из этих совместных ограничений получается всего 2 в 20-й степени (1024 цилиндра, 16 головок, 63 сектора на дорожке) секторов, или до 504 Мбайт.

Для преодоления этих ограничений в BIOS приходится преобразовывать адреса обращений к диску по некоторым правилам, что отражается в опциях BIOS, устанавливаемых для диска в режиме адресации:

- NORMAL – обычная CHS-геометрия;
- LARGE – или ECHS (*Extended* CHS) – расширенная CHS-геометрия;
- LBA – логическая адресация блоков.

Способов преобразования адреса для режима ECHS может быть несколько, поэтому использование данного режима может привести к тому, что на других машинах ECHS диск не будет читаться или будет читаться неверно.

Более распространенным является режим LBA, при котором регистры контроллера используют линейную нумерацию секторов и позволяют задать 28-разрядный адрес сектора. Этот режим обеспечивает работу с дисками объемом до 128 Гбайт.

Следует учитывать, что еще одно ограничение на размер логических дисков может накладывать файловая система операционной системы. Так, FAT16 не позволяет работать с дисками более 2 Гбайт.

Геометрия диска также определяется BIOS и сообщается в момент загрузки вместе с маркой жесткого диска.

Кроме того, BIOS также сообщает и о режиме передачи данных, реализуемом контроллером диска (встроенным и на системной плате):
PIO или UDMA (для ATA IDE дисков) с указанием номера режима, определяющего скорость его передачи.

Во всех используемых в лабораторной работе диагностических программах предусмотрена возможность определения временных характеристик жесткого диска. (Кстати, ПЭВМ может иметь до двух физических жестких дисков, или даже больше при наличии RAID-контроллера.) Обычно это скорость вращения диска, время поиска информации и скорость передачи данных.

Эти программы также определяют и разбиение диска на логические диски, их размер и количество свободного места на них.

Разбиение диска на разделы и логические диски отражено в загрузочном (первом), или BOOT-секторе диска (еще одно название содержимого этого сектора – MBR: *Master Boot Record* – главная загрузочная запись), где, начиная с байта 1BEh, размещаются четыре 16-байтных записи о разделах (*partitions*), на которые разбит диск, а также в первом секторе каждого раздела, где, начиная с байта 1BEh, размещаются четыре 16-байтных записи о логических дисках, на которые разбит раздел. Формат записи о разделе следующий:

```
Байт 0     - флаг загрузочного (активного) раздела - Boot Flag
             (80h - раздел активный (загрузочный), 0 - нет)
Байт 1     - номер начальной головки раздела - Begin Head
Байты 2,3  - номер начального сектора и цилиндра раздела в формате
             обращения через BIOS Int 13h  (см. выше)
Байт 4     - код (файловой) системы - System Code
             (до 80 значений: 06 - первичный раздел DOS, FAT16
                              0E - расширенный раздел DOS, FAT16
                              0B - первичный раздел DOS, FAT32
                              0F - расширенный раздел DOS, FAT32
```

```
                                07 - раздел WINDOWS NT, NTFS и т.д.)
Байт 5      - номер конечной головки раздела - End Head
Байты 2,3   - номер конечного сектора и цилиндра раздела в формате
              обращения через BIOS Int 13h  (см. выше)
Байты 8-11 - относительный (линейный) номер начального сектора раздела
Байты 12-15- количество секторов в разделе.
```

Содержимое записей о разделах можно просмотреть либо с помощью утилиты DISKEDIT, либо с помощью специальных утилит, например *Partition Magic* или *PartInfo*, представляющих эти записи в более удобной для чтения форме.

Результаты анализа жесткого диска следует свести в таблицу по форме 5, строки должны описывать следующие характеристики:

Форма 5

Характеристика жесткого диска	**Значение характеристики**		
	Aida (Everest)	DrHard	SANDRA
Изготовитель			
Тип			
Объем			
CHS-организация			
Логическая геометрия (CHS, ECHS, LBA)			
Режим обмена			
Скорость вращения шпинделя			
Время поиска			
Скорость передачи			
Логические диски			
Структура записи о разделах в BOOT-секторе			

5. Содержание отчета

1. Перечень программ и номера их версий (или даты), использованных для тестирования ПЭВМ. Краткая характеристика (назначение, функциональные возможности) каждой программы.

2. Результаты тестирования ПЭВМ, полученные с помощью диагностических программ и представленные в таблицах по формам $2 \div 5$.

3. Перечень основных разделов BIOS Setup и список параметров, устанавливаемых в каждом из разделов.

4. Схема подключения CMOS в адресном пространстве ввода-вывода и адреса параметров, прочитанных из CMOS.

5. Описание порядка вызова обработчика аппаратного прерывания и начальные команды обработчика для заданного типа прерывания.

3.4. Методические указания к проведению практических занятий

По дисциплине предусмотрено выполнение четырех практических заданий. Варианты заданий выбираются в соответствии с цифрами шифра студента, причем вариант микропрограммы для второго задания выбирается по предпоследней цифре шифра. (Задание 5 выполняется по указанию преподавателя студентами очных форм обучения.)

Практические задания должны оформляться в отдельной тетради (файле) с указанием названия дисциплины, фамилии, инициалов и шифра студента, группы, факультета и специальности.

3.4.1. Задания на практические занятия

Задание 1

Представить граф-схему микропрограммы, заданной в виде логической схемы, и определить среднее время выполнения микропрограммы, предполагая, что проверка логических условий не требует времени. Время выполнения операторных вершин и вероятности выполнения логических условий x_i (т. е. вероятности того, что $x_i = 1$) микропрограмм представлены в табл. П1.

Вычислить также значение вероятности выполнения условия x_i, определяющего циклический участок микропрограммы (если таких условий несколько, то для одного из них), при котором цикл будет выполнен в среднем $m+n+5$ раз, где m – последняя, а n – предпоследняя цифры шифра.

Таблица П.1

Посл. цифра шифра	Время выполнения оператора в тактах						Вероятность выполнения условия			
	A_1	A_2	A_3	A_4	A_5	A_6	x_1	x_2	x_3	x_4
0	2	5	7	4	3	2	0,5	0,5	0,6	0,2
1	3	8	6	6	1	5	0,2	0,4	0,9	0,8
2	1	4	3	7	3	2	0,1	0,9	0,5	0,4
3	4	2	4	3	2	6	0,7	0,6	0,2	0,8
4	2	3	7	8	6	5	0,4	0,2	0,4	0,9
5	3	6	4	2	3	1	0,5	0,1	0,1	0,6
6	2	8	9	1	10	6	0,9	0,3	0,2	0,7
7	1	1	5	7	9	8	0,9	0,4	0,6	0,5
8	4	2	9	3	2	4	0,6	0,6	0,7	0,9
9	3	1	7	6	8	5	0,5	0,3	0,2	0,9

Варианты микропрограммы

Вариант 0

$A_н\, A_1\, x_1 \uparrow {}^1A_4 \downarrow {}^6\, x_4 \uparrow {}^2A_6\, \omega \uparrow {}^3 \downarrow {}^1\, x_2 \uparrow {}^4 \downarrow {}^5A_3 x_3 \uparrow {}^5 \downarrow {}^4A_2\, \omega \uparrow {}^6 \downarrow {}^2A_5 \downarrow {}^3A_к$

Вариант 1

$A_н\, A_1\, x_1 \uparrow {}^1A_3 x_3 \uparrow {}^2\, x_4 \uparrow {}^3A_6 \omega \uparrow {}^4 \downarrow {}^3A_5 \omega \uparrow {}^5 \downarrow {}^2 \downarrow {}^8\, x_2 \uparrow {}^6A_4 \omega \uparrow {}^7 \downarrow {}^6 \downarrow {}^1A_2 \omega \uparrow {}^8 \downarrow {}^4 \downarrow {}^5 \downarrow {}^7A_к$

Вариант 2

$A_н\, A_1 \downarrow {}^1A_2 x_1 \uparrow {}^1\, x_2 \uparrow {}^2\, x_4 \uparrow {}^3A_6\, \omega \uparrow {}^4 \downarrow {}^3 \downarrow {}^6A_5\, \omega \uparrow {}^5 \downarrow {}^2A_3 x_3 \uparrow {}^6\, A_4 \downarrow {}^4 \downarrow {}^5A_к$

Вариант 3

$A_н \downarrow {}^3A_1 \downarrow {}^2A_2 \downarrow {}^1A_3 x_1 \uparrow {}^1\, x_2 \uparrow {}^2\, A_4 x_3 \uparrow {}^3 x_4 \uparrow {}^4A_6\, \omega \uparrow {}^5 \downarrow {}^4A_5 \downarrow {}^5A_к$

Вариант 4

$A_н\, A_1 \downarrow {}^2x_1 \uparrow {}^1A_2 \downarrow {}^1\, x_2 \uparrow {}^2A_5\, x_4 \uparrow {}^3A_6\, \omega \uparrow {}^4 \downarrow {}^3\, x_3 \uparrow {}^5A_4\, \omega \uparrow {}^6 \downarrow {}^5A_3 \downarrow {}^4 \downarrow {}^6A_к$

Вариант 5

$A_н\, A_1\, x_1 \uparrow {}^1 \downarrow {}^4A_2 \downarrow {}^3 \downarrow {}^1A_3 \downarrow {}^2A_4 x_2 \uparrow {}^2\, x_3 \uparrow {}^3A_5\, x_4 \uparrow {}^4A_6 A_к$

Вариант 6

$A_н\, A_1 \downarrow {}^4A_2 \downarrow {}^3A_3 \downarrow {}^2A_4 \downarrow {}^1A_5\, x_1 \uparrow {}^1x_2 \uparrow {}^2\, x_3 \uparrow {}^3\, A_6\, x_4 \uparrow {}^4A_к$

Вариант 7

$A_н\, A_1 x_1 \uparrow {}^1 \downarrow {}^4A_3\, x_3 \uparrow {}^2A_6\, \omega \uparrow {}^3 \downarrow {}^2 \downarrow {}^6\, x_4 \uparrow {}^4A_5\, \omega \uparrow {}^5 \downarrow {}^1A_2\, x_2 \uparrow {}^6A_4 \downarrow {}^3 \downarrow {}^5A_к$

Вариант 8

$A_н\, A_1 x_1 \uparrow {}^1A_2 \downarrow {}^5A_3 \downarrow {}^3A_4 \downarrow {}^4x_4 \uparrow {}^2A_5 \downarrow {}^6\, x_3 \uparrow {}^3\, \omega \uparrow {}^4 \downarrow {}^1\, x_2 \uparrow {}^5\, \omega \uparrow {}^6 \downarrow {}^2A_к$

Вариант 9

$A_н\, A_1\, x_1 \uparrow {}^1 \downarrow {}^3A_4\, x_4 \uparrow {}^2A_6 \omega \uparrow {}^3 \downarrow {}^2 \downarrow {}^6A_5 \omega \uparrow {}^4 \downarrow {}^1 \downarrow {}^7 \downarrow {}^8\, x_2 \uparrow {}^5A_3 x_3 \uparrow {}^6 \omega \uparrow {}^7 \downarrow {}^5A_2\, \omega \uparrow {}^8 \downarrow {}^4A_к$

Задание 2

Построить модуль оперативного запоминающего устройства, имеющий заданную информационную емкость, на микросхемах памяти заданной серии. Емкость модуля и серия микросхем выбираются из табл. П.2.

Таблица П.2

Предпоследняя цифра шифра	Емкость модуля (Кбайт)	Серия микросхем
0	64	К132
1	64	К537
2	64	К1500

Окончание таблицы П.2

3	128	К537
4	128	К541
5	128	К1500
6	256	К132
7	256	К537
8	256	К541
9	512	К132

Задание 3

Построить в заданном элементном базисе функциональную схему одного разряда операционной части устройства с магистральной структурой или с непосредственными связями, состоящего из четырех регистров: *P*1, *P*2, *P*3 и *P*4, на синхронных *D*-триггерах (можно использовать также и *DV*-триггеры) и допускающего заданные в табл. П.3 передачи информации между регистрами.

Таблица П.3

Предпоследняя цифра шифра	Перечень передач	Последняя цифра шифра	Тип связей устройства	Элементный базис
0	$A_1, A_2, A_5, A_6, A_7, A_9$	0	М	ИЛИ-НЕ
1	$A_1, A_4, A_7, A_9, A_{11}, A_{12}$	1	Н	"
2	$A_2, A_5, A_8, A_9, A_{10}, A_{12}$	2	М	"
3	$A_2, A_6, A_7, A_9, A_{10}, A_{11}$	3	Н	"
4	$A_2, A_3, A_4, A_7, A_8, A_{11}$	4	М	"
5	$A_3, A_4, A_5, A_7, A_{10}, A_{12}$	5	Н	И-НЕ
6	$A_3, A_5, A_6, A_8, A_9, A_{12}$	6	М	"
7	$A_1, A_2, A_3, A_5, A_7, A_{10}$	7	Н	"
8	$A_1, A_4, A_5, A_7, A_8, A_{11}$	8	М	"
9	$A_2, A_5, A_6, A_9, A_{10}, A_{11}$	9	Н	"

Примечание.

A_1: $(P2) := (P1)$ A_7: $(P1) := (P3)$

A_2: $(P3) := (\overline{P1})$ A_8: $(P2) := (\overline{P3})$

A_3: $(P4) := (P1)$ A_9: $(P4) := (P3)$

A_4: $(P1) := (P2)$ A_{10}: $(P1) := (P4)$

A_5: $(P3) := (P2)$ A_{11}: $(P2) := (\overline{P4})$

A_6: $(P4) := (\overline{P2})$ A_{12}: $(P3) := (P4)$

М – устройство с магистральной структурой;

Н – устройство с непосредственными связями.

Задание 4

Составить фрагмент структурной схемы устройства управления, показать формат команд, обрабатываемых устройством, и микропрограмму одного из этапов выполнения команды. Содержание этапа команды и его особенности выбираются из табл. П.5.

Таблица П.4

Последняя цифра шифра	Содержание этапа выполнения команды	Особенности этапа
0	Выборка команды	ОП – 4 байта, команда – 6 байтов
1	Выборка команды	ОП – 2 байта, команда – 3 байта
2	Выборка команды	ОП – 8 байтов, команда – 4 байта
3	Формирование адреса	Индексная адресация
4	Формирование адреса	Адресация относительно счетчика команд
5	Формирование адреса	Косвенная регистровая адресация
6	Запись счетчика команд в стек и извлечение его из стека	Продвижение указателя стека вверх
7	Запись счетчика команд в стек и извлечение его из стека	Продвижение указателя стека вниз
8	Выборка операндов и запись результата	Двухадресная команда с прямой адресацией
9	Выборка операндов и запись результата	Трехадресная команда с прямой адресацией

Задание 5

На множестве микроопераций $Y = \{y_1, y_2, \ldots, y_{12}\}$, состоящем из 12 микроопераций, определено множество микрокоманд $W = \{w_1, w_2, \ldots, w_8\}$, состоящее из 8 микрокоманд, выполняемых некоторой ЭВМ с микропрограммным управлением.

Построить для заданных множеств Y и W матрицу S совместимости микроопераций и найти подмножества Y_i несовместимых операций методом прямого включения. Закодировать двоичными кодами микрооперации, входящие в построенные подмножества, и определить разрядность операционной части микрокоманд (суммарное количество разрядов полей микроопераций) при выбранном распределении микроопераций по подмножествам. Привести кодиро-

ванное представление операционной части двух (на выбор) микрокоманд из исходного множества *W*.

Исходное множество микрокоманд *W* определяется из табл. П.5.

Таблица П.5

а)

Третья от конца цифра шифра									
0	1	2	3	4	5	6	7	8	9
5	A	9	9	C	3	9	5	C	9
8	4	2	4	1	4	4	8	2	2
2	1	4	2	2	8	4	2	1	4
8	4	2	4	1	4	4	2	8	2
4	8	8	1	8	2	8	1	8	1
1	8	4	4	0	2	1	8	2	1
8	4	2	4	1	4	4	8	2	2
3	3	5	A	6	9	3	6	5	C
5	A	9	9	C	3	9	5	C	9

б)

Вторая от конца цифра шифра									
0	1	2	3	4	5	6	7	8	9
4	1	2	1	8	4	1	1	4	8
1	2	1	4	2	2	8	2	2	4
8	4	8	8	1	8	2	8	1	1
2	8	4	2	4	1	2	8	1	1
4	1	3	1	8	4	1	1	4	8
9	6	9	C	3	A	A	A	3	5
1	2	1	4	2	2	8	2	2	4
2	4	1	2	8	4	8	1	4	1
4	1	2	1	8	4	1	1	4	8

в)

Последняя цифра шифра									
0	1	2	3	4	5	6	7	8	9
2	8	4	2	C	1	4	4	8	2
5	2	1	4	2	6	8	2	6	4
9	7	9	C	3	A	B	A	3	D
2	8	6	2	C	1	4	5	8	2
6	8	4	3	4	5	4	4	C	2
1	3	1	4	2	2	9	2	2	C
3	A	7	6	6	3	C	7	A	6
1	2	1	5	2	2	8	2	2	4
2	8	4	2	C	1	4	4	8	2

3.4.2. Методические указания к выполнению практических занятий

При выполнении **задания 1** необходимо вначале составить граф-схему микропрограммы по ее логической схеме. Логическая схема читается слева направо от начального оператора $A_{\text{н}}$ к конечному – $A_{\text{к}}$. Операторы в этой схеме обозначены через A_i , а логические условия – через x_j . Стоящая сразу за логическим условием x_j направленная вверх стрелка с номером *k* указывает, куда осуществляется переход при нулевом значении этого условия. Оператор или условие, к которому осуществляется такой переход, расположен непосредственно справа от стрелки, имеющей тот же номер *k* и направленной вниз. При выполнении логического условия x_j (при $x_j = 1$) переход осуществляется к следующему в строке оператору или условию, записанному справа от x_j , а все стрелки игнорируются. Наконец, ω означает тождественно ложное условие, всегда вызывающее переход по стрелке, расположенной справа от символа ω.

Среднее время выполнения микропрограммы определяется с учетом вероятностей выполнения условий переходов и времени выполнения операторных вершин. Основными расчетными формулами при этом являются формулы для подсчета времени выполнения участка с разветвлениями и циклического участка микропрограммы.

Пусть, например, задана микропрограмма вида

$$A_{\text{н}}\, x_1 \uparrow {}^1A_2 \downarrow {}^3 \uparrow {}^4A_3x_2 \uparrow {}^2\omega \uparrow {}^3 \downarrow {}^1A_1\omega \uparrow {}^4 \downarrow {}^2A_4A_{\text{к}},$$

граф-схема которой представлена на рис. П.1, где точками *a*, *b*, *c* и *d* выделены линейный участок *cd* микропрограммы, участок *ab* с разветвлениями и циклический участок *bc*. Для среднего времени $\overline{T}$ выполнения микропрограммы можно записать:

$$\overline{T} = \bar{t}_{ab} + \bar{t}_{bc} + \bar{t}_{cd} = \bar{t}_{ab} + \bar{t}_{bc} + t_4,$$

где t_4 - время выполнения оператора A_4, $\bar{t}_{kl}$ - среднее время выполнения (реализации) участка граф-схемы, начало и конец которого отмечены точками *k* и *l* соответственно.

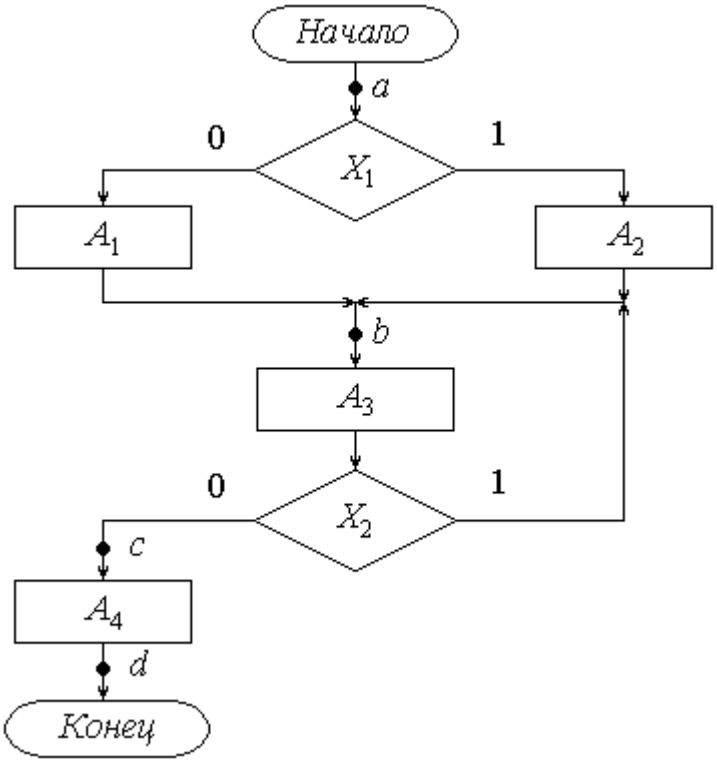


Рис П.1. Граф-схема заданной микропрограммы

Участок *ab* может быть выполнен различным образом. С вероятностью p_1 выполнения условия x_1 на этом участке будет выполнен оператор A_2, на что будет затрачено время t_2. С вероятностью $(1 - p_1)$, т. е. с вероятностью того, что $x_1 = 0$, будет выполнен оператор A_1.

Следовательно, среднее время реализации участка *ab* микропрограммы составит $\bar{t}_{bc} = p_1\, t_2 + (1 - p_1)\, t_1$, что и является расчетным соотношением для участков с разветвлениями.

Время реализации циклического участка *bc* определяется следующим образом. После выполнения оператора A_3 (время t_3) с вероятностью p_2 микропрограмма вернется в точку *b*. Следовательно, с этой вероятностью оператор A_3 будет выполнен еще раз. Далее опять проводится анализ условия x_2, и ситуация повторится. Таким образом, можно записать:

$$\bar{t}_{bc} = t_3 + p_2\,(t_3 + p_2\,(t_3 + \ldots = t_3 + p_2\, t_3 + p_2^2\, t_3 + \ldots =$$

$$= t_3 \sum_{i=0}^{i=\infty} p_2^{\,i} = t_3 / (1 - p_2)\,.$$

Это выражение и есть основное расчетное соотношение для времени выполнения циклических участков микропрограмм.

Тогда среднее время выполнения приведенной микропрограммы

$$\bar{T} = \bar{t}_{ab} + \bar{t}_{bc} + t_4 = p_1\, t_2 + (1 - p_1)\, t_1 + t_3 / (1 - p_2) + t_4\,.$$

При наличии в микропрограммах более сложных разветвлений, вложенных циклов и циклов с несколькими выходами целесообразно производить замену различных участков эквивалентными операторами, рассчитывая время их выполнения и последовательно упрощая граф-схему, а также проводить аналогичные рассуждения.

При наличии двух различных точек входа в один и тот же цикл бывает целесообразно изобразить на граф-схеме два этих цикла раздельно. Схема при этом становится более удобной и наглядной для анализа и расчета.

Типичной ошибкой при выполнении данного задания является то, что при наличии в микропрограмме нескольких разветвлений подряд (в том числе, с операторными вершинами между условными) не учитываются вероятности выполнения условий первого разветвления. Кроме того, при разбиении микропрограммы на участки непосредственное суммирование времен их выполнения допустимо лишь в том случае, если участки расположены последовательно друг за другом. Игнорирование этого правила приводит к ошибкам в записи в общем виде выражения для среднего времени выполнения микропрограммы.

В вариантах микропрограммы с двумя выходами из цикла (по двум условиям) при подсчете среднего времени выполнения такого цикла необходимо составить общее выражение, аналогичное рассмотренному выше, но с учетом обоих условий.

При выполнении **задания 2** требуется изобразить функциональную схему модуля памяти на микросхемах заданной серии. Выбор микросхем в рамках серии не ограничен. Поэтому для упрощения выполнения задания целесообразно выбирать микросхемы, имеющие наибольшую информационную емкость. Параметры микросхем и назначение их выводов можно найти в справочной литературе (например, Большие интегральные схемы запоминающих устройств: Справочник / А.Ю.Гордонов, Н.В.Бекин и др.; Под ред. А.Ю.Гордонова и

Ю.Н.Дьякова. - М.: Радио и связь, 1990. - 288 с. или Лебедев О.Н. Микросхемы памяти и их применение. - М.: Радио и связь, 1990. - 160 с.), либо загрузить по ссылке www.ord.com.ru/files/org_evm/ram132.png, …/ram537.png, …/ram541.png или …/ram1500.png в зависимости от серии микросхем, указанной в задании.

При построении модуля рекомендуется не вводить в него буферные схемы (шинные формирователи, регистры и пр.), а ограничиться только микросхемами памяти и дешифраторами. Наиболее важно понять и показать способ соединения адресных, информационных и управляющих выводов микросхем при объединении их в модуль памяти (поскольку задача носит учебный характер, для упрощения в ней использованы только микросхемы статических ЗУ ранних серий).

При изображении схемы, если количество микросхем, используемых для построения модуля, велико, следует показать начальную и конечную части схемы, а общее число используемых микросхем указать отдельно.

При выполнении **задания 3** следует изобразить триггеры регистров, не раскрывая их схемы в заданном элементном базисе.

В устройстве с магистральной структурой имеется шина данных (для одного разряда – это одна линия связи), на которую могут выдавать информацию все регистры, являющиеся для заданного набора микроопераций источниками информации. С шины данных информация может быть передана в любой регистр, являющийся приемником информации. Передача осуществляется при подаче одновременно двух сигналов управления: выдача информации из регистра-источника на шину данных и прием информации с шины на регистр-приемник.

При непосредственных связях на входе каждого регистра должна иметься схема, управляющая занесением в него информации из любого регистра, с которым в заданном наборе микроопераций есть связь, при подаче соответствующего управляющего сигнала.

Выполняя это задание, следует записать логические функции, соответствующие сигналам на управляющих и информационных входах триггеров всех регистров (в варианте с непосредственными связями) или на управляющих входах триггеров и на шине данных (в вариантах с магистральными связями). Например, $D1 = A_1(P2) + A_2(\overline{P3})$, где $D1$ – вход D-триггера регистра $P1$, ($P2$) и ($\overline{P3}$) – содержимое регистра $P2$ и инверсия содержимого регистра $P3$ соответственно. По записанным функциям далее строятся схемы связи регистров.

Наиболее распространенными ошибками при выполнении этого задания являются "закорачивание" нескольких выходов логических элементов на одну шину и использование общего сигнала синхронизации для всех регистров на D-триггерах.

Первое допустимо только для элементов со специальными выходами (какими ?), второе – приводит к сбросу триггеров, не участвующих в выполняемой передаче, так как на их информационных входах в момент подачи синхросигнала будет нулевой сигнал.

При выполнении **задания 4** для вариантов, последняя цифра которых 3÷5, целесообразно рассматривать устройство управления одноадресной ЭВМ, тогда количество отображаемых действий (стереотипных для всех адресов двух- и трехадресных команд) будет меньшим.

Выполнение задания следует начать с изображения формата команды, этап выполнения которой будет разрабатываться.

Формат операционной части (код операции – КОп) в данном задании не влияет на ход обработки команды и его можно задать произвольно, выбрав для него от 5 до 8 разрядов.

Формат адресной части команды может зависеть от способа адресации, объема адресуемой памяти и набора регистров, используемых при формировании адреса. Возможные варианты формата для различных способов адресации показаны на рис. П.2.

Количество регистров, в которых хранятся индексы или косвенные адреса, не влияет принципиально на разрабатываемые этапы, и можно ограничиться 8 или 16 регистрами, что требует 3 или 4 разряда для указания номера регистра. Смещение в вариантах с индексной адресацией можно задать 12-16-ю разрядами. Наконец, признак адресации можно задать одним-двумя разрядами, если не использовать задание способа адресации в коде операции.

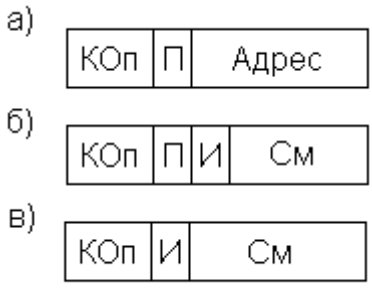


Рис. П.2. Возможные варианты формата команд

После этого необходимо проработать детали выполнения заданного этапа обработки команды. Ряд рекомендаций по отдельным вариантам задания приведен ниже. Представляя результаты выполнения задания, следует отобразить фрагмент структурной схемы устройства управления, содержащий задействованные на данном этапе узлы, перечень выполняемых в них микроопераций и микропрограмму реализации разработанного этапа.

Вид представления фрагмента структурной схемы рекомендуется сделать соответствующим рис. 3.7, а микропрограммы – соответствующим рис. 3.8 (см. тему 3.3 опорного конспекта).

Выборка команды

При выполнении вариантов задания, в которых необходимо проработать этап выборки команды следует учесть, что при длине команды, не совпадающей с разрядностью памяти, команда в памяти может располагаться различными способами по отношению к границе слова памяти. Поэтому, при выборке команды из памяти следует учитывать расположение команды.

Возможные случаи расположения команды для вариантов, последняя цифра шифра которых 0, 1 и 2, показаны на рис. П.3, а, б, и в соответственно

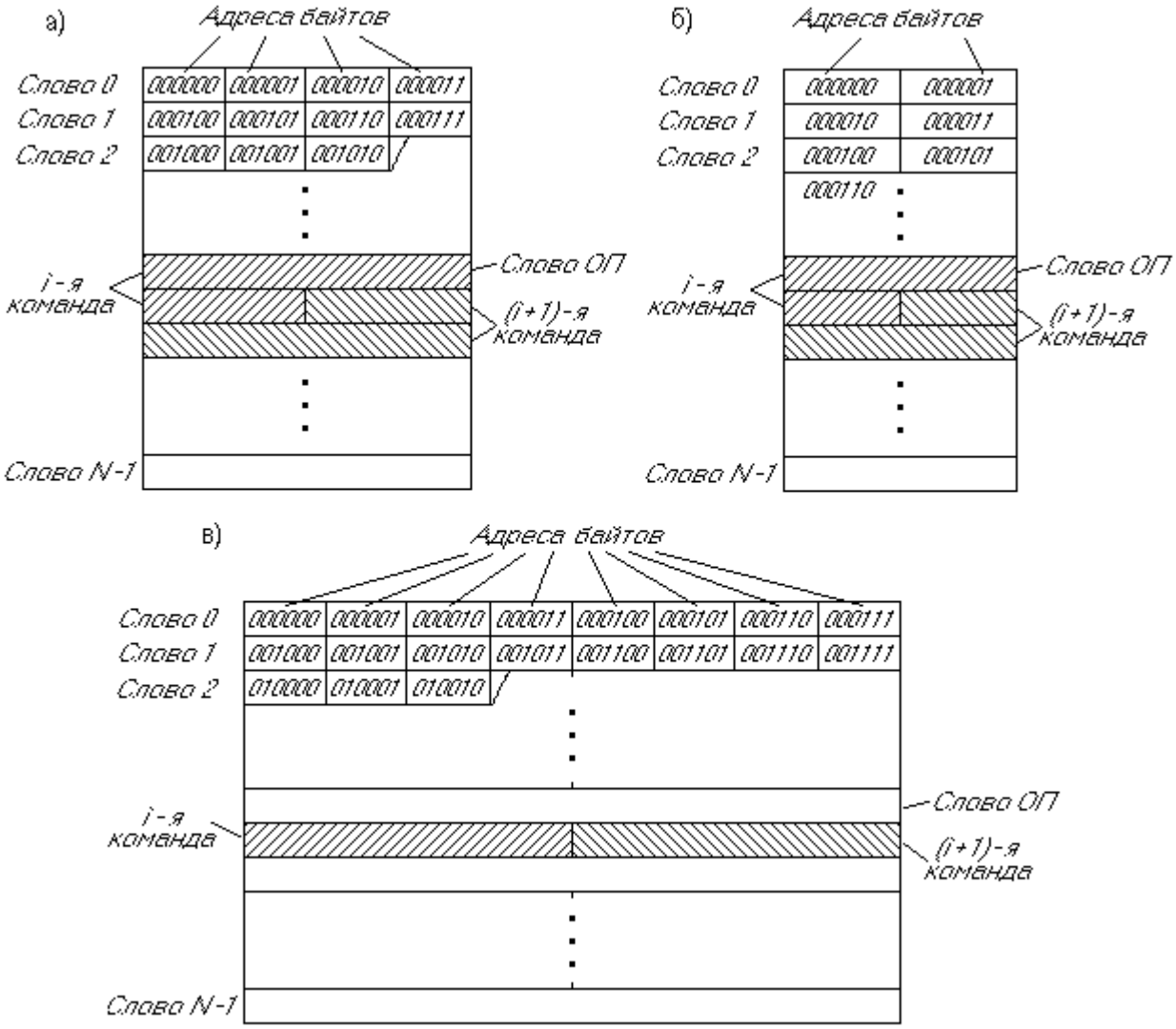


Рис П.3. Варианты расположения команд в оперативной памяти

Кроме того, во всех указанных вариантах при выборке команд, начинающихся с границы слова (*i*-я команда на каждом из рисунков), выбирается и часть (или вся – в третьем случае) следующей команды. Поскольку эта команда во всех случаях, когда текущая команда не является командой перехода, будет следующей командой, ее обычно помещают в специальный буферный регистр, чтобы не обращаться к памяти за ней снова.

Поэтому общий план выборки команды из памяти можно представить следующими шагами:

1. Анализ расположения команды по отношению к границе слова памяти. Для этого в рассматриваемых случаях достаточно проверить последние разряды счетчика команд. Адреса команд (точнее, первых их байтов), начинающихся с границы слова будут иметь на конце 2, 1 и 3 нуля для вариантов с шифрами, оканчивающимися на 0, 1 и 2 соответственно.

2-а. В случае расположения команды, начиная с границы слова, выполнить ее выборку, обращаясь к памяти. Затем в вариантах,

соответствующих последней цифре шифра 0 и 1, необходимо нарастить содержимое счетчика команд на 4 или 2 соответственно и выбрать последнюю часть команды.

При этом, если одновременно с командой или ее последней частью будет выбрано и начало следующей команды (или вся команда, как для вариантов с шифром, оканчивающимся на 2), целесообразно запомнить эту начальную часть в отдельном буферном регистре следующей команды (БРСК), из которого ее можно будет взять без обращения за ней в память.

Факт помещения начала следующей команды в буфер необходимо отметить специальным флагом действительности содержимого буфера. (Если выбранная команда является командой перехода, то в процессе ее выполнения, если адрес перехода будет отличаться от следующего за командой адреса, необходимо сбросить флаг действительности БРСК.)

2-б. В случае расположения команды с середины слова памяти следует проверить возможность использования содержимого буферного регистра и, если это возможно (о чем должен говорить флаг), взять из него начало команды (или всю команду для вариантов с последней цифрой 2). Затем увеличить содержимое счетчика команд (на 2 или 1 в вариантах 0 и 1) и, обратившись к памяти, выбрать оставшуюся часть команды.

При этом следует сбросить флаг действительности БРСК.

3. Выбирая команду (или ее часть) из памяти или буферного регистра БРСК, следует разместить ее в регистре команд.

4. По завершении выборки команды следует продвинуть счетчик команд на адрес следующей команды.

При оформлении этих вариантов задания следует изобразить фрагмент структуры УУ, включающий в себя счетчик команд, оперативную память (с регистрами адреса и данных) и регистр команд, со связями между ними и перечислением выполняемых в них микроопераций, а также привести микропрограмму выборки команды.

Формирование адресов

При выполнении вариантов задания, в которых необходимо проработать этап формирования (исполнительных) адресов, в вариантах индексной и косвенной регистровой адресации следует выбрать количество регистров, для индексной адресации выбрать разрядность смещения. Во всех случаях следует принять разрядность исполнительного адреса равной 20 двоичным разрядам (что соответствует емкости памяти в 1Мбайт).

Формат команды, как отмечалось выше, целесообразно считать одноадресным, так как в этом случае придется формировать только один адрес.

Собственно формирование адреса начинается с проверки способа адресации, соответствующего команде, обработка которой производится. Способ адресации может указываться в команде явно, специальным полем, либо косвенно определяться кодом команды. Выбор принимаемого в данном

случае способа осуществляется на этапе определения формата команды. При этом следует предположить, что система команд, к которой относится обрабатываемая команда, реализует не менее трех различных способов адресации.

Несмотря на то что в задании указан только один способ адресации, необходимо предусмотреть шаг, связанный с анализом способа адресации. В микропрограмме этот шаг следует отразить условной вершиной (или группой условных вершин), для которой будет продолжена ветвь только по одному выходу, соответствующему заданному в команде способу адресации.

В вариантах с индексной адресацией целесообразно получать адрес с помощью отдельного сумматора адреса, так как это позволяет не прорабатывать (хотя бы частично) структуру АЛУ, что пришлось бы делать при использовании сумматора АЛУ.

Доступ к регистрам в вариантах индексной и косвенной регистровой адресации можно представить на структурной схеме так, как это показано на рис. П.4. На этом рисунке регистры представлены в виде банка регистров, обращение к которому производится подачей номера требуемого регистра на вход дешифратора и при чтении – подачей управляющего сигнала *ЧтРг* чтения регистра, что приводит к появлению на шине данных регистров *ШдРг* содержимого требуемого регистра. При записи одновременно с подачей номера регистра на шину данных регистров выставляется информация, которую надо записать в регистр и подается сигнал *ЗпРг* записи в регистр.

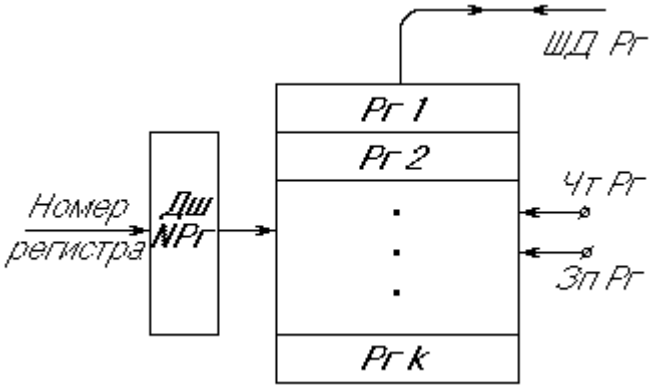


Рис П.4. Представление структуры банка регистров

Завершить этап формирования адреса следует занесением его в регистр адреса оперативной памяти.

При оформлении этих вариантов задания следует изобразить фрагмент структуры УУ, включающий в себя регистр команд, оперативную память (с регистрами адреса и данных) и адресный сумматор при индексной адресации и адресации относительно счетчика команд, а в последнем варианте и счетчик команд со связями между ними и перечислением выполняемых в них микроопераций. Также следует привести микропрограмму формирования адреса.

Запись счетчика команд в стек и извлечение его из стека

При выполнении этих вариантов задания необходимо проработать этап формирования записи в стек содержимого счетчика команд (этот этап характерен для команд вызова подпрограмм и для прерываний), а также этап извлечения из стека и установку содержимого счетчика команд (этот этап характерен для команд возврата из подпрограмм и прерываний). Допускается не выполнять проверок выхода указателя стека за допустимые пределы и считать, что анализ типа команды и формирование адреса перехода (начала подпрограммы) уже выполнены. Также можно принять произвольной разрядность адреса оперативной памяти (а значит, и счетчика команд).

Адресацию области стека в оперативной памяти целесообразно выполнять с помощью регистра – указателя стека, разрядность которого будет такой же, как и разрядность адреса. Направление продвижения указателя стека при записи (вверх или вниз – уменьшение адреса или его увеличение) можно выбрать произвольно. Соответственно при чтении информации из стека направление его продвижения будет противоположным.

Основные шаги выполнения этапа записи в стек счетчика команд сводятся к продвижению указателя стека, передаче содержащегося в нем адреса на регистр адреса оперативной памяти (ОП), передаче содержимого счетчика команд на регистр данных ОП и подаче сигнала записи в ОП. При этом указатель стека после записи будет указывать на последнее записанное слово (адрес возврата из подпрограммы).

Этап извлечения из стека содержимого счетчика команд включает в себя передачу адреса из указателя стека на регистр адреса оперативной памяти (ОП), подачу сигнала чтения из ОП, передачу извлеченного из ОП содержимого из регистра данных ОП в счетчик команд и продвижение указателя стека в обратном записи направлении.

При оформлении этого варианта задания следует изобразить фрагмент структуры УУ, включающий в себя регистр указателя стека, оперативную память (с регистрами адреса и данных) и счетчик команд, со связями между ними и перечислением выполняемых в них микроопераций. Также следует привести микропрограмму записи содержимого счетчика команд в стек и извлечения из стека ранее занесенного содержимого счетчика команд.

Поскольку названные этапы, строго говоря, относятся к разным командам, то рекомендуется включить их в общую блок схему так, как показано на рис. П.5, где *CALL* и *RET* соответственно означают команды вызова подпрограммы и возврата из подпрограммы.

Выборка операндов и запись результата

При выполнении вариантов задания, в которых необходимо проработать этап выборки операндов и записи результата, при составлении формата команд допускается использование прямой адресации, не требующей формирования

исполнительного адреса. При этом можно выбрать произвольную разрядность адресного поля, отведя, например, на каждый из адресов по 16 разрядов.

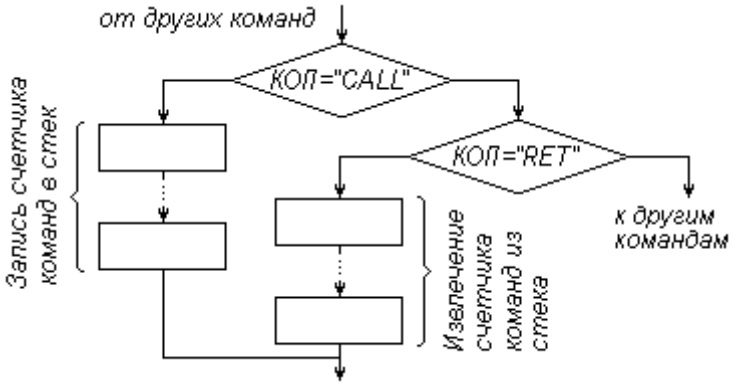


Рис П.5. Представление команд вызова подпрограммы и возврата из подпрограммы

Размещать извлеченные из памяти операнды следует в регистрах АЛУ, которые для определенности можно обозначить как регистры Р1 и Р2. Также рекомендуется предположить, что по окончании выполнения операции в АЛУ ее результат оказывается в регистре Р3.

Разрядности операндов для упрощения можно считать равными разрядности слова памяти, что позволяет извлекать их из памяти и записывать в память результат за одно обращение.

Общий порядок выполнения команды для вариантов с двухадресной командой считается следующим:

$$(A_1) * (A_2) \rightarrow (A_i),$$ где $i = 1$ или 2,

т.е., заданная операция $*$ выполняется над операндами, извлекаемыми из адресов A_1 и A_2, а результат записывается по одному из этих адресов (на усмотрение разработчика).

Команда с трехадресной адресацией выполняется по схеме

$$(A_1) * (A_2) \rightarrow (A_3).$$

При оформлении этих вариантов задания, следует изобразить фрагмент структуры УУ, включающий в себя регистр команд, оперативную память (с регистрами адреса и данных) и регистры Р1, Р2 и Р3 АЛУ со связями между ними и перечислением выполняемых в них микроопераций. Также следует привести микропрограмму выборки операндов и записи результата. Выполняемую между извлечением операндов из памяти и записью результата операцию следует обозначить на составляемой микропрограмме данного этапа одной операторной вершиной, содержание которой записать как “Выполнение операции в АЛУ”.

Примерный вид представляемых результатов может соответствовать фрагментам рис. 2 и 3 из пособия [6].

При выполнении **задания 5** исходной является матрица микрокоманд *W*, которая определяется по цифрам шифра следующим образом. Из табл. П.5 в соответствии с тремя последними цифрами шифра выбираются три столбца по восемь шестнадцатеричных цифр. Составленные вместе, эти столбцы образуют закодированную матрицу *W*. Чтобы получить окончательный вид матрицы, являющейся одной из форм задания множества *W* микрокоманд, следует представить все шестнадцатеричные цифры их двоичными эквивалентами. Тогда будет получена булева матрица размерностью 12 х 8, столбцы которой соответствуют 12 микрооперациям: y_1 ... y_{12}, а строки – 8 микрокомандам: w_1 ... w_8. Причем если в микрокоманду w_i входит микрооперация y_i, то элемент, стоящий на пересечении *j*-го столбца и *i*-й строки (т. е. элемент $w_{i\,j}$ матрицы *W*) равен единице. В противном случае этот элемент равен нулю.

Так, шифру, оканчивающемуся цифрами 978, соответствует матрица *W*:

$$W = \begin{vmatrix} 9 & 1 & 8 \\ 2 & 2 & 6 \\ 4 & 8 & 3 \\ 2 & 4 & 8 \\ 1 & 1 & C \\ 0 & A & 2 \\ 1 & 2 & A \\ C & 9 & 2 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

В этой матрице строки описывают микрокоманды, например, третья строка задает микрокоманду w_3, включающую в себя микрооперации y_2, y_5, y_{11}, y_{12}, т. е. $w_3 = \{y_2, y_5, y_{11}, y_{12}\}$.

Далее необходимо построить матрицу размерностью 12 х 12, симметричную относительно главной диагонали. Элементы s_{ij} матрицы *S* отражают совместимость микроопераций y_i и y_j. Причем $s_{ij} = 1$, если микрооперации y_i и y_j совместимы, т. е. встречаются вместе, хотя бы в одной из микрокоманд. В противном случае $s_{ij} = 0$, а микрооперации y_i и y_j считаются несовместимыми. Построить матрицу *S* можно, анализируя матрицу микрокоманд *W* с использованием формального соотношения

$$s_{ij} = \bigvee_{k=1}^{k=8} (w_{ki} \,\&\, w_{kj}),$$

т. е. просматривая попарно элементы w_{ki} и w_{kj} всех строк матрицы *W*, указывающие на то, входят ли микрооперации y_i и y_j в микрокоманду w_k.

Главную диагональ матрицы *S* можно заполнить нулями (что не является существенным для дальнейших построений). После этого следует определить подмножества несовместимых операций методом прямого включения, изложенным в [6], выполнить кодирование полученных подмножеств (микроопераций в них) и сопоставить их с полями операционной части микрокоманды. За-

тем произвести кодирование операционных частей двух микрокоманд (на выбор) из заданного исходного множества микрокоманд.

Следует также указать, что термин “совместимые” в данном тексте трактуется в смысле “совмещаемые”, т. е. используемые вместе хотя бы в одной микрокоманде, тогда как в ряде работ имеет место обратная интерпретация совместимости.

4. Блок контроля освоения дисциплины

4.1. Задание на курсовой проект

По дисциплине выполняется курсовой проект, содержание которого связано с разработкой структуры и алгоритмов функционирования простейшего процессора. Целью проекта является приобретение практических навыков в использовании полученных знаний при разработке структуры ЭВМ, закрепление основных теоретических положений курса, получение более детального представления о взаимодействии основных узлов и блоков ЭВМ в процессе обработки информации.

В проекте должны быть разработаны и представлены:

- структурная схема арифметико-логического устройства (АЛУ) и микропрограммы выполнения в нем заданных операций,
- принципиальная (или функциональная) схема АЛУ,
- структурная схема процессора и микропрограммы обработки команд.

Первые два пункта задания соответствуют части “АЛУ” курсового проекта, последний – части “Процессор”.

Вариант задания выбирается из табл. 4.1 и 4.2 по двум последним цифрам шифра. Образец оформления задания приведен ниже.

Курсовой проект оформляется в виде пояснительной записки и графической части.

Методические указания к выполнению курсового проекта изложены в [10], а также размещены на учебном сайте СЗТУ и в разделе сайта www.ord.com.ru/files/org_evm.

Образец оформления задания

ЗАДАНИЕ

на курсовой проект по дисциплине
“Организация ЭВМ и систем”

студент _____ _____
(фамилия, имя, отчество) (шифр)

Данные для проектирования

1. Организация сумматора АЛУ
2. Микропрограммы АЛУ
3. Реализация микрооперации сдвига
4. Разрядность данных АЛУ и ОП
5. Представление чисел с фиксированной запятой
6. Представление порядков чисел с плавающей запятой
7. Разрабатываемое устройство управления
8. Относительные частоты выполнения операций
9. Элементный базис
10. Разрабатываемая принципиальная схема
11. Объем адресуемой ОП (байт)
12. Способы адресации
13. Наличие адресного сумматора
14. Тип связей в устройстве управления
15. Количество адресов в команде
16. Набор команд
17. Размещение РОН
18. Количество РОН
19. Коэффициенты критерия эффективности

Задание выдано "___" _____ 200_ г.

Срок представления проекта для проверки "____" _____ 200_ г.

Студент _____
(подпись)

Руководитель проекта _____
(подпись)

Варианты задания на курсовой проект

Таблица 4.1

Предпоследняя цифра шифра	Организация сумматора АЛУ	Операции, для которых составляются микропрограммы выполнения в АЛУ	Реализация микроопераций сдвига	Разрядность данных АЛУ и ОП (байт)	Представление чисел с фиксированной запятой	Представление порядков чисел с плавающей запятой	Разрабатываемое устройство управления	Относительная частота выполнения операций				Элементный базис (серия микросхем)	Разрабатываемая принципиальная схема
								a_1	a_2	a_3	a_4		
0	Н	$С_ф$, $М2_ф$, СЕ, ИЛ	С	2	Ц	нет	П	0.3	0.3	0.2	0.2	К133 К556	БМПУ
1	К	$С_ф$, $Д_ф$, СЕ, И	С	2	Д	нет	П	0.4	0.1	0.3	0.2	К531 К556	БМПУ
2	Н	$С_ф$, $С_п$, СЛ, Л	С	4	Л	З	А	0.4	0.3	0.15	0.15	К555	АЛУ
3	К	$М_ф$, $М_п$, СА, И	С	4	Д	З	А	0.3	0.3	0.3	0.1	К555 К556	МПА АЛУ
4	Н	$Д_ф$, $Д_п$, СЛ, ИЛ	С	2	Д	Х	А	0.3	0.2	0.25	0.25	К564	АЛУ
5	К	$С_ф$, $М_ф$, СЕ, ИЛ	Л	2	Ц	нет	П	0.5	0.1	0.3	0.1	К133 К556	БМПУ
6	Н	$С_ф$, $Д_ф$, СЕ, И	Л	4	Д	нет	П	0.4	0.1	0.3	0.2	К531 К556	БМПУ
7	К	$С_ф$, $С_п$, СЛ, Л	Л	4	Л	Х	А	0.3	0.25	0.15	0.3	К555	МПА АЛУ
8	Н	$М2_ф$, $М_п$, СА, И	Л	2	Ц	Х	А	0.4	0.2	0.2	0.2	К176	АЛУ
9	К	$Д_ф$, $Д_п$, СЛ, НЕ	Л	4	Ц	З	П	0.4	0.3	0.2	0.1	К564	АЛУ

Таблица 4.2

Последняя цифра шифра	Объем адресуемой ОП (байт)	Способы адресации	Наличие адресного сумматора	Тип связей в устройстве управления	Количество адресов в команде	Набор команд	Размещение РОН	Количество РОН	Коэффициенты критерия Эффективности
									k_1 k_2 k_3 k_4 k_5
0	128К	И, АИ, К	есть	М	1	Ч, З, С, В, М, Д, (СЕ, ВЕ), И, ИЛ, СА, СЛ, УПН, БП, УЦ, БПВ	ОП	8	0,006 0,0004 0,1 0,004 0,025
1	64К	КР, Н, АИ	нет	М	2	П, С, В, М, Д, (СЕ, ВЕ), И, Л, ИЛ, СА, СЛ, ЗгР, ЗпР, БП, БПВ, УПП, УЦ	Пр	16	0,01 0,0004 0,1 0,004 0,025
2	32К	П, КР, И	есть	М	3	П, ПГ, С, В, М, Д, (СЕ, ВЕ), И, Л, ИЛ, СА, СЛ, ЗгР, ЗпР, БП, БПВ, УПО, УЦ	ПР	8	0,003 0,0004 0,5 0,004 0,025
3	512К	И, Д, АИ	нет	Н	2	П, С, В, М, Д, (СЕ, ВЕ), И, Л, ИЛ, СА, СЛ, БП, УПО, БПВ, УЦ	ОП	16	0,016 0,0005 0,2 0,008 0,04
4	256К	К, Н, И	есть	Н	3	П, ПГ, С, В, М, Д, (СЕ, ВЕ), И, Л, ИЛ, СА, СЛ, БП, УПП, УЦ	ОП	8	0,009 0,0008 0,2 0,008 0,04

Окончание таблицы 4.2

5	128К	И, Д, КР	нет	Н	1	Ч, З, С, В, М, Д, (СЕ, ВЕ), И, Л, ИЛ, СА, СЛ, ЗгР, ЗпР, БП, УПП, БПВ, УЦ	Пр	16	0,004 0,05	0,001	0,1	0,008
6	64К	И, Н, П	есть	М	2	П, С, В, М, Д, (СЕ, ВЕ), И, Л, ИЛ, СА, СЛ, ЗгР, ЗпР, БП, БПВ, УПН, УЦ	Пр	8	0,002 0,03	0,0005	0,4	0,002
7	32К	П, К, И	нет	М	3	П, ПГ, С, В, М, Д, (СЕ, ВЕ), И, Л, ИЛ, СА, СЛ, БП, БПВ, УПО, УЦ	ОП	16	0,004 0,03	0,0003	0,2	0,002
8	256К	К, И, ОС	есть	Н	2	П, С, В, М, Д, (СЕ, ВЕ), И, Л, ИЛ, СА, СЛ, БП, УПП, БПВ, УЦ	ОП	8	0,006 0,04	0,0003	0,5	0,002
9	128К	И, Д, Н	есть	Н	3	П, ПГ, С, В, М, Д, (СЕ, ВЕ), И, Л, ИЛ, СА, СЛ, БП, БПВ, УПН, УЦ	ОП	16	0,01 0,05	0,001	0,1	0,005

* Команды СЕ и ВЕ имеются только в вариантах с предпоследней цифрой шифра 0, 1, 5 и 6. Команды С, В, М, Д в вариантах с предпоследней цифрой шифра 2, 3, 4, 7, 8 и 9 имеются в двух модификациях: с фиксированной и плавающей запятой.

Условные обозначения

Организация сумматора АЛУ: К – комбинационный, Н – накапливающий.

Микрооперации АЛУ; индекс "ф" – с фиксированной запятой, "п" – с плавающей запятой, М2 – умножение с одновременным анализом двух разрядов множителя (остальные обозначения см. "Набор команд")

Реализация микрооперации сдвига: С – в специальном блоке, Л – в любом блоке

Представление чисел с фиксированной запятой: Ц – целое, Д – дробное

Представление порядков чисел с плавающей запятой: З – со знаком, Х – без знака (характеристика)

Разрабатываемое устройство управления: П – блок микропрограммного управления процессора, А – автомат управления АЛУ

Разрабатываемая принципиальная схема: БМПУ – блок микропрограммного управления, МПА АЛУ – микропрограммный автомат управления АЛУ

Способы адресации: АИ – автоиндексирование, Д – двойная индексация, И – индексная, К – косвенная, КР – косвенная регистровая, Н – непосредственная, ОС – относительно счетчика команд, П – прямая

Тип связей в устройстве управления: М – магистральные, Н – непосредственные

Набор команд: Ч – чтение, З – запись, П – пересылка слова, ПГ – пересылка группы слов, С – сложение, В – вычитание, М – умножение, Д – деление, СЕ – десятичное сложение, ВЕ – десятичное вычитание, И – логическое умножение, Л – логическое сложение, ИЛ – сумма по модулю два (исключающее ИЛИ), СА – сдвиг арифметический СЛ – сдвиг логический, ЗгР – загрузка РОН, ЗпР – запись содержимого РОН в ОП, БП – безусловный переход, БПВ – безусловный переход с возвратом (вызов подпрограммы), УПН – условный переход по нулевому результату предыдущего действия (РПД), УПО – условный переход по отрицательному РПД, УПП – условный переход по положительному РПД, УЦ – управление циклом (условный переход по индексу или счетчику)

Размещение РОН: ОП – в оперативной памяти, Пр – в процессоре

4.2. Тренировочные тесты

Репетиционный тест по разделу 1

1. MIPS – это …

a) название внутреннего блока персональной ЭВМ
b) способ доступа к памяти со стороны внешних устройств
c) единица измерения быстродействия ЭВМ
d) единица измерения скорости передачи данных

2. Технологическая норма микропроцессоров – это …

a) допустимая степень загрязненности помещений, в которых изготавливаются микросхемы процессоров
b) технология изготовления интегральных микросхем с рабочей частотой более 1 Ггц
c) минимальный физический размер элементов и соединений микропроцессора
d) технология изготовления микросхем процессоров

3. Принцип Дж. фон Неймана, вышедший из употребления в ЭВМ, это …
a) принцип хранения программ в памяти
b) выполнение в АЛУ операций только с фиксированной запятой
c) использование иерархической памяти
d) выполнение операций над всеми разрядами одновременно

4. Эффективность ЭВМ оценивается по …
a) быстродействию
b) надежности
c) универсальности
d) специальному критерию

5. Универсальность ЭВМ понимается обычно в … смысле.
a) логическом
b) алгоритмическом
c) стоимостном
d) скоростном

6. Для оценки производительности вычислительных систем используется …
a) динамическое программирование
b) линейное программирование
c) теория массового обслуживания
d) теория алгоритмов

7. Более удобным режимом работы ЭВМ для пользователя является …
a) пакетный
b) конвейерный
c) коллективного доступа
d) реального времени

8. Использование простых команд характерно для ЭВМ с … архитектурой.
a) RISC
b) динамической
c) векторной
d) CISC

Ответы к тесту по разделу 1

1.c, 2.c, 3.b, 4.d, 5.b, 6.c, 7.c, 8.a

Репетиционный тест по разделу 2

1. EPROM – это …
a) комитет стандартизации передачи данных
b) оперативная память
c) название интерфейса ЭВМ
d) перепрограммируемое ПЗУ

2. Плотность записи служит характеристикой для …
a) ассоциативных ЗУ
b) динамических ЗУ
c) кэш-памяти
d) жестких дисков

3. Операционную систему лучше устанавливать на первом разделе жесткого диска, потому что …

a) первый раздел больше по объему
b) время поиска на первом разделе меньше
c)первый раздел более надежный
d)скорость передачи информации на первом разделе выше

4. Среднее время обращения к двухуровневой памяти, включающей кэш со временем обращения T_c= 1,5 нс и ОЗУ со временем обращения T_o= 10 нс при отношении попаданий h = 0,92, составляет …
a) 2,53 нс
b) 2,32 нс
c) 2,18 нс
d) 1,92 нс

5. В кэше прямого отображения (на 8192 строки по 32 байта) запрещено одновременное размещение физических адресов …
a) 00F0ABCA и 01F0ABEA
b) 00F0ABCA и 01F0ABDA
c) 00F0ABCA и 00F0ABEA
d) 00F0ABCA и 00F0ABDA

6. Параметр … не относится к синхронной динамической памяти.
a) время предзаряда строки
b) время задержки сигнала записи
c) задержка появления данных
d) время между стробом строки и стробом столбца

7. Механические перемещения при доступе к информации отсутствуют в …
a) ЗУ с переносом зарядов
b) жестких дисках
c) оптических дисках
d) стриммерах

8. Для хранения BIOS используются …
a) жесткие диски
b) кэш-память
c) флэш-память
d) оперативные ЗУ

9. Использование ассоциативных ЗУ более эффективно для задач …
a) вычислительного характера
b) обработки графической информации
c) связанных с обработкой сигналов
d) связанных с поиском информации

10. Пакетный режим работы статических и динамических ЗУ используется для …
a) увеличения частоты работы
b) увеличения скорости передачи
c) повышения надежности
d) повышения пропускной способности

Ответы к тесту по разделу 2

1.d, 2.d, 3.d, 4.c, 5.b, 6.b, 7.a, 8.c, 9.d, 10.d

Репетиционный тест по разделу 3

1. Естественный и принудительный порядкb следования команд отличаются тем, что …
a) при естественном порядке команды имеют одинаковую длину
b) при принудительном порядке обязателен счетчик команд
c) при естественном порядке команды могут располагаться в памяти произвольно
d) при принудительном порядке адрес очередной команды указывается в текущей

2. Использование в АЛУ 16-ричной системы счисления дает …
a) снижение погрешности вычислений
b) увеличение диапазона представления чисел с плавающей запятой
c) ускорение выполнения операций с фиксированной запятой
d) компактность представления чисел внутри АЛУ

3. С помощью 32-разрядного физического адреса можно адресовать память объемом …
a) 512 Мбайт
b) 32 Гбайт
c) 1 Гбайт
d) 4 Гбайт

4. Микропрограммные УУ уступают схемным в …
a) функциональных возможностях
b) быстродействии
c) допустимых размерах микропрограмм
d) возможности изменения микропрограмм

5. Для сложения двоично-десятичных чисел в типовых АЛУ используется …
a) специальный сумматор
b) схема десятичной коррекции
c) специальная кодировка чисел
d) блокировка переносов между четными разрядами

6. Принудительный порядок следования команд – это такой порядок, при котором …
a) команды располагаются в памяти последовательно
b) адрес следующей команды указывается в текущей
c) адреса команд задаются извне
d) данные определяют последовательность выбора команд программы

7. Укажите правильную последовательность этапов выполнения операции сложения с плавающей запятой.
a) нормализация, сложение мантисс, выравнивание порядков
b) выравнивание порядков, сложение мантисс, нормализация
c) выравнивание порядков, нормализация, сложение мантисс
d) сложение мантисс, нормализация, выравнивание порядков

8. Одновременный анализ нескольких разрядов множителя при умножении дает …
a) ускорение выполнения операции
b) повышение точности выполнения операции
c) контроль ошибок при выполнении операции
d) автоматическое округление результата

9. Коррекция результата сложения двоично-десятичных чисел производится посредством …
a) вычитания кода 0110
b) добавления кода 0111
c) вычитания кода 1001
d) добавления кода 0110

10. Адресация с автоиндексированием осуществляется посредством автоматического …
a) наращивания или уменьшения индекса
b) сложения индекса и базы
c) вычитания индекса из базы
d) записи индекса в стек

11. Адресация с масштабированием в процессорах Pentium (Core) используется для …
a) учета формата обрабатываемых данных
b) увеличения масштаба обрабатываемых величин
c) уменьшения масштаба обрабатываемых величин
d) перевода дробных чисел в целые

12. Непосредственная адресация – это адресация, при которой …
a) в команде указывается операнд

b) в команде указывается значение адреса
c) в команде не указывается адрес
d) операнд извлекается из стека

13. Только в командах перехода присутствует этап …
a) формирования исполнительного адреса команды
b) формирования исполнительных адресов операндов
c) выполнения операции
d) выборки операндов из памяти

14. Стек обычно используется при выполнении команд …
a) вызова подпрограммы
b) условного перехода
c) безусловного перехода
d) умножения

15. При формировании адреса используются компоненты …
a) база, индекс, смещение
b) смещение, граница, база
c) база, индекс, граница
d) индекс, граница, смещение

Ответы к тесту по разделу 3

1.d, 2.b, 3.d, 4.b, 5.b, 6.b, 7.b, 8.a, 9.a, 10.a, 10.d, 11.a, 12.a, 13.a, 14.a, 15.a

Репетиционный тест по разделу 4

1. Векторные процессоры соответствуют параллельной архитектуре …
a) SISD
b) SIMD
c) MISD
d) MIMD

2. Состояние процессора при прерываниях сохраняется …
a) в специальной памяти
b) в стеке
c) на жестком диске
d) в кэше

3. Увеличение количества ступеней в конвейере процессора позволяет …
a)сократить время выполнения этапов обработки команды
b) уменьшить количество неправильно определяемых переходов
c) сократить время выполнения команды
d) упростить процедуру формирования адреса следующей команды

4. Не обеспечивает возможности повысить рабочую частоту процессора …
a) применение алгоритмов ускоренного выполнения операций
b) использование RISC-архитектуры
c) увеличение количества ступеней конвейера
d) использование более быстродействующих элементов

5. Защита памяти – это …
a) предотвращение сбоев памяти
b) предотвращение доступа программ к кодам и локальной информации других программ
c) стабилизация напряжения питания памяти
d) разделение памяти на независимые блоки

6. Сегментная организация памяти, в отличие от страничной, может обеспечить …
a) защиту памяти
b) выделение программам блоков памяти разной длины
c) обмен между ступенями памяти

d) перемещаемость программ

7. К прерываниям ПЭВМ IA-32, относятся …

a) прерывания от таймера, прерывания от клавиатуры, немаскируемые прерывания
b) прерывания от таймера, немаскируемые прерывания, внешние прерывания
c) прерывания от клавиатуры, немаскируемые прерывания, прерывания от средств прямого управления
d) немаскируемые прерывания, внешние прерывания, прерывания от жесткого диска

8. Страничная организация памяти не обеспечивает …

a) выделение программам блоков памяти разной длины
b) защиту памяти
c) перемещаемость программ
d) приоритет обмена страницами

9. Неупорядоченное выполнение команд представляет собой …

a) способ повышения производительности
b) результат сбоя в управлении
c) особый режим работы
d) специальный алгоритм решения задач

10. Суперскалярный процессор – это процессор …

a) который обрабатывает скалярные величины в конвейерном режиме
b) у которого имеется более одного исполнительного конвейера
c) суперЭВМ
d) для обработки векторов

11. ЭВМ с CISC-архитектурой – это ЭВМ …

a) с обычной системой команд
b) с конвейерной обработкой графики
c) управляющит ЭВМ
d) для обработки сигналом

12. После включения персональной ЭВМ запускается тест …

a) POST
b) ROST
c) жесткого диска
d) питания

13. Команда, вызвавшая нарушение защиты памяти обрабатывается следующим образом …

a) выполнение команды прекращается
b) команда не вызывается на исполнение
c) команда завершается обычным образом
d) команда игнорируется

14. Более высокой скорости передачи данных требует …

a) гибкий диск
b) принтер
c) сканер
d) звуковая карта

15. При работе системы прерываний программно выполняется …

a) установка маски запросов прерываний
b) прием запросов прерываний
c) формирование общего сигнала прерывания для процессора
d) запоминание счетчика команд

16. Таблица страниц – это таблица …

a) указывающая количество страниц в памяти
b) содержащая физические адреса страниц
c) задающая размеры страниц
d) обращений, выполненных к страницам памяти

17. Архитектуру многопроцессорных ЭВМ с однородным доступом к памяти характеризует то, что …

a) время доступа к различным модулям памяти одинаково
b) все процессоры обращаются к одному модулю памяти
c) все модули памяти одинаковы
d) обращения к памяти производятся только с помощью многоступенчатых сетей

18. Порядок узла в сети межсоединений вычислительной системы характеризует …

a) количество передач, выполняемых узлом
b) роль узла в вычислительной системе
c) расположение узла, по отношению к входу сети
d) количество подключенных к нему узлов

19. Бисекция сети межсоединений вычислительной системы – это …

a) секция сети, имеющая два входа и два выхода
b) секция сети, имеющая вход и выход
c) часть сети, разделенной на две части с одинаковым количеством узлов
d) секция сети, состоящая из двух узлов

20. Масштабирование при решении задач на АВМ применяется для …

a) представления величин в допустимых диапазонах
b) уменьшения погрешности решения
c) повышения скорости решения
d) проверки правильности схемы решения

Ответы к тесту по разделу 4

1.b, 2.b, 3.a, 4.a, 5.b, 6.b, 7.a, 8.a, 9.a, 10.b, 11.a, 12.a, 13.a, 14.c, 15.a, 16.b, 17.a, 18.d, 19.c, 20.a

4.3. Вопросы для подготовки к экзамену

1. Сравнительная оценка аналоговых и цифровых вычислительных машин
2. Структура ЦВМ, принципы Неймана
3. Структура ПЭВМ
4. Классификация ЭВМ
5. Оценка производительности ЭВМ
6. Оценка эффективности ЭВМ
7. Режимы работы ЭВМ
8. Этапы проектирования ЭВМ
9. Классификация ЗУ по функциональному назначению
10. Конструктивно-логические особенности организации ЗУ
11. Основные типы и сравнительная оценка полупроводниковых ЗУ
12. Элементы памяти статических и динамических ОЗУ
13. Структурная организация БИС ОЗУ
14. Динамические оперативные ЗУ
15. Организация модулей оперативных ЗУ на БИС
16. Постоянные и репрограммируемые ЗУ
17. Флэш-память
18. Статические ОЗУ и организация кэш-памяти ПЭВМ
19. Ассоциативные и многофункциональные ЗУ
20. Назначение, состав и структура АЛУ
21. Классификация АЛУ
22. Типы функций, реализуемых в ЭВМ на различных уровнях

23. Языки описания АЛУ
24. Преобразование алгоритмов и порядок перехода от функционального к структурному представлению АЛУ
25. Базовые преобразования структур АЛУ
26. Оценка эффективности структур АЛУ
27. Обобщенная структура устройства для сложения чисел с плавающей запятой
28. Обобщенная структура устройства для умножения
29. Обобщенная структура устройства для деления
30. Структура АЛУ и алгоритм выполнения сложения с плавающей запятой
31. Структура АЛУ и алгоритм выполнения умножения с фиксированной запятой
32. Структура АЛУ и алгоритм выполнения деления с фиксированной запятой
33. Структура АЛУ и алгоритм выполнения десятичного сложения
34. Устройства управления (УУ) ЭВМ. Основные понятия и определения. Функции устройств управления
35. Управление выполнением последовательности команд
36. Управление выполнением операций
37. Способы адресации данных
38. Способы адресации в ПЭВМ с 32-разрядной архитектурой
39. Классификация устройств управления ЭВМ
40. Схемные устройства управления
41. Правила перехода от граф-схемы микропрограммы к графам микропрограммных автоматов Мили и Мура
42. Принцип микропрограммного управления. Модель Уилкса
43. Общая структура устройств микропрограммного управления
44. Способы кодирования микроопераций и схемы формирования управляющих сигналов
45. Формирование адресов микрокоманд
46. Последовательность выполнения микрокоманд
47. Назначение и основные характеристики систем прерывания программ
48. Функции и типы систем прерывания программ
49. Запоминание состояния, переход к прерывающей программе и возврат из нее
50. Приоритетное обслуживание прерываний
51. Особенности системы прерывания ПЭВМ
52. Защищенный режим в ПЭВМ. Слово состояния программы
53. Классификация систем памяти
54. Защита памяти
55. Страничная адресация памяти
56. Сегментная адресация памяти (на примере ПЭВМ)
57. Алгоритмы замещения информации в основной памяти
58. Алгоритмы управления очередностью обмена информацией с внешними ЗУ
59. Принципы построения систем ввода-вывода
60. Периферийные устройства
61. Организация интерфейсов ввода-вывода
62. Программное управление вводом-выводом в ЭВМ
63. Передача данных (ввод-вывод) с прямым доступом к памяти
64. Архитектура классических ЭВМ (Структура ЭВМ Единой Системы и СМ ЭВМ)
65. Основные типы микропроцессоров. Структура микроЭВМ
66. Процессоры с RISC-архитектурой

67. ЭВМ, управляемые потоками данных
68. Принципы конвейерной обработки команд
69. Суперскалярная архитектура
70. Гиперпоточная архитектура и архитектура ЭВМ с большой длиной командного слова
71. Классификация вычислительных систем
72. Организация доступа к памяти в ВС
73. Топологии соединений в ВС
74. Средства тестирования и отладки в процессорах ПЭВМ (регистры отладки, интерфейс JTAG)
75. Цифровой аудиоканал
76. Системный порт, таймер и спикер (динамик)
77. Пространство ввода-вывода в ПЭВМ (адреса портов, аксессуаров системной платы)
78. BIOS. Основные установки и тест начального включения
79. CMOS-память и часы реального времени
80. Адресация секторов и записи о разделах жесткого диска
81. Блоки питания ПЭВМ
82. Заземление ПЭВМ
83. Принципы построения аналоговых вычислительных машин
84. Операционный усилитель. Основные соотношения и режимы инвертора, сумматора, интегратора
85. Нелинейные блоки АВМ

Содержание

Копейкин Михаил Васильевич
Спиридонов Виктор Валентинович
Шумова Елена Олеговна

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

Учебно-методический комплекс

Редактор И.Н. Садчикова

Сводный темплан 2009 г.
Лицензия ЛР № 020308 от 14.02.97
Санитарно-эпидемиологическое заключение № 78.01.07.953.П.005641.11.03 от 21.11.2003 г.

Подписано в печать			Формат 60×84 1/16.
Б. кн.-журн.	П.л.	Б.л.	Изд-во СЗТУ
Тираж	экз.		Заказ

Северо-Западный государственный заочный технический университет
Издательство СЗТУ, член Издательско-полиграфической ассоциации университетов России
191186, Санкт-Петербург, ул. Миллионная, 5