УЧЕБНО-НАУЧНО-ПРОИЗВОДСТВЕННЫЙ КОМПЛЕКС
«МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ КЫРГЫЗСТАНА»

«УТВЕРЖДЕНО»
Ректор НОУ УНПК «МУК»
к.т.н., доцент Савченков Е.Ю.
_____
«16» 10 2018 г.

БАКАЛАВРИАТ

Кафедра «Компьютерные информационные системы и управление»

Учебно-методический комплекс дисциплины

ЭВМ и периферийные устройства

Направление: **710100 «Информатика и вычислительная техника»**

Профиль: **Компьютерные информационные системы в бизнесе**

Академическая степень **- бакалавр**

Форма обучения **(очная)**

График проведения модулей 6-семестр

неделя	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
лекц. зан.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
прак./лаб. зан.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

«РАССМОТРЕНО»
Протокол заседания кафедры
«КИСиУ»
№ 2 от 16.10.2018
Зав. кафедрой д.т.н., проф. Миркин Е.Л.
_____

«СОГЛАСОВАНО»
Проректор по академ. вопросам
проф. Мадалиев М.М.
_____

Составитель _____ к.ф.-м..н., и.о.доцент Красниченко Л.С.

Директор Научной библиотеки _____ Асанова Ж.Ш.

БИШКЕК 2018

ОГЛАВЛЕНИЕ

АННОТАЦИЯ

Дисциплина «ЭВМ и периферийные устройства» представляет собой специализированный курс, который является одним из важнейших при подготовке специалистов в области информационных технологий. 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. Данный курс состоит из четырех модулей, итоговый контроль в виде экзамена.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ (МОДУЛЕЙ)

1. Пояснительная записка

1.1. Миссия и стратегия

Миссия НОУ УНПК "МУК" – подготовка международно - признанных, свободно мыслящих специалистов, открытых для перемен и способных трансформировать знания в ценности на благо развития общества.

Видение НОУ УНПК «МУК»- создание динамичного и креативного университета с инновационными научно-образовательными программами и с современной инфраструктурой, способствующие достижению академических и профессиональных целей.

Стратегии развития - модернизация образовательной деятельности университета – совершенствование образовательного процесса в соответствии с требованиями Болонского процесса.

1.2. Цель и задачи дисциплины

Цель дисциплины:

предоставление обучаемым знаний по вопросам функциональной и структурной организации ЭВМ, ее составных частей с применением современных информационных технологий; усвоение этих знаний студентами, а также формирование у них мотивации к самообразованию за счет активизации самостоятельной познавательной деятельности.

Задачи дисциплины:

В результате усвоения материала настоящего курса студенты должны знать:

- основные принципы организации технических средств ЭВМ и систем;
- функциональную и структурную организацию ЭВМ;
- принципы построения основных устройств ЭВМ;
- важнейшие этапы и тенденции в развитии цифровой, аналоговой и гибридной вычислительной техники;
- методы оценки параметров ЭВМ и отдельных их устройств.

1.3. Формируемые компетенции, а также перечень планируемых результатов обучения по дисциплине

Дисциплина направлена на формирование следующих компетенций:

- ✓ способен разрабатывать бизнес-планы и технические задания на оснащение отделов, лабораторий, офисов компьютерным и сетевым оборудованием (ПК-1);
- ✓ способен сопрягать аппаратные и программные средства в составе информационных и автоматизированных систем (ПК-10);
- ✓ способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем (ПК-11).

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования:

Знать:

- основы построения и архитектуры ЭВМ
- основные понятия и терминологию в области вычислительной техники ;
- технические и эксплуатационные характеристики компьютеров;

- классификации ЭВМ;
- особенности организации различных типов ЭВМ;
- функциональную и структурную организацию центрального процессора, памяти компьютера;
- организацию прерываний и ввода-вывода;
- современное состояние и тенденции развития ЭВМ;

Уметь:

- выбирать, комплексировать и тестировать аппаратные средства вычислительных систем:
- проводить анализ всего многообразия типов ЭВМ с целью выбора наиболее приемлемого варианта для конкретного использования;
- проводить сравнительный анализ параметров основных технических средств ЭВМ (процессора, памяти);
- уметь выбирать базовую конфигурацию компьютера;
- использовать сеть Internet для работы с Web-серверами ведущих фирм производителей средств вычислительной техники;

Владеть: навыками конфигурирования компьютеров различного назначения

1.4. Место дисциплины (модулей) в структуре ООП ВПО

Дисциплина «ЭВМ и периферийные устройства» является частью профессионального цикла (блока) дисциплин учебного плана по направлению подготовки 710100 «Информатика и вычислительная техника» подготовки бакалавров (специализации Компьютерные информационные системы).

Для освоения дисциплины (модулей) необходимы компетенции, сформированные в ходе изучения следующих дисциплин и прохождения практик: основные разделы математики, программирования.

В результате освоения дисциплины (модулей) формируются компетенции, необходимые для изучения следующих дисциплин и прохождения практик: написание выпускной квалифицированной работы.

2. Структура дисциплины

Общая трудоемкость дисциплины составляет 4 кредита, 120ч., в том числе аудиторная работа обучающихся с преподавателем 64ч., самостоятельная работа обучающихся 38ч., 18ч-СРСП.

№ п/п	*Раздел, Темы Дисциплины*	*Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)*				*Формы текущего контроля успеваемости (по неделям семестра) Форма промежуточной аттестации (по семестрам)*
		Лекции	*Лаб.*	*СРС*	*СРСП*	

1	**Раздел 1. Общие сведения об ЭВМ**	***8***	***8***	***8***	***4***	
2	Тема 1. Этапы развития ЭВМ	*4*	*4*	*4*	*2*	
	Тема 2. Обобщенная структура ЭВМ	*4*	*4*	*4*	*2*	*Сдача модуля*
3	**Раздел 2. Архитектура классической ЭВМ.**	***8***	***8***	***8***	***5***	
4	Тема 3. Принцип программного управления	*3*	*3*	*3*	*2*	
5	Тема 4. Способы адресации команд	*3*	*3*	*3*	*2*	
	Тема 5Способы адресации операндов	*2*	*2*	*2*	*1*	*Сдача модуля*
6	**Раздел 3. Запоминающие устройства ЭВМ**	***8***	***8***	***8***	***4***	
7	Тема 6. Основные характеристики ЗУ	*4*	*4*	*4*	*2*	
8	Тема 7. Структура ОЗУ с произвольной выборкой	*4*	*4*	*4*	*2*	*Сдача модуля*
9	**Раздел 4. Принципы организации процессоров**	***8***	***8***	***14***	***5***	
10	Тема 8. Обобщенные структуры процессоров с непосредственными и магистральными связями	*4*	*4*	*7*	*5*	
11	Тема 9. Развернутая структура процессора и его функционирование	*4*	*4*	*7*	*2*	*Сдача модуля*
		32	*32*	*38*	*18*	

Содержание дисциплины

Раздел 1. Общие сведения об ЭВМ

Тема 1. Этапы развития ЭВМ.

Характеристики ЭВМ. Классификация средств ЭВТ.Структуры ЭВМ.

Тема 2. Обобщенная структура ЭВМ
Структура ЭВМ на основе общей шины

Раздел 2. Архитектура классической ЭВМ.

Тема 3. Принцип программного управления
Принцип программного управления. Принцип хранимой в памяти программы. Обобщенный формат команд.

Тема 4. Способы адресации команд
Процессоры с принудительной адресацией. Естественная адресация команд.

Тема 5. Способы адресации операндов
Прямая адресация. Косвенная адресация. Регистровая адресация. Непосредственная адресация. Неявная адресация. Относительная адресация. Индексная (автоинкрементная и автодекрементная) адресация

Раздел 3. Запоминающие устройства ЭВМ

Тема 6. Основные характеристики ЗУ
Основные характеристики ЗУ. Структура ОЗУ с произвольной выборкой. Особенности организации динамической памяти.

Тема 7. Структура ОЗУ с произвольной выборкой.
ОЗУ магазинного типа. Ассоциативные ЗУ

Раздел 4. Принципы организации процессоров

Тема 8. Обобщенные структуры процессоров с непосредственными и магистральными связями. Декомпозиция процессора на УА и ОУ. Классификация УУ. Микропрограммные УУ. Принцип микропрограммного управления Уилкса. Структура блока микропрограммного управления.

Тема 9. Развернутая структура процессора и его функционирование
Обобщенная структура процессора с микропрограммным управлением. Рабочий цикл процессора. Понятие о слове состояния процессора (PSW). Процедура выполнения команд перехода (условного и безусловного). Процедура выполнения команд вызова подпрограмм.

4. Конспект лекций – ПРИЛОЖЕНИЕ 1

5. Информационные и образовательные технологии

Информационные и образовательные технологии

№ п/п	***Наименование раздела***	***Виды учебной работы***	***Формируемые компетенции (указывается код компетенции)***	***Информационные и образовательные технологии***
1	***2***	***3***	***4***	***5***
2	Тема 1. Этапы развития ЭВМ	Лекция Лабораторная работа Самостоятельная работа	(ПК-1) (ПК-10) (ПК-11)	**Лекция-визуализация** с применением проектора **Лабораторная работа** согласно текущей темы **Подготовка к занятию** с использованием электронного курса лекций
3	Тема 2. Обобщенная структура ЭВМ	Лекция Лабораторная работа Самостоятельная работа	(ПК-1) (ПК-10) (ПК-11)	**Лекция-визуализация** с применением проектора **Лабораторная работа** согласно текущей темы **Подготовка к занятию** с использованием электронного курса лекций
6	Тема 3. Принцип программного управления	Лекция Лабораторная работа Самостоятельная работа	(ПК-1) (ПК-10) (ПК-11)	**Лекция-визуализация** с применением проектора **Лабораторная работа** согласно текущей темы **Подготовка к занятию** с использованием электронного курса лекций
7	Тема 4. Способы адресации команд	Лекция Лабораторная работа Самостоятельная работа	(ПК-1) (ПК-10) (ПК-11)	**Лекция-визуализация** с применением проектора **Лабораторная работа** согласно текущей темы **Подготовка к занятию** с использованием электронного курса лекций
8	Тема 5Способы адресации операндов	Лекция	(ПК-1) (ПК-10) (ПК-11)	**Лекция-визуализация** с применением проектора

		Лабораторная работа Самостоятельная работа		**Лабораторная работа** согласно текущей темы **Подготовка к занятию** с использованием электронного курса лекций
10	Тема 6. Основные характеристики ЗУ	Лекция Лабораторная работа Самостоятельная работа	(ПК-1) (ПК-10) (ПК-11)	**Лекция-визуализация** с применением проектора **Лабораторная работа** согласно текущей темы **Подготовка к занятию** с использованием электронного курса лекций
11	Тема 7. Структура ОЗУ с произвольной выборкой	Лекция Лабораторная работа Самостоятельная работа	(ПК-1) (ПК-10) (ПК-11)	**Лекция-визуализация** с применением проектора **Лабораторная работа** согласно текущей темы **Подготовка к занятию** с использованием электронного курса лекций
	Тема 8. Обобщенные структуры процессоров с непосредственными и магистральными связями	Лекция Лабораторная работа Самостоятельная работа	(ПК-1) (ПК-10) (ПК-11)	**Лекция-визуализация** с применением проектора **Лабораторная работа** согласно текущей темы **Подготовка к занятию** с использованием электронного курса лекций
	Тема 9. Развернутая структура процессора и его функционирование	Лекция Лабораторная работа Самостоятельная работа	(ПК-1) (ПК-10) (ПК-11)	**Лекция-визуализация** с применением проектора **Лабораторная работа** согласно текущей темы **Подготовка к занятию** с использованием электронного курса лекций

6. Фонд оценочных средств для текущего, рубежного и итогового контролей по итогам освоению дисциплины (модулей)

6.1. Перечень компетенций с указанием этапов их формирования в процессе освоения

дисциплины

Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины представляется в виде таблицы:

№ п/п	*Контролируемые разделы дисциплины (модулей)*	*Код контролируемой компетенции (компетенций)*	*Наименование оценочного средства*
1	Тема 1. Этапы развития ЭВМ	*(*(ПК-1) (ПК-10) (ПК-11)	Практическое задание «*Исследование архитектуры ПК»* Коллоквиум (Вопросы по темам дисциплины)
2	Тема 2. Обобщенная структура ЭВМ		
3	Тема 3. Принцип программного управления	(ПК-1) (ПК-10) (ПК-11) (	Практическое задание «*Исследование программной архитектуры ПК»* Коллоквиум (Вопросы по темам дисциплины)
4	Тема 4. Способы адресации команд		
5	Тема 5Способы адресации операндов		
6	Тема 6. Основные характеристики ЗУ	*(*(ПК-1) (ПК-10) (ПК-11)	Практическое задание *«Работа с машинными командами и командами ассемблера с помощью отладчика debug»* Коллоквиум (Вопросы по темам дисциплины) Контрольная работа
7	Тема 7. Структура ОЗУ с произвольной выборкой		
8	Тема 8. Обобщенные структуры процессоров с непосредственными и магистральными связями	*(*(ПК-1) (ПК-10) (ПК-11)	Практическое задание *«Работа с машинными командами и командами ассемблера с помощью отладчика debug»* Коллоквиум (Вопросы по темам дисциплины) Тест
9	Тема 9. Развернутая структура процессора и его функционирование		

6.2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Методические материалы составляют систему текущего, рубежного и итогового (экзамена) контролей освоения дисциплины (модулей), закрепляют виды и формы текущего, рубежного и итогового контролей знаний, сроки проведения, а также его сроки и формы проведения (устный экзамен, письменный экзамен и т.п.). В системе контроля указывается процедура оценивания

результатов обучения, при использовании балльно-рейтинговой системы приводится таблица с баллами и требованиями к пороговым значениям достижений по видам деятельности обучающихся; показывается механизм получения оценки (из чего складывается оценка по дисциплине (модулю).

Текущий контроль осуществляется в виде опроса, участие в дискуссии на семинаре, выполнение самостоятельной - оценивается до **80 баллов**.

Рубежный контроль (сдача модулей) проводится преподавателем и представляет собой письменный контроль, либо компьютерное тестирование знаний по теоретическому и практическому материалу. Контрольные вопросы рубежного контроля включают полный объём материала части дисциплины (модулей), позволяющий оценить знания, обучающихся по изученному материалу и соответствовать УМК дисциплины, которое оценивается до **20 баллов**.

Итоговый контроль (экзамен) знаний принимается по экзаменационным билетам, включающий теоретические вопросы и практическое задание, и оценивается до **20 баллов**.

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - Прием лабораторных работ	1,2,3,4недели	8 баллов	До 40 баллов
-опрос	1, ,2,3,4недели	6 баллов	До 30 баллов
- посещаемость	1, ,2,3,4неделя	2 балла	10 баллов
Рубежный контроль: (сдача модуля)	4 неделя	100%×0,2=20 баллов	
Итого за I модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - Прием лабораторных работ	5,6,7,8 недели	10 баллов	До 40 баллов
-опрос	5,6,7,8 недели	6 баллов	До 30 баллов
- посещаемость	5,6,7,8 недели	2 балла	10 баллов
Рубежный контроль: (сдача модуля)	**8 неделя**	100%×0,2=20 баллов	
Итого за II модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - Прием лабораторных работ -опрос - посещаемость	 9, 10, 11 недели 9, 10, 11 недели 9, 10, 11 недели	 8 баллов 6 баллов 2 балла	 До 40 баллов До 30 баллов 10 баллов
Рубежный контроль: (сдача модуля)	**11 неделя**	100%×0,2=20 баллов	
Итого за III модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - Прием лабораторных работ -опрос - посещаемость	 12,13,14 недели 12,13,14 недели 12,13,14 недели	 10 баллов 6 баллов 2 балла	 До 40 баллов До 30 баллов 10 баллов
Рубежный контроль: (сдача модуля)	**14 неделя**	100%×0,2=20 баллов	
Итого за IV модуль			До 100 баллов

Экзаменатор выставляет по результатам балльной системы в семестре экзаменационную оценку без сдачи экзамена, набравшим суммарное количество баллов, достаточное для выставления оценки от 55 и выше баллов – автоматически (при согласии обучающегося).

Полученный совокупный результат (максимум 100 баллов) конвертируется в традиционную шкалу:

Рейтинговая оценка (баллов)	Оценка экзамена
От 0 - до 54	неудовлетворительно
от 55 - до 69 включительно	удовлетворительно
от 70 – до 84 включительно	хорошо
от 85 – до 100	отлично

6.1.Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Текущий контроль (0 - 80 баллов)

При оценивании посещаемости, опроса и приема лабораторных работ из расчета на одну неделю учитываются:

- посещаемость (2 балла одно занятие (10 баллов за модуль)
- степень раскрытия содержания материала (2.8 балла одно занятие (14 баллов за модуль);

- изложение материала (грамотность речи, точность использования терминологии и символики, логическая последовательность изложения материала (2.8 балла одно занятие (14 баллов за модуль);
- знание теории изученных вопросов (2.8 балла одно занятие (14 баллов за модуль);
- сформированность и устойчивость используемых при ответе умений и навыков (2.8 балла одно занятие (14 баллов за модуль);
-точность решения задачи (2.8 балла одно занятие (14 баллов за модуль).

Рубежный контроль (0 – 20 баллов)

При оценивании контрольной работы учитывается:

- полнота выполненной работы (задание выполнено не полностью и/или допущены две и более ошибки или три и более неточности) – 8 баллов;
- обоснованность содержания и выводов работы (задание выполнено полностью, но обоснование содержания и выводов недостаточны, но рассуждения верны) – 14 баллов;
- работа выполнена полностью, в рассуждениях и обосновании нет пробелов или ошибок, возможна одна неточность - 17 баллов.
- работа выполнена полностью, в рассуждениях и обосновании нет пробелов или ошибок - 20 баллов.

При оценивании теста учитывается:

- полнота выполненной работы (задание выполнено не полностью и/или допущены две и более ошибки или три и более неточности) – до 20 баллов;

Итоговый контроль (экзаменационная сессия) - ИК = Бср × 0,8+Бэкз × 0,2

При проведении итогового контроля обучающийся должен ответить на 3 вопроса (два вопроса теоретического характера и один вопрос практического характера).

При оценивании ответа на вопрос теоретического характера учитывается:

- теоретическое содержание не освоено, знание материала носит фрагментарный характер, наличие грубых ошибок в ответе (2 балла);
- теоретическое содержание освоено частично, допущено не более двух-трех недочетов (5 баллов);
- теоретическое содержание освоено почти полностью, допущено не более одного-двух недочетов, но обучающийся смог бы их исправить самостоятельно (8 баллов);
- теоретическое содержание освоено полностью, ответ построен по собственному плану (10 баллов).

При оценивании ответа на вопрос практического характера учитывается:

- ответ содержит менее 20% правильного решения (3 балла);
- ответ содержит 21-89 % правильного решения (7 баллов);
- ответ содержит 90% и более правильного решения (10 баллов).

6.2.Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности.

Раздел УМК включает образцы оценочных средств, примерные перечни вопросов и заданий в соответствии со структурой дисциплины и системой контроля.

Контрольные вопросы :

Контрольные вопросы Общие сведения об ЭВМ

По каким признакам классифицируются ЭВМ?
В чем различие структур ЭВМ на основе локальных шин и общей шины?
Каково назначение процессора в ЭВМ?
Назначение ОП и УВВ?

Контрольные вопросы Архитектура классической ЭВМ.

Какова структура команды? Какие поля включает команда? Чем определяется длина команды?
В чем заключается естественная адресация команд в ЦВМ?
В чем заключается принудительная адресация команд в ЦВМ?
Перечислите достоинства и недостатки естественной адресации?
Перечислите достоинства и недостатки принудительной адресации?
Какие существуют способы адресации операндов?
Достоинства неявной и регистровой адресации?
В чем заключается непосредственная адресация?
В чем заключается прямая адресация?
Какие преимущества косвенной адресации?
Каково назначение относительной адресации?
Каково назначение индексной адресации?

Контрольные вопросы запоминающие устройства эвм

По каким признакам классифицируются запоминающие устройства?
Назначение ВЗУ и СОЗУ?
Назовите признаки ЗУ прямого и последовательного доступов?
Расшифруйте сокращения ПЗУ и ЗУПВ.
Перечислите основные характеристики ЗУ.
Что такое “Цикл памяти”?
Каковы преимущества ЗУ с произвольной выборкой?
Перечислите основные узлы ЗУПВ.
Какова организация стековых ЗУ и где они применяются?
Что общего в работе стековой памяти типов LIFO и FIFO?
В чем заключается принцип действия ассоциативных ЗУ?

Контрольные вопросы Принципы организации процессоров

Перечислите функции процессора.
Каковы функции РК и СчК в процессоре?
Назначение АЛУ процессора?
Что дает введение в состав АЛУ РОНов?
Назначение УУ процессора?
Основное отличие между аппаратными и микропрограммными УУ?
Назначение РАМК УУ?
Перечислите основные узлы блока микропрограммного управления.
Опишите последовательность выполнения команды пересылки данных между РОН, используя структуру процессора с микропрограммным управлением.
Что такое ССП (PSW)?
Опишите процедуру выполнения команд условного и безусловного переходов.
Опишите процедуру выполнения команды вызова подпрограммы.

Какое основное отличие процедур выполнения команд вызова подпрограмм и выполнения команд условного и безусловного переходов.

Тест ***Приложение III***

Самостоятельная работа студентов ***Приложение II***

7. Учебно-методическое и информационное обеспечение дисциплины

7.1.Список источников и литературы

Основные учебники

1. Жмакин А.П. Архитектура ЭВМ: Учебное пособие / А.П. Жмакин. - СПб.: БХВ - Петербург, 2006 - 320с
2. Олифер В.Г. Компьютерные сети : Принципы, технологии, протоколы / В.Г. Олифер, Н.А. Олифер. - СПб: Питер, 2010 - 944с.
3. Таненбаум Э. Компьютерные сети / Э. Таненбаум. - СПб: Питер, 2010 - 992с.
4. Чекмарев Ю.В. Вычислительные системы, сети и телекоммуникации / Ю.В. Чекмарев.- ДМК Пресс, 2009 – 184с.

Дополнительная литература

1. Вильховченко О.С Современный компьютер: устройство, выбор, модернизация / О.С. Вильховченко. - СПб: Питер, 2000 - 512с.
2. Гук М. Аппартные средства локальных сетей : Энциклопедия / М. Гук. - СПб: Питер, 2004 - 573с.
3. Гук М. Аппаратные средства IBM PC. / М. Гук. - СПб: Питер, 2000 - 816с.
4. Колесниченко О.В. Аппаратные средства PC / О.В. Колесниченко, И.В. Шишигин. - СПб.: БХВ - Петербург, 2004 - 1152с.
5. Косцов А. Все о персональном компьютере: Большая энциклопедия / А. Косцов, В. Косцов. - М.: Мартин, 2005 - 720с.
6. Леонтьев В.П. Новейшая энциклопедия персонального компьютера 2003 / В.П. Леонтьев. - М.: ОЛМА-Пресс, 2003 - 920с
7. Мельников Д.А Информационные процессы в компьютерных сетях : Протоколы, стандарты, интерфейсы, модели / Д.А. Мельников. - М.: Кудиц-Образ, 2001 - 256с.

7.2.Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимый для освоения дисциплины (модулей)

1. https://www.intuit.ru/search
2. https://www.twirpx.com/ Библиотека все для студента
3. https://uk.sagepub.com/en-gb/asi/home
4. https://uk.sagepub.com/en-gb/asi/sage-premier
5. https://www.nejm.org/
6. https://uk.sagepub.com/en-gb/asi/imeche
7. http://global.oup.com/?cc=kg
8. https://www.cambridge.org
9. https://www.intellectbooks.co.uk/journals/index/
10. http://iopscience.iop.org/journalList

11. https://royalsociety.org/journals/
12. https://www.elibrary.imf.org/?redirect=true
13. https://www.elgaronline.com/page/70/journals
14. http://www.dukejournals.org/
15. ttp://www.iprbookshop.ru/
16. http://kyrlibnet.kg/ru/
17. http://biblioteka.kg/

8. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся

8.1.Планы практических (семинарских) и лабораторных занятий. -ПРИЛОЖЕНИЕ II

8.2. Методические указания по организации и проведению лабораторных занятий

8.3. Методические указания для обучающихся по освоению дисциплины (модулей)

Методические указания предназначены для рационального распределения времени студента по видам **самостоятельной работы** и разделам дисциплины. Они составляются на основе сведений о трудоемкости дисциплины, ее содержании и видах работы по ее изучению, а также учебно-методического и информационного обеспечения. В раздел включаются: рекомендации по изучению дисциплины (модулей) или отдельных тематических разделов, вопросы и задания для самостоятельной работы, материалы, необходимые, для подготовки к занятиям (разделы книг, статьи и т.д.). Раздел может быть представлен в табличной форме.

8.4.Методические рекомендации по подготовке отчетов по лабораторным работам

Требования при оформлении лабораторных работ:

1. Требования
    - Первая страница Титульный лист
    - Условия задачи, цели, этапы выполнения
    - Программный код
    - Графики
    - Результаты
    - Выводы

Правила оформления лабораторных работ:

- текст печатается на странице формата А4;
- шрифт – Times New Roman;
- размеры полей: левое – 3 см, верхнее – 2 см, правое – 2 см и нижнее – 2 см;
- выравнивание по ширине.
- размер шрифта основного текста – 12;
- интервал межстрочный (полуторный) – 1,5;
- название работы печатается полужирным, размер шрифта – 14;
- заголовки печатаются жирным шрифтом 14-ым размером, перед ними следует оставить пустую строку, выравниваются по центу;
- подзаголовки печатаются жирным шрифтом 12-ым размером выравниваются по центу;

- нумерация страниц – внизу по центру.
- Нумерация рисунков, графиков и т.п. Например: (рис.1 Название рисунка) рисунки нумеруются снизу и по центру, таблица (Таблица 1. Название таблицы) таблицы нумеруются сверху выравнивание к правому краю.
    - Библиографические ссылки при цитировании приводятся в конце статьи и нумеруются согласно порядку цитирования в тексте. Указываются автор (сначала фамилия, потом инициалы), название, место и год издания, страница. Порядковые номера ссылок должны быть написаны внутри квадратных скобок (например: [1], [2]). Источники приводятся с указанием в алфавитном порядке фамилий и инициалов всех авторов, сначала отечественных, затем иностранных, полного названия статьи, названия источника, где напечатана статья, том, номер, страницы (от и до) или полное название книги, место и год издания. Фамилии иностранных авторов, название и выходные данные их работ даются в оригинальной транскрипции. Каждый источник приводится с новой строки.

9. Материально-техническое обеспечение дисциплины

Минимальные требования к материально-техническому обеспечению дисциплины:

- Компьютерный класс
- проектор, экран
- колонки
- программное обеспечение NNT Matlab.

10. 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 управлениямикропрограммное – 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

Приложения

ПРИЛОЖЕНИЕ 1

1. ОБЩИЕ СВЕДЕНИЯ О ЭВМ

1.1 Этапы развития ЭВМ

Идея использования программного управления для построения устройств, автоматически выполняющих арифметические вычисления, была впервые высказана английским математиком Ч. Бэббиджем в 1833 г. Однако его попытки построить механическое вычислительное устройство с программным управлением не увенчались успехом.

Фактически эта идея была реализована спустя более чем 100 лет, когда в 1942 г. К. Цюзе в Германии и в 1944 г. Г. Айкен в США построили на электромагнитных реле вычислительные машины с управлением от перфоленты, на которую записывалась программа вычислений.

Идея программного управления вычислительным процессом была существенно развита американским математиком Дж. фон Нейманом, который в 1945 г. сформулировал принцип хранимой в памяти программы. Первые ЭВМ с программным управлением и с хранимой в памяти программой появились практически одновременно в Англии, США и СССР.

На протяжении более шести десятилетий электронная вычислительная техника бурно развивается. Появились, сменяя друг друга, несколько поколений ЭВМ. Появление новых поколений ЭВМ вызывалось расширением областей и развитием методов их применения, требовавших более производительных, более дешевых и более надежных машин.

Поколение ЭВМ определяется совокупностью взаимосвязанных и взаимообусловленных существенных особенностей и характеристик, используемых при построении машин, конструктивно-технологической (в первую очередь элементной) базы и реализуемой в машине архитектуры.

Первое поколение образовали ламповые ЭВМ, промышленный выпуск которых начался в начале 50-х гг. В качестве компонентов логических элементов использовались электронные лампы. ЭВМ этого поколения характеризовались низкой надежностью и высокой стоимостью. Их быстродействие составляло всего 5 ÷ 8 тыс. опер/с.

Второе поколение ЭВМ появилось в конце 50-х годов. Элементной базой второго поколения ЭВМ были полупроводниковые приборы, благодаря чему повысилась их надежность, а производительность возросла до 30 тыс. опер/с (Минск-2, Минск-22, Минск-32, Урал-10, БЭСМ-4, М-220).

В рамках ЭВМ 2-го поколения академик. Лебедев С.А. создал ЭВМ БЭСМ-6 с производительностью до 1 млн. опер/с.

С середины 60-х годов отсчитывается начало появления ЭВМ 3-го поколения. Их элементной базой стали ИМС. В рамках этого поколения фирма IBM создала систему машин IBM-386. В г. Пензе была разработана ЭВМ Урал-16. Однако в это время стало заметно отставание СССР в области элементной базы, что не могло не сказаться и на характеристиках отечественных ЭВМ. Поэтому правительством было принято решение о переходе на производство техники, разработанной фирмой IBM. В СССР она выпускалась под названием Единая Система ЭВМ (ЕС ЭВМ). Наиболее быстродействующая ЭВМ ряда ЕС ЭВМ выпускалась заводом ВЭМ (г. Пенза). Она выполняла до 5 млн. опер/с.

Конструктивно-технологической основой ЭВМ четвертого поколения являются большие (БИС) и сверхбольшие (СБИС) ИМС.

К четвертому поколению относятся реализованные на СБИС такие новые средства вычислительной техники, как микропроцессоры и создаваемые на их основе микро-ЭВМ.

Микропроцессоры и микро-ЭВМ нашли широкое применение в устройствах и системах автоматизации измерений, обработки данных и управления технологическими процессами, при построении различных специализированных цифровых устройств и машин.

Вычислительные возможности микро-ЭВМ оказались достаточными для создания на их основе в рамках ЭВМ четвертого поколения, нового по ряду эксплуатационных характеристик и способу использования типа вычислительных устройств - персональных ЭВМ , получивших в настоящее время широкое распространение.

В ЭВМ четвертого поколения достигается дальнейшее упрощение контактов человека с ЭВМ путем повышения уровня машинного языка, значительного расширения функций устройств (терминалов), используемых человеком для связи с ЭВМ, начинается практическая реализация голосовой связи с ЭВМ. Использование БИС позволяет аппаратурными средствами реализовывать некоторые функции программ операционных систем (аппаратурная реализация трансляторов с алгоритмических языков высокого уровня и др.), что способствует увеличению производительности машин.

Характерным для крупных ЭВМ четвертого поколения является наличие нескольких процессоров, ориентированных на выполнение определенных операций, процедур или на решение некоторых классов задач. В рамках этого поколения создаются многопроцессорные вычислительные системы с быстродействием в несколько десятков и даже сотен миллионов операций в секунду. К этому же поколению относятся и многопроцессорные управляющие комплексы повышенной надежности с автоматическим изменением структуры (автоматической реконфигурацией).

Примером крупных вычислительных систем, которые следует отнести к четвертому поколению, является многопроцессорный комплекс «Эльбрус-2» с суммарным быстродействием до 100 млн. опер/с, с системой команд, приближенной к языкам высокого уровня, стековой организацией обращений к памяти.

В 90-е годы прошлого века определились контуры нового, пятого поколения ЭВМ. В значительной степени этому способствовали публикации сведений о проекте ЭВМ пятого поколения, разрабатываемом ведущими японскими фирмами и научными организациями, поставившими перед собой цель захвата в 90-х годах японской промышленностью мирового лидерства в области вычислительной техники. Поэтому этот проект часто называют “японским вызовом”. Согласно этому проекту ЭВМ и вычислительные системы пятого поколения, помимо более высокой производительности и надежности при более низкой стоимости должны, обладать качественно новыми свойствами. В первую очередь к ним относятся возможность взаимодействия с ЭВМ при помощи языка, человеческой речи и графических изображений, способность системы обучаться, производить ассоциативную обработку информации, делать логические суждения, вести “разумную” беседу с человеком в форме вопросов и ответов. Вычислительные системы пятого поколения должны также “понимать” содержимое базы данных, которая при этом превращается в “базу знаний”, и использовать эти “знания” при решении задач. В настоящее время исследования по подобным проблемам ведутся и в России.

1.2 ХАРАКТЕРИСТИКИ ЭВМ

Электронная вычислительная машина — это комплекс технических и программных средств, предназначенный для автоматизации подготовки и решения задач пользователей. Выбирая ЭВМ для решения своих задач, пользователь интересуется функциональными возможностями технических и программных средств, начиная со следующих характеристик ЭВМ:

- технические и эксплуатационные характеристики ЭВМ (быстродействие и производительность, показатели надежности, достоверности, точности, емкость оперативной и внешней памяти, габаритные размеры, стоимость технических и программных средств, особенности эксплуатации и др.);

- характеристики и состав функциональных модулей базовой конфигурации ЭВМ; возможность расширения состава технических и программных средств; возможность изменения структуры;
- состав программного обеспечения ЭВМ и сервисных услуг (операционная система или среда, пакеты прикладных программ, средства автоматизации программирования).

Важнейшими характеристиками ЭВМ являются быстродействие и производительность. Эти характеристики тесно связаны. Быстродействие характеризуется числом определенного типа команд, выполняемых ЭВМ за одну секунду. Производительность — это объем работ (например, число стандартных программ), выполняемый ЭВМ в единицу времени.

Определение характеристик быстродействия и производительности представляет собой очень сложную задачу, не имеющую единых подходов и методов решения.

Одной из единиц измерения быстродействия была и остается величина, измеряемая в MIPS (Million Instructions Per Second — миллион операций в секунду). В качестве операций здесь обычно рассматриваются наиболее короткие операции типа сложения. MIPS широко использовалась для оценки больших машин второго и третьего поколений, но для оценки современных ЭВМ применяется достаточно редко по следующим причинам:

- набор команд современных микропроцессоров может включать сотни команд, значительно отличающихся друг от друга длительностью выполнения;
- значение, выраженное в MIPS, меняется в зависимости от особенностей программ;
- значение MIPS и значение производительности могут противоречить друг другу, когда оцениваются разнотипные вычислители (например, ЭВМ, содержащие сопроцессор для чисел с плавающей точкой и без такового).

При решении научно-технических задач в программах резко увеличивается удельный вес операций с плавающей точкой. Опять же для больших однопроцессорных машин в этом случае использовалась и продолжает использоваться характеристика быстродействия, выраженная в MFLOPS (Million Floating Point Operations Per Second — миллион операций с плавающей точкой в секунду). Для персональных ЭВМ этот показатель практически не применяется из-за особенностей решаемых на них задач и структурных характеристик ЭВМ.

Для более точных комплексных оценок существуют тестовые наборы, которые можно разделить на три группы:

- наборы тестов фирм-изготовителей для оценивания качества собственных изделий (например, компания Intel для своих микропроцессоров ввела показатель iCOMP-Intel Comparative Microprocessor Performance);
- стандартные универсальные тесты для ЭВМ, предназначенных для крупномасштабных вычислений (например, пакет математических задач Linpack, по которому ведется список TOP 500, включающий 500 самых производительных компьютерных установок в мире);
- специализированные тесты для конкретных областей применения компьютеров (например, для тестирования ПК по критериям офисной группы приложений используется тест Winstone97-Business,для группы «домашних компьютеров» — WinBench97-CPUMark32, а для группы ПК для профессиональной работы — 3DWinBench97-UserScene).

Результаты оценивания ЭВМ по различным тестам несопоставимы. Наборы тестов и области применения компьютеров должны быть адекватны.

Другой важнейшей характеристикой ЭВМ является емкость запоминающих устройств. Она измеряется количеством структурных единиц информации, которые одновременно можно разместить в памяти. Этот показатель позволяет определить, какой набор программ и данных может быть одновременно размещен в памяти.

Обычно отдельно характеризуют емкость оперативной памяти и емкость внешней памяти. Современные персональные ЭВМ могут иметь емкость оперативной памяти, равную 64 — 256 Мбайтам и даже больше. Этот показатель очень важен для определения, какие программные пакеты и их приложения могут одновременно обрабатываться в машине.

Емкость внешней памяти зависит от типа носителя. Так, емкость одной дискеты составляет 1,2; 1,4; 2,88 Мбайта в зависимости от типа дисковода и характеристик дискет. Емкость жесткого диска и дисков DVD может достигать нескольких десятков Гбайтов, емкость компакт-диска (CD-ROM) — сотни Мбайтов (640 Мбайт и выше) и т.д. Емкость внешней памяти характеризует объем программного обеспечения и отдельных программных продуктов, которые могут устанавливаться в ЭВМ. Например, для установки операционной среды Windows 2000 требуется объем памяти жесткого диска более 600 Мбайт и не менее 64 Мбайт оперативной памяти ЭВМ.

Надежность — это способность ЭВМ при определенных условиях выполнять требуемые функции в течение заданного времени (стандарт ISO (Международная организация стандартов)-2382/14-78).

Точность — возможность различать почти равные значения (стандарт ISO — 2382/2-76). Точность получения результатов обработки в основном определяется разрядностью ЭВМ, которая в зависимости от класса ЭВМ может составлять 32, 64 и 128 двоичных разрядов.

Во многих применениях ЭВМ не требуется большой точности, например при обработке текстов и документов, при управлении технологическими процессами. В этом случае достаточно воспользоваться 8- и 16-разрядными двоичными кодами. При выполнении же сложных математических расчетов следует использовать высокую разрядность (32, 64 и даже более). Программными способами диапазон представления и обработки данных может быть увеличен в несколько раз, что позволяет достигать очень высокой точности.

Достоверность — свойство информации быть правильно воспринятой. Достоверность характеризуется вероятностью получения безошибочных результатов. Заданный уровень достоверности обеспечивается аппаратно-программными средствами контроля самой ЭВМ. Возможны методы контроля достоверности путем решения эталонных задач и повторных расчетов. В особо ответственных случаях проводятся контрольные решения на других ЭВМ и сравнение результатов.

1.3 КЛАССИФИКАЦИЯ СРЕДСТВ ЭВТ

По виду представления обрабатываемой информации электронная вычислительная техника разделяется на аналоговую и цифровую.

В аналоговых вычислительных машинах (АВМ) обрабатываемая информация представляется соответствующими значениями аналоговых величин: тока, напряжения, угла поворота какого-то механизма и т.п. АВМ обеспечивают приемлемое быстродействие, но не очень высокую точность вычислений (0,001 — 0,01). Подобные машины мало распространены. Они используются в основном в проектных и научно-исследовательских учреждениях в составе стендов по отработке новых образцов техники. По назначению их можно рассматривать как специализированные вычислительные машины.

В настоящее время под словом ЭВМ обычно понимают цифровые вычислительные машины, в которых информация кодируется двоичными кодами чисел. Именно эти машины из-за их универсальности являются самой массовой вычислительной техникой.

По быстродействию ЭВМ можно разделить на:

- суперЭВМ для решения крупномасштабных вычислительных задач, для обслуживания крупнейших информационных банков данных;
- большие ЭВМ для комплектования ведомственных, территориальных и региональных вычислительных центров;
- средние ЭВМ широкого назначения для управления сложными технологическими производственными процессами. ЭВМ этого типа могут

использоваться и для управления распределенной обработкой информации в качестве сетевых серверов;

- персональные и профессиональные ЭВМ, позволяющие удовлетворять индивидуальные потребности пользователей. На базе этого класса ЭВМ строятся автоматизированные рабочие места (АРМ) для специалистов различного уровня;
- встраиваемые микропроцессоры (микропроцессорные системы), осуществляющие автоматизацию управления отдельными устройствами и механизмами.

С развитием сетевых технологий все больше начинает использоваться другой классификационный признак, отражающий место и роль ЭВМ в сети, а именно:

- мощные машины и вычислительные системы для управления сетевыми хранилищами информации;
- кластерные структуры;
- серверы;
- рабочие станции;
- сетевые компьютеры.

Мощные машины и вычислительные системы предназначаются для обслуживания крупных сетевых банков данных и банков знаний. По своим характеристикам их можно отнести к классу суперЭВМ, но в отличие от них они являются более специализированными и ориентированными на обслуживание мощных потоков информации.

Кластерные структуры представляют собой многомашинные распределенные вычислительные системы, объединяющие несколько серверов. Это позволяет гибко управлять ресурсами сети, обеспечивая необходимую производительность, надежность, готовность и другие характеристики.

Серверы — это вычислительные машины и системы, управляющие определенным видом ресурсов сети. Различают файл-серверы, серверы приложений, факс-серверы, почтовые, коммуникационные, Web-серверы и др.

Термин “рабочая станция” отражает факт наличия в сетях абонентских пунктов, ориентированных на работу профессиональных пользователей с сетевыми ресурсами. Этот термин как бы отделяет их от ПЭВМ, обеспечивающих работу основной массы непрофессиональных пользователей, работающих обычно в автономном режиме.

Сетевые компьютеры представляют собой упрощенные персональные компьютеры, вплоть до карманных ПК. Их основным назначением является обеспечение доступа к сетевым информационным ресурсам. Вычислительные возможности у них достаточно низкие.

Высокие скорости вычислений, обеспечиваемые ЭВМ различных классов, позволяют перерабатывать и выдавать все большее количество информации, что, в свою очередь, порождает потребности в создании связей между отдельно используемыми ЭВМ. Поэтому все современные ЭВМ в настоящее время имеют средства подключения к сетям связи и объединения в системы.

1.4 СТРУКТУРЫ ЭВМ

1.4.1 ОБОБЩЕННАЯ СТРУКТУРА ЭВМ

Простейшая структура ЭВМ с локальными шинами между ее устройствами, приведена на рисунке 1.4.1.

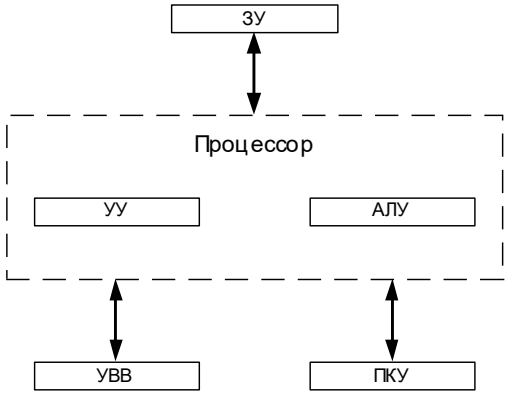


Рисунок 1.4.1-Обобщенная структура ЭВМ

В состав ЭВМ входят:

- оперативное запоминающее устройство (ОЗУ, более короткое обозначение- оперативная память ОП);
- процессор;
- устройство ввода- вывода (УВВ, другое обозначение- периферийное устройство ПУ);
- пульт контроля и управления (ПКУ).

Процессор предназначен для обработки информации. Он состоит из 2-х частей: УУ - устройство управления (управляющий автомат), и АЛУ - арифметико-логическое устройство.

Обработку информации процессор осуществляет под управлением программы, хранящейся в ОЗУ. В ОЗУ наряду с программой также хранятся и данные, подлежащие обработке. Программа и данные поступают из ОЗУ в процессор по каналу связи между ОЗУ и процессором, называемым в вычислительной технике шиной. Такие же шины соединяют процессор и с другими устройствами ЭВМ.

УВВ предназначено для ввода программ и данных в ОЗУ, то есть они сначала подготавливаются либо в виде перфокарт (ПФК), перфолент (ПФЛ), либо в виде магнитных лент, магнитных дисков и т.п., а затем вводятся в ОП машины. После этого программа запускается на обработку. В современных машинах диалогового режима данные в ОП могут заноситься и непосредственно с клавиатуры.

ПКУ предназначен для ручного пуска различного рода тестовых программ, контроля хода вычислительного процесса или функционирования устройств ЭВМ.

1.4.2 СТРУКТУРА ЭВМ НА ОСНОВЕ ОБЩЕЙ ШИНЫ

При организации ЭВМ на основе общей шины (ОШ) взаимодействие между ее устройствами осуществляется через общую шину, к которой подключены все устройства, входящие в состав ЭВМ.

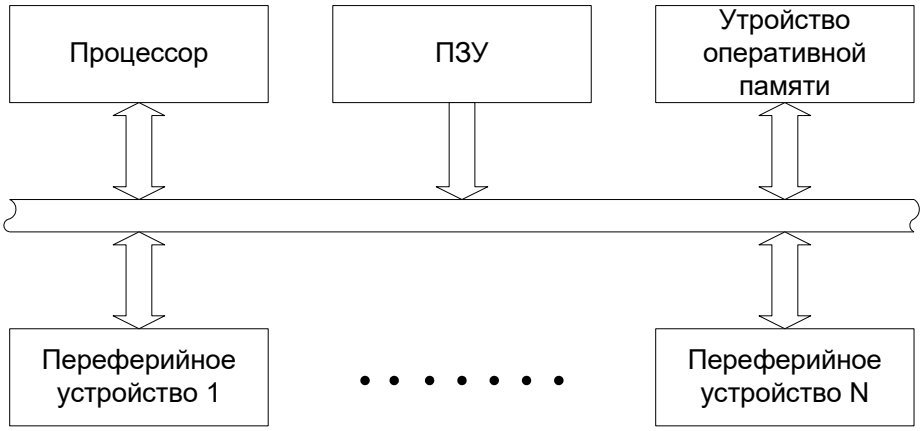


Рисунок 1.4.2- Структура ЭВМ на основе ОШ

Взаимодействие между всеми устройствами ЭВМ осуществляется в режиме разделения времени общей шины (т.е. поочередно). Такой способ не обеспечивает (принципиально) высокой пропускной способности, ввиду чего производительность ЭВМ ниже, чем при наличии локальных шин между различными устройствами ЭВМ. Однако простота реализации и возможность построения ОШ с высокой пропускной способностью обеспечили широкое использования такой структуры в персональных ЭВМ (ПК) и микропроцессорных системах (МПС).

2 Архитектура классической ЭВМ

2.1 Принцип программного управления

Принцип программного управления впервые был реализован в ЭВМ “Марк-1”. Он заключается в том, что алгоритм вычислений (например, вычисление некоторого выражения) преобразуется в упорядоченную последовательность команд, преобразующих исходные данные (операнды) в результат. Таким образом действия, предписанные алгоритмами, закладываются в команды (например действия по сложению, вычитанию, умножению и делению чисел, логическим операциям над ними и т.д.). Последовательность команд называется программой. Программа управляет ходом вычислительного процесса.

Пусть, например, необходимо вычислить выражение: $Y = \dfrac{a \cdot b}{(a+b) \cdot c}$.

Возможная программа его вычисления содержит следующие команды:

1-я команда: умножить операнд a на b;

2-я команда: сохранить результат умножения **(**a*b**)** в ОП;

3-я команда: сложить операнды a и b ;

4-я команда: умножить результат (a+b) на c;

5-я команда: считать из ОП (a*b***)**;

6-я команда: разделить результат (a*b***)** на результат (a+b)* c.

Если числа представлены в двоичной системе счисления, и команды также закодированы двоичным кодом, то для реализации программы можно ввести следующую систему команд:

К ОП	***1-й операнд***	***2-й операнд***

где КОП-код операции, или закодированные в двоичной системе счисления операции (+,-, /, * и т.д.), выполняемые процессором.

Такой подход к реализации команд приводил к очень длинным программам, так как в перфоленточных устройствах, используемых в первых ЭВМ для ввода программ, отсутствовала возможность возврата к ранее выполненным участкам программ. Пусть, например, нам необходимо вычислить выражение:

$$Y = \sum_{i=0}^{n} a_i b_i = a_0 b_0 + a_1 b_1 + a_2 b_2 + \dots$$

Программа вычислений в предложенной системе команд будет следующей:

1-я команда: умножить a_0 на b_0;

2-я команда: умножить a_1 на b_1;

3-я команда: сложить результат 1-й команды с результатом 2-й команды;

4-я команда: умножить a_2 на b_2;

5-я команда: сложить результат 3-й команды с результатом 4-й команды:

6-я команда: умножить a_3 на b_3**;**

7-я команда: сложить результат 3-й команды с результатом 4-й команды и т.д.

При использовании предложенной системы команд программа будет состоять из n - команд умножения и n - команд сложения, всего - из 2n-команд. Большое количество команд обусловлено тем, что нет возможности оперативного возврата к некоторым участкам программы, которые могли бы выполняться многократно.

2.2 Принцип хранимой в памяти программы

Принцип хранимой в памяти программы был предложен фон Нейманом в 1945 году. Этот принцип стал основой современных машин. В соответствии с этим принципом команды хранятся в памяти, также как и данные. При этом под программу отводится одна отдельная область памяти, под данные - другая область. В командах указываются не операнды, а их адреса, то есть номера ячеек памяти ОЗУ, где они помещаются. Для вызова команд из ОП также надо указывать их адреса в ОП. При такой организации можно многократно вызывать из памяти одну и ту же команду или последовательность из нескольких команд (подпрограмму) и одни и те же данные. Кроме этого, над командами и над данными можно производить операции, так как они с точки зрения обработки становятся равноценными. Структура команды для ЭВМ, организованной в соответствии с принципом фон Неймана (фон Неймановская машина), будет следующей:

К ОП	***Адрес операнда***

Такой тип команды оказался намного более универсальным и напряду с ранее приведенным он широко используется в современной вычислительной технике.

Программа вычисления выражения: $Y = \sum_{i=1}^{n} a_i b_i$ при использовании команд последнего типа намного сокращается.

1-я команда: i=0;

2-я команда: умножение $a_i * b_i = X_i$;

3-я команда: сложение $Y_i + X_i = Y_i$;

4-я команда: i:=i+1;

5-я команда: i>n?. Если нет, то переход на 2-ю команду;

6-я команда: Конец.

2.3 Обобщенный формат команд

Команды в ***ЦВМ*** могут быть одноадресными, двухадресными и трехадресными (в машинах с так называемой естественной адресацией команд).

Одноадресная команда:

КОП	А -Адрес операнда

Двухадресная команда:

КОП	А1-Адрес 1-го операнда	А2- Адрес 2-го операнда

Трехадресная команда:

КОП	А1-Адрес 1-го операнда	А2-Адрес 2-го операнда	А3-Адрес 3-го операнда

Команда состоит из операционной части- кода операции (КОП) и адресной части. В операционной части указывается тип выполняемой операции в виде двоичного числа. В адресной части указывается адрес ячейки памяти, в которой размещается операнд (одноадресная команда). Если в команде указывается адреса 1-го и 2-го операнда, то такая команда называется двухадресной. В трехадресной машине указывается еще и адрес результата, то есть ячейка ОП, куда помещается результат.

Какая из систем лучше? В современных машинах большого класса могут сочетаться все типы. Приведенные типы команд относятся к так называемым машинам с естественной адресацией, когда команды из программы выбираются последовательно одна за другой. Адресация производится с помощью счетчика команд СчК (PC- Program Count). Однако существовали машины и с принудительной адресацией, в которых очередная команда выбиралась по адресу, указанному в предыдущей команде (такой способ адресации сохранен в настоящее время только в так называемых микропрограммных устройствах управления).

Структура команд такой машины:

КОП	А1	А2	Ар	Аск

где $А_р$- адрес результата;

Аск -адрес следующей команды .

Если операндов два, и еще существует поле адреса результата, то команда становится четырехадресной:

2.4 Способы адресации команд

Процессоры в зависимости от реализации УУ бывает двух видов: с принудительным порядком выполнения команд (принудительной адресацией команд) и с естественным порядком выполнения команд (естественной адресацией команд).

2.4.1 Процессоры с принудительным порядком выполнения команд

Структура процессора с принудительной адресацией команд приведена на рисунке 2.4.1.

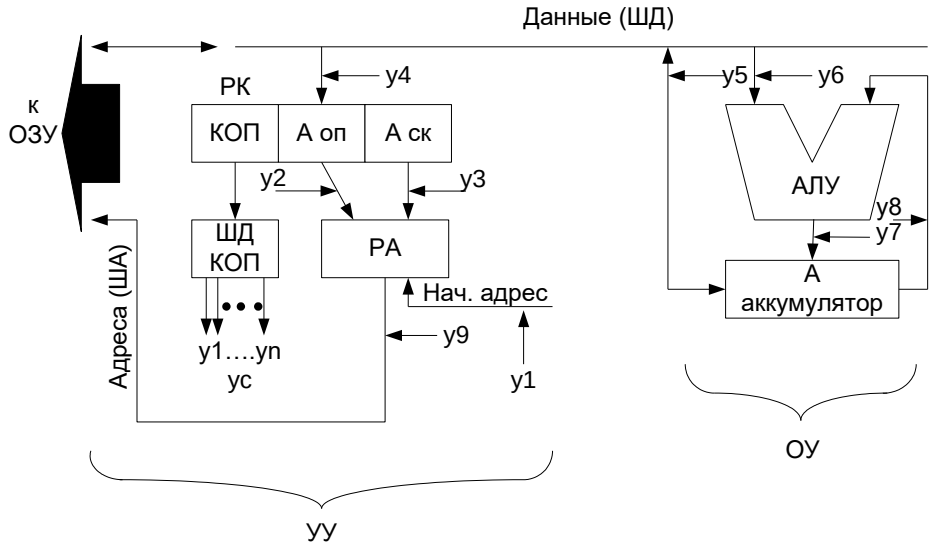


Рисунок 2.4.1- Структура процессора с принудительной адресацией

Процесс выполнения команд процессором следующий: в начальный момент в регистр команд (РК) заносится адрес первой выполняемой команды (по сигналу “Сброс” или “Пуск” или каким либо иным способом). По этому адресу считываются команда, которая содержат код операции (КОП), адрес операнда (Аоп), а так же адрес следующей команды (Аск). Поле КОП команды поступает на схему формирования управляющих сигналов (ДшКОП- дешифратор КОП) которая вырабатывает нужную последовательность управляющих сигналов у1...yn, необходимых для выполнения команды в процессоре.

Адрес операнда через РА задает номер ячейки ОП, в которой он хранится. Операнд, считанный из ячейки памяти с заданным адресом, поступает на обработку в АЛУ.

Рассмотрим следующий пример. Положим, что процессор имеет следующую систему команд в машинных кодах (для более краткой записи представим ее в шестнадцатеричной системе счисления):

01Н- вызов операнда из ОЗУ в аккумулятор;

02Н- запись содержимого А в ОЗУ;

1АН - команда сложения;

00Н – остановка выполнения программы.

Пусть необходимо составить программу сложения 2-х чисел, находящихся в ячейках ОЗУ с адресами 0841Н и 0842Н и записать результат в ячейку 0843Н. Программа хранится в ячейках памяти с начальным адресом 1300Н. Ширина выборки команд и данных из ОЗУ - 1 байт.

№ яч	КОП	Аоп	Аск	Комментарий
1300	01	0841	1305	Вызов 1-го операнда из ОЗУ и переход к считыванию следующей команды из ячейки ОП с номером 1305Н.
1305	1А	0842	130А	Вызов 2-го операнда, сложение и переход к считыванию следующей команды из ячейки ОП с номером 130АН.

130A	02	0843	130F	Запись результата в ОЗУ и переход к ячейке 130FH.
130F	00	0000	0000	Остановка.

Нетрудно подсчитать, что при использовании принудительной адресации команд длина программы составляет 20 байт.

Процессоры с естественной адресацией команд Структура процессора с естественной адресацией команд приведена на рисунке 2.4.2.

Рисунок 2.4.2- Структура процессора с естественной адресацией команд

В таких процессорах РК не имеет поля с адресом команд. Адрес следующей команды образуется путем добавления единицы к адресу выполняемой команды. Для этой цели служит счетчик команд СчК (PC- Program Counter) , который предварительно загружают начальным адресом, по которому выбирается первая команда. По окончании выполнения команды в СчК автоматически добавляется число единиц равное количеству байт выполняемой команды, затем выбирается следующая команда и т.д.

Предыдущая программа для этого типа процессора будет иметь вид:

№ яч. памяти	КОП	Аоп	Комментарий
1300:	01	08 41	Вызов 1-го операнда из ОЗУ в аккумулятор и переход к считыванию следующей команды из ячейки ОП с номером 1303Н.
1303:	1A	08 42	Вызов 2-операнда, сложение его с 1- м операндом и переход к считыванию следующей команды из ячейки ОП с номером 1306Н.
1306:	02	08 43	Запись вычисленного результата в ячейку ОП с адресом 0843Н и переход к считыванию следующей команды.
1309:	00	00 00	Остановка.

Нетрудно подсчитать, что длина этой программы составляет 12 байт.

В процессорах с естественной адресацией длина программы и требуемый под нее объем памяти сокращается, однако система команд усложняется, так как для организации ветвления программ требуется специальные команды - условных и безусловных переходов. Первое обстоятельство оказалось сильнее и традиционные машины выполняются по второму способу.

2.5 СПОСОБЫ АДРЕСАЦИИ ОПЕРАНДОВ

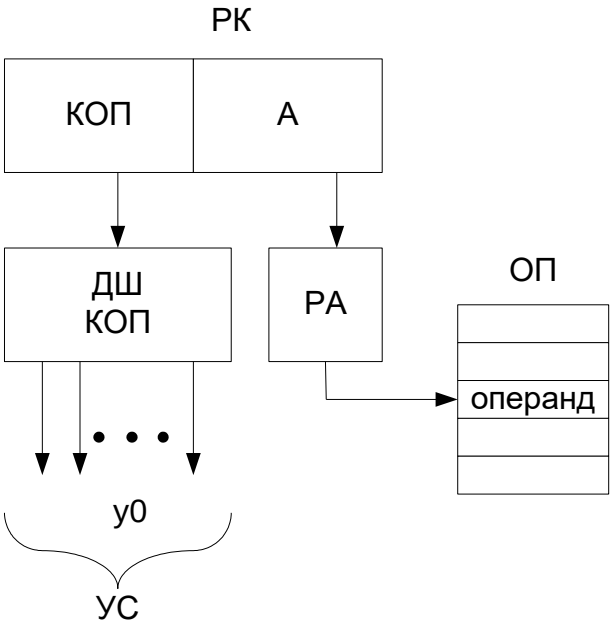
В современных ЭВМ используется большое число способов адресации операндов. Рассмотрим наиболее часто используемые.

2.5.1 ПРЯМАЯ АДРЕСАЦИЯ

При прямой адресации адрес операнда указывается в адресной части команды. Поле адреса может быть одно, двух и трехадресным. Длина адресного поля n_A должна быть

такой, чтобы перекрывать все адресное пространство – $n_A=\log_2 M$, где М-емкость памяти в байтах.

Рисунок 2.5.1- Порядок выборки операнда при прямой адресации



Формат одноадресной команды следующий:

КОП	Адрес операнда

Формат двухадресной команды:

КОП	А1	А2

n_{A1} n_{A2}

Формат трехадресной команды:

КОП	А1	А2	Ар

n_{A1} n_{A2} n_p

где A_p- адрес результата.

Размер команд, использующих прямую адресацию – большой, поэтому выполняется они достаточно медленно.

2.5.2 КОСВЕННАЯ АДРЕСАЦИЯ

При косвенной адресации в адресной части команды указывается адрес ячейки памяти (ОЗУ или СОЗУ) в которой находится адрес операнда (косвенная адресация - это адресация адреса).

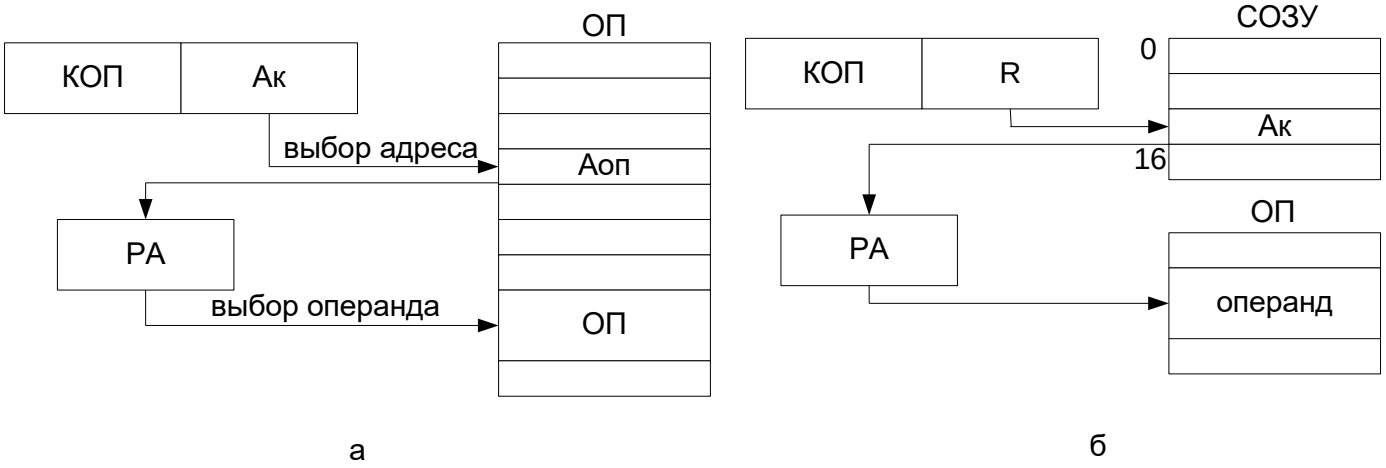


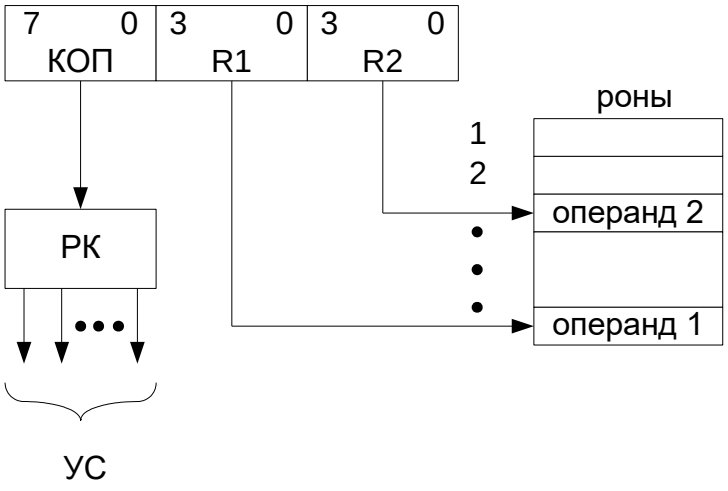
Рисунок 2.5.2- Порядок выборки операнда при косвенной адресации: а- при адресации через ОП; б- при адресации через СОЗУ

Такая адресация используется в машинах, имеющих малую разрядность, то есть в мини и микро- ЭВМ. Для адресации операнда требуется как минимум два шинных цикла: 1-й для выборки адреса, второй - для выборки операнда по этому адресу. Часто косвенный адрес хранится во внутренней памяти процессора, состоящей из регистров двойной длины.

2.5.3 РЕГИСТРОВАЯ АДРЕСАЦИЯ

Регистровая адресация является укороченной. В поле адреса указываются адреса ячеек сверхоперативной памяти (СОЗУ), число которых невелико. Такой способ адресации позволяет сократить длину команды и увеличить скорость выполнения операции, так как СОЗУ является быстродействующей памятью, выполняемой на быстрых регистрах. Эти регистры являются частью процессора и называются регистрами общего назначения (РОН). Следующий рисунок поясняет порядок выборки операндов при использовании 2- х адресной команды: R1- адрес первого операнда, R2- - адрес второго операнда.

Рисунок 2.5.3- Порядок выборки операндов при регистровой адресации: R1- адрес 1-



го операнда, R2- адрес второго операнда

2.5.4 НЕПОСРЕДСТВЕННАЯ АДРЕСАЦИЯ

В поле адреса команд находится не адрес, а сам операнд. В этом случае нет необходимости обращаться за операндом в память. Используется для хранения констант.

Непосредственный операнд может иметь любую длину (байт, слово, 2-е слово). Этим определяется длина команды. Формат команды при непосредственной адресации следующий:

ОП	К	Непосредственный операнд

2.5.5 НЕЯВНАЯ АДРЕСАЦИЯ

Неявная (подразумеваемая) адресация. В команде нет явных указаний об адресе операнда, они подразумеваются, но фактически их адреса указаны в КОП команды. Это самая короткая адресация. Используется в микроЭВМ.

2.5.6 ОТНОСИТЕЛЬНАЯ АДРЕСАЦИЯ

Адрес операнда определяется как сумма содержимого адресного поля команды и некоторого числа, называемого базовым адресом. Для этого в команде предусмотрено поле В для указания адреса базового регистра. Поле команды, в котором находится адрес операнда называют смещением.

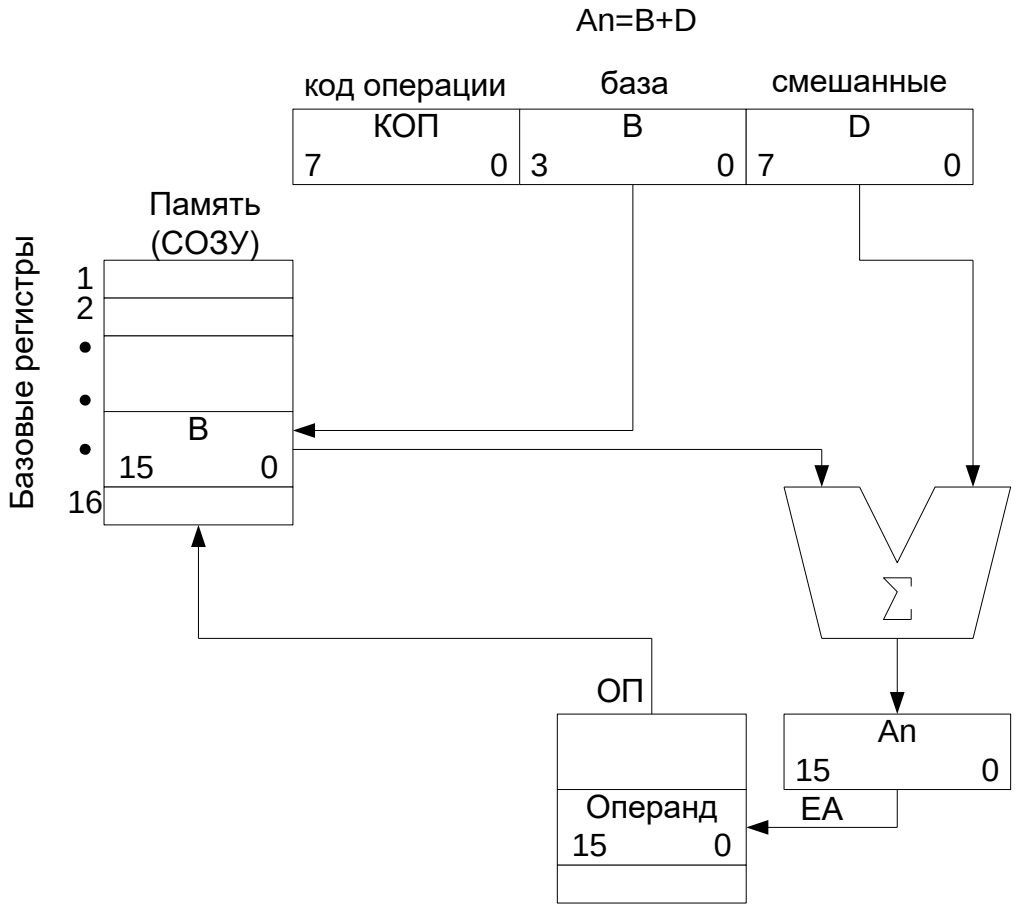


Рисунок 2.5.6-Формирование адреса операнда при относительной адресации

Полученный адрес **EA=[B]+D** называется эффективным или исполнительным адресом. Прямые скобки при В **([B])** – означают, что первое слагаемое EA берется по адресу **B.** При выборке некоторого участка данных базовый адрес является неизменным. Адресация ячеек памяти относительно базового адреса осуществляется полем смещения.

Относительная адресация обеспечивает перемещаемость программ, то есть возможность передвижения программ в памяти без изменений внутри самой программы, что не требует перетрансляции программы. Для назначения новых адресов, изменяется только значение B

2.5.7 ИНДЕКСНАЯ (АВТОИНКРЕМЕНТНАЯ И АВТОДЕКРЕМЕНТНАЯ) АДРЕСАЦИЯ

При обработке больших массивов данных, выбираемых последовательно друг за другом, нет смысла каждый раз обращаться в память за новым адресом. Для этого достаточно автоматически менять содержимое специального регистра, называемого индексным, чтобы выбирать последовательно размещенные данные. Индексный регистр является косвенным. Его загружают начальным адресом массива (специальной командой). Дальнейшая адресация осуществляется путем автоматического добавления или вычитания единицы или шага адреса из его содержимого.

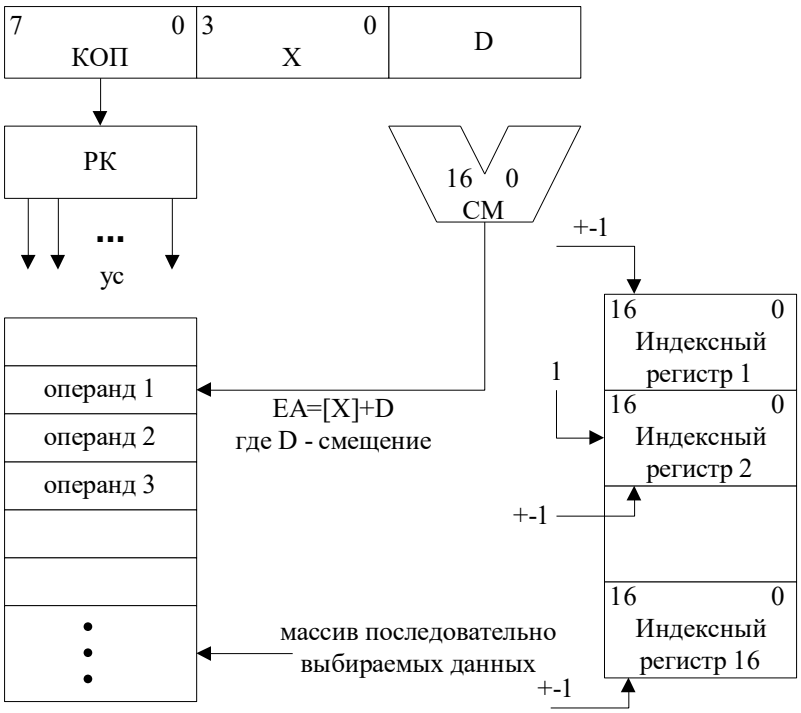


Рисунок 2.5.7- Формирование адреса операнда при индексной адресации

В некоторых процессорах применяют более сложную адресацию, которая сочетает индексную адресацию с базовым смещением.

Часто в команду с индексной адресацией включают признак, определяющий шаг индексации T (T=1,2,4 и т.д.), что позволяет осуществлять адресацию массивов через байт, слово, двойное слово и т.д.

В современных процессорах (например в Intel 80386 и выше) применяют все возможные сочетания из базового адреса, индексного адреса, относительного адреса и шага. Например:

- Индексная адресация с шагом. Содержимое индексного регистра умножается на шаг и суммируется со смещением- EA=[X]·T+D, где T - величина шага;
- Базово- индексная адресация EA=[B]+[X];
- Базово- индексная адресация с шагом EA=[B]+[X]·T;
- Базово- индексная адресация со смещением EA=[B]+[X]+D:
- Базово- индексная адресация со смещением и шагом EA=[B]+[X]·T+D.

3. Запоминающие устройства ЭВМ

КЛАССИФИКАЦИЯ ЗУ

Запоминающие устройства (ЗУ) предназначены для хранения информации, а именно - программ и данных.

ЗУ подразделяются на:

- сверхоперативные;
- оперативные;
- внешние.

Оперативной память – это основная память машины. В соответствии с принципом фон Неймана она предназначена для хранения программы (программ в многозадачном режиме), выполняемой ЭВМ в данный момент времени и необходимых для нее данных.

Назначение внешнего ЗУ -хранение массивов информации.

Сверхоперативное ЗУ (СОЗУ) обеспечивает увеличение быстродействия ЭВМ, благодаря использованию команд меньшей (сокращенной) длины и более быстрой элементной базы ее регистров.

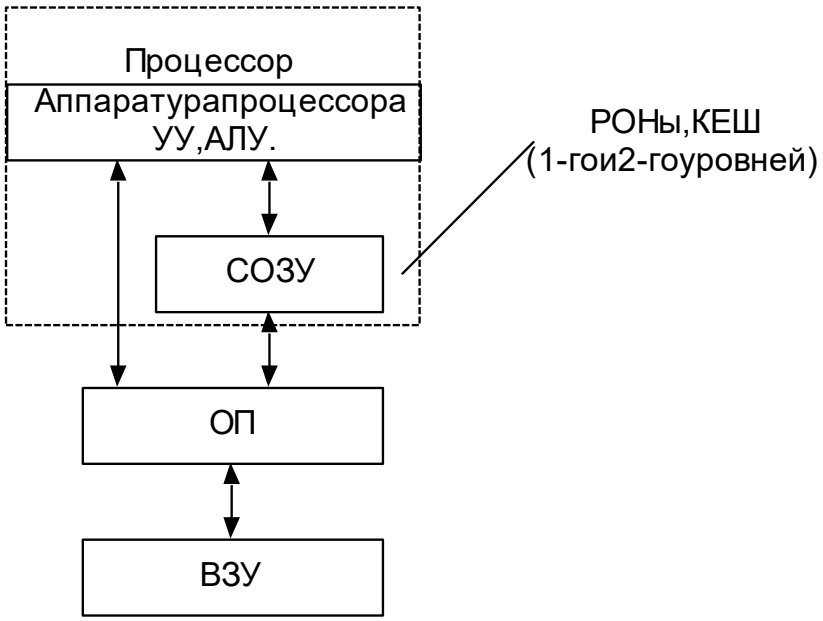


Рисунок 3.1- Иерархическая структура ЗУ

Внешние запоминающие устройства (ВЗУ) чаще всего выполняются на барабанах, магнитных и оптических дисках, ленточных накопителях.

Первые два типа ВЗУ называют устройствами прямого доступа (циклического доступа). Магнитные и оптические поверхности этих устройств непрерывно вращаются, чем обеспечивается быстрый доступ к хранимой информации (время доступа этих устройств составляет от нескольких мс до десятка мс). Накопители на магнитных лентах (МЛ) называют устройствами последовательного доступа, из-за последовательного просмотра участков носителя информации (время доступа этих устройств составляет от нескольких секунд до нескольких минут).

Оперативная память делится на:

-оперативные ЗУ (ОЗУ) (Оперативные ЗУ с произвольной выборкой ЗУПВ);

-постоянные ЗУ (ПЗУ).

Назначение и функции оперативных ЗУ: запись и чтение информации.

Назначение и функции ПЗУ – только чтение информации.

Выполняются на ферритовых сердечниках, полупроводниковых микросхемах, магнитных доменах (тонких пленках) и др. Время обращения к памяти составляет от нескольких нс до нескольких мкс.

Сверхоперативная память выполняется на элементной базе процессора, входит в структуру процессора и позволяет значительно повысить его производительность.

ОСНОВНЫЕ ХАРАКТЕРИСТИКИ ЗУ

1. Основная характеристика ЗУ (любого типа) – емкость памяти. Определяет максимальное количество информации, которое может в ней храниться. Емкость может измеряться в битах, байтах или машинных словах. Наиболее распространенной единицей измерения является байт. При большом размере памяти ее емкость выражают в килобайтах (Кбайт) – 1024 байт, в мегобайтах (Мбайт) – миллион байт (точнее 1024*1024 байт), в гигобайтах (Гбайт) – миллиард байт.
2. Время обращения к памяти. Время обращения при чтении:

$$t_0^{чт} = t_д + t_{чт} + t_{рег}, \text{ где}$$

$t_д$ - время доступа (подготовительное время) - промежуток времени между началом операции обращения и моментом начала процесса чтения;

$t_{чт}$ - продолжительность физического процесса считывания;

$t_{рег}$ - время регенерации (восстановления), если в процессе чтения информации произошло ее разрушение.

Время обращения при записи:

$$t^{зп}_0 = t_д + t_п + t_{зп}, \text{ где}$$

$t_п$ - время подготовки, расходуемое на приведение запоминающих элементов в исходном состоянии, если это необходимо;

$t_{зп}$ - время, необходимое для физического изменения состояния запоминающих элементов при записи информации.

3. Цикл памяти. Принимается равным минимальному допустимому интервалу между двумя обращениями в память:

$$t_ц = \max\left(t_0^{чт}, t_0^{зп}\right).$$

Положим, что процессы чтения и записи имеют следующие временные диаграммы:

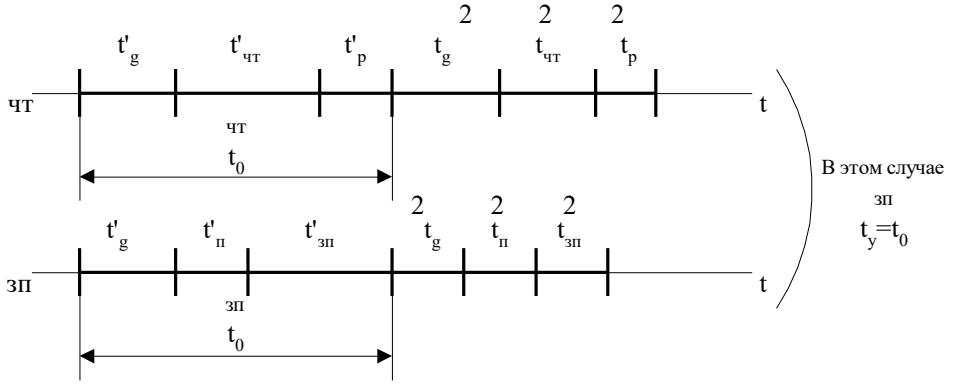


Рисунок 3.2 – Выбор значения цикла памяти $t_ц$

3.3 СТРУКТУРА ОЗУ С ПРОИЗВОЛЬНОЙ ВЫБОРКОЙ (ЗУПВ)

В оперативных ЗУ с произвольной выборкой (ЗУПВ) запись или чтение в/из памяти осуществляется по адресу, указанному регистром адреса (РА). Чтение или запись слова осуществляется за один цикл. Информация, необходимая для осуществления процесса записи- чтения поступает из процессора, а именно: адрес, данные и управляющие сигналы.

Адресная часть с процессора сначала поступает на регистр адреса (РА), а с него- на дешифратор адреса ДшА, который выбирает строку запоминающего массива (номер ячейки памяти). По сигналу запись (Зп) производится запись данных в заданную ячейку памяти.

Структура ОЗУ имеет следующий вид:

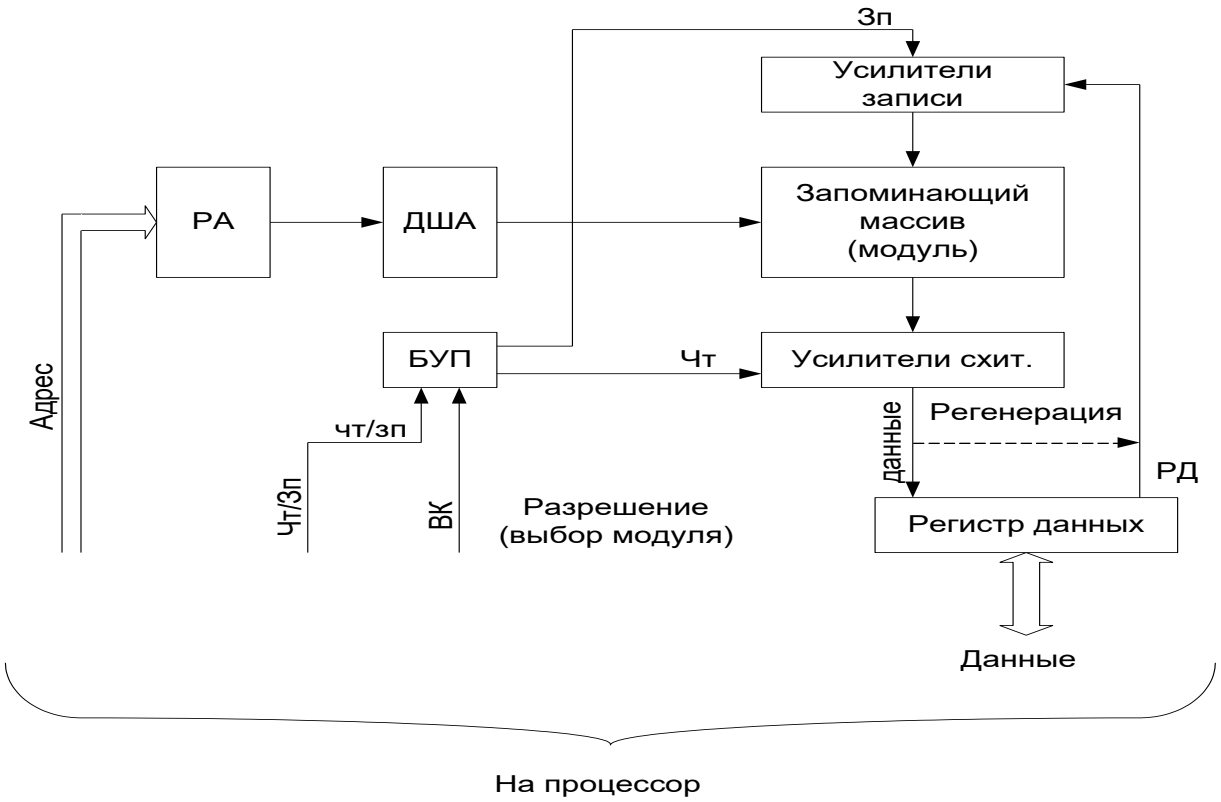


Рисунок 3.3- Структура ЗУПВ

Запоминающий массив содержит множество одинаковых запоминающих элементов В памяти статического типа в их качестве используются электронные триггеры, в динамической памяти- полевые транзисторы, работающие на принципе накопления заряда в области затвор-исток.

3.4 Особенности организации динамической памяти

Структура микросхем динамической памяти (DRAM) в целом близка к структуре статической памяти. Для уменьшения количества выводов (а следовательно, габаритов и стоимости), в микросхемах динамической памяти (DRAM) используется мультиплексированная ША. Полное количество разрядов ША, подаваемое на микросхему DRAM делится на две части- адрес строки и адрес столбца. При адресации ячеек DRAM эти части адреса, последовательно во времени, подаются на адресные входы микросхемы в сопровождении соответственно стробов адреса строки (RAS) и столбца (CAS) (см. рисунок 3.4.1).

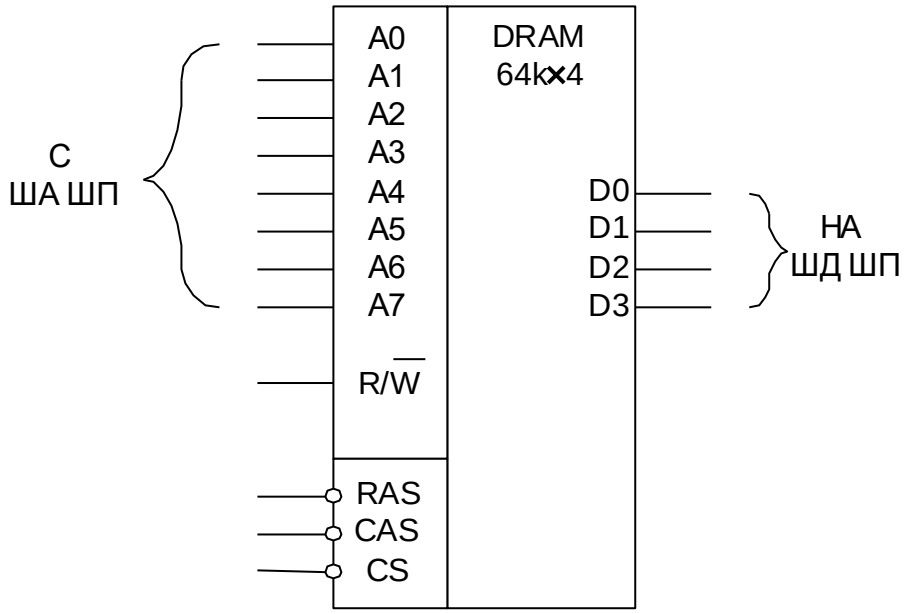


Рисунок 3.4.1- УГО микросхемы DRAM 64*4

Временные диаграммы ввода адреса запоминающего элемента микросхемы DRAM приведены на рисунке 3.4.2.

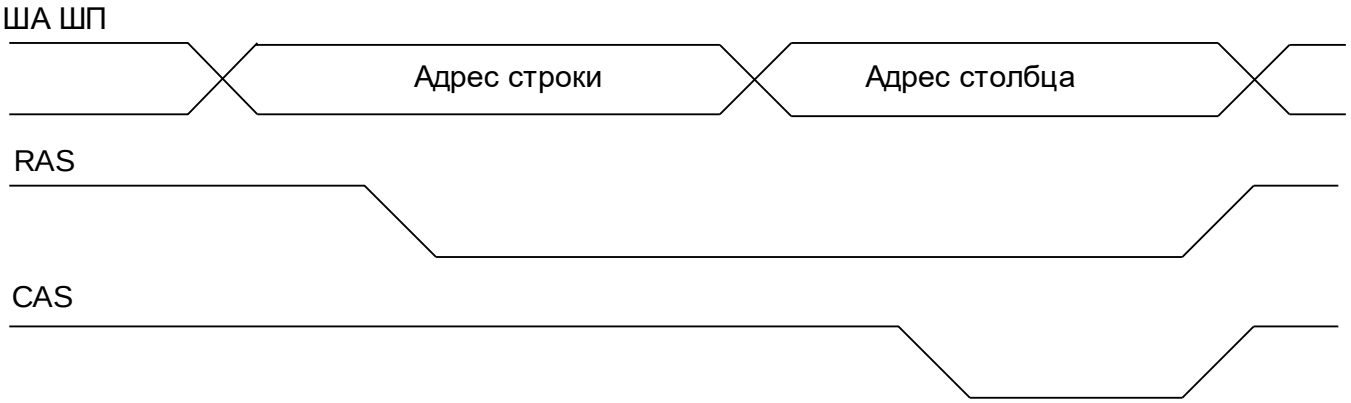


Рисунок 3.4.2 – Временные диаграммы сигналов ввода адреса в микросхему DRAM

Разделение полного адреса запоминающего элемента и последовательную выдачу его на микросхему осуществляет мультиплексор, являющийся частью контроллера динамической памяти.

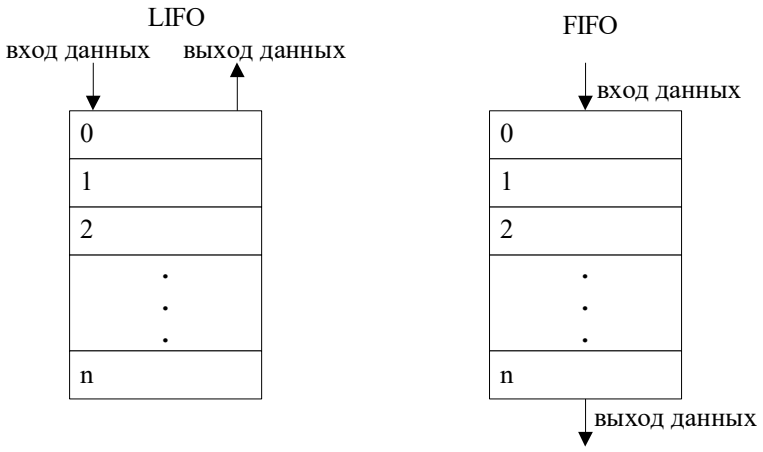
Матрица элементов памяти (МЭП) микросхемы DRAM разбита на строки, количество которых равно 2^n, где n- количество разрядов адреса строки или столбца. При вводе адреса строки выбранная строка МЭП считывается в регистр-защелку статического типа, входящего в состав микросхемы DRAM. При считывании строки ее содержимое разрушается, но копия содержимого строки оказывается записанной в регистр- защелку.

Подача адреса столбца в сопровождении строба CAS выбирает в регистре- защелке, в зависимости от организации микросхемы DRAM, бит, тетраду, байт и т.д. При появлении сигнала чтения выбранная информация выдается на ШД после чего записывается на прежнее место в строку МЭП.

При записи информация, поступившая на микросхему DRAM с ШД, записывается сначала в соответствующие разряды регистра- защелки, после чего его содержимое переписывается в прежнюю строку микросхемы DRAM.

3.5 ОЗУ **МАГАЗИННОГО ТИПА (СТЕКОВАЯ ПАМЯТЬ)**

Магазинная (стековая) память организуется по принципу “последний пришел, первый вышел” (LIFO), или “первый пришел, первый вышел” (FIFO). Такая организация памяти является фактически безадресной. Однако регистр адреса в такой памяти имеется и называется указателем стека (УС) (SP-Steak Pointer). Принцип организации стековой памяти показан на рисунке 3.5.1.



1.1.1.1.1 Рисунок 3.5.1-. Принцип организации стековой памяти

В первом типе памяти новое слово заносится в верхнюю ячейку, ранее занесенные данные проталкиваются вниз. При считывании наоборот, последнее слово выталкивается вверх первым.

В случае организации типа FIFO новое слово заносится в верхнюю ячейку, ранее записанные слова выталкиваются вниз.

Чаще используют память типа “последний пришел, первый вышел”. Организуется следующим образом:

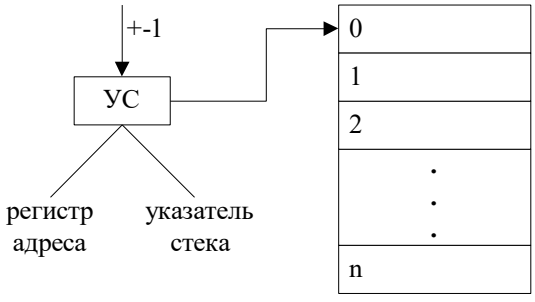


Рисунок 3.5.2- Адресация стека типа LIFO с помощью УС

Перед началом работы в указатель стека занося начальный адрес. Дальнейшая адресация осуществляется автоматически при выполнении операции записи- чтения путем увеличения- уменьшения адреса на единицу. Физический процесс записи и считывания данных происходит точно так же, как в обычной памяти с произвольным обращением.

Возможные изменения состояния УС стековой памяти типа LIFO при записи-чтении показаны на следующих рисунках:

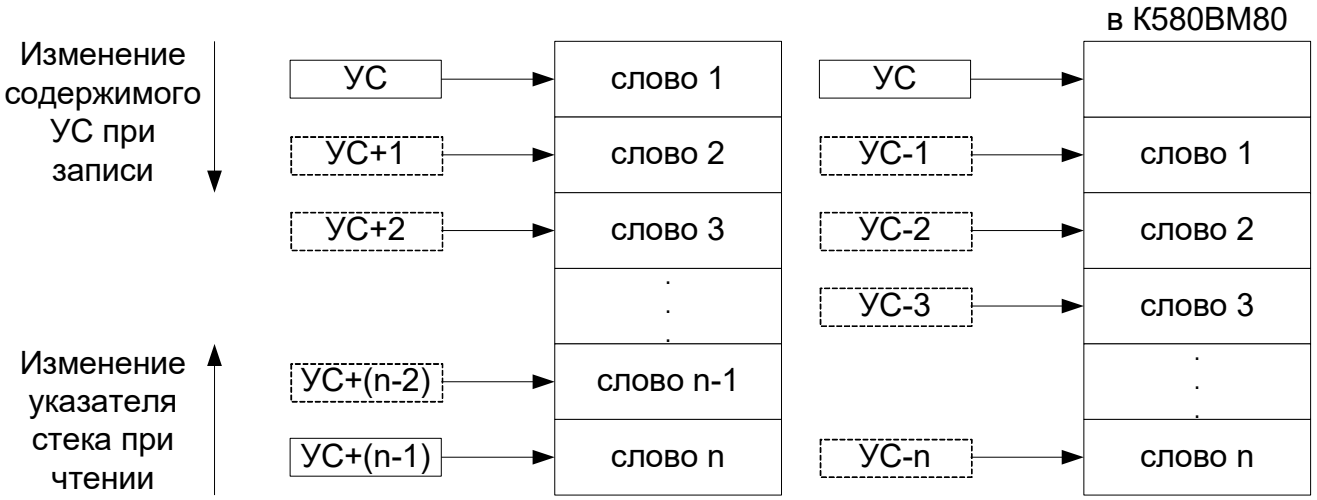


Рисунок 3.5.3- Изменение состояния УС при записи и чтении стековой памяти типа LIFO

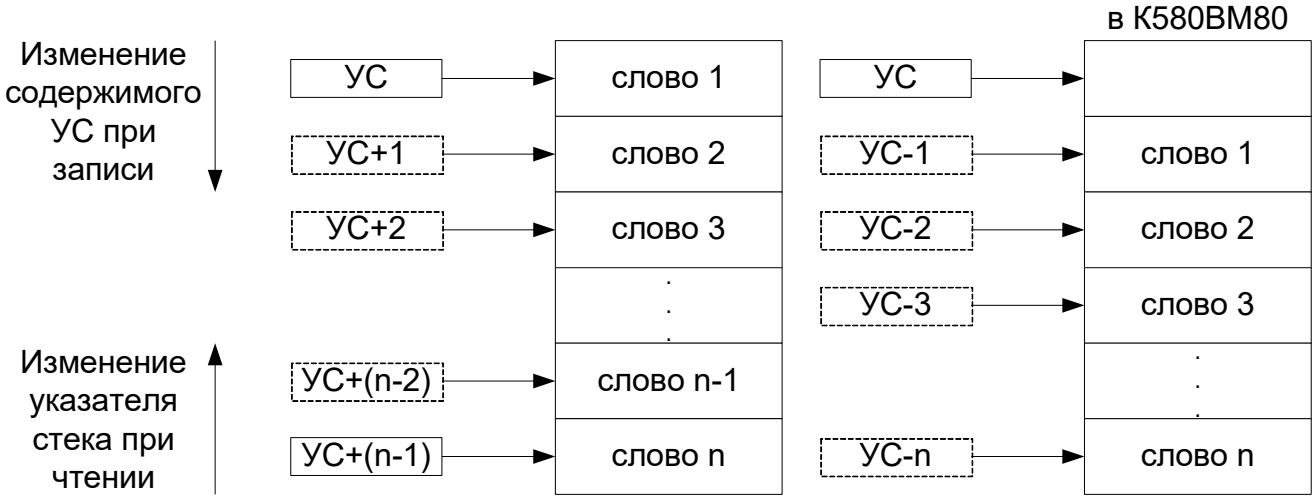


Рисунок 3.5.4- Изменение состояния УС стековой памяти микропроцессорной системы на основе МП Intel 8080

Если число слов, записанных в стек и считанных из стека равно, то стек приходит в исходное состояние.

3.6 АССОЦИАТИВНЫЕ ЗУ

Выборка информации в ассоциативных ЗУ (АЗУ) осуществляется по содержанию. Пусть, например, в АЗУ хранятся данные на студентов: год рождения, место работы, стаж работы, № группы и т.д., при этом необходимо выбрать фамилии всех студентов 1980 года рождения и работающих. Эта информация является ключом для поиска. Ассоциативная память позволяет выбрать эту информацию за один или несколько циклов памяти. Обычные (программные) средства поиска и сортировки будут работать очень долго.

Каждая ячейка такого АЗУ должна содержать (см. рисунок 3.6.1) регистр для хранения слова данных (как в обычных ОЗУ) и специальные комбинационные логические схемы для сравнения текущего содержимого регистра с ключевым словом, поступающим одновременно на вход всех ячеек. При поиске формируется сигнал чтения из всех ячеек

АЗУ. Импульс опроса появится на выходе ячейки, в которой содержимое совпадает с ключом.

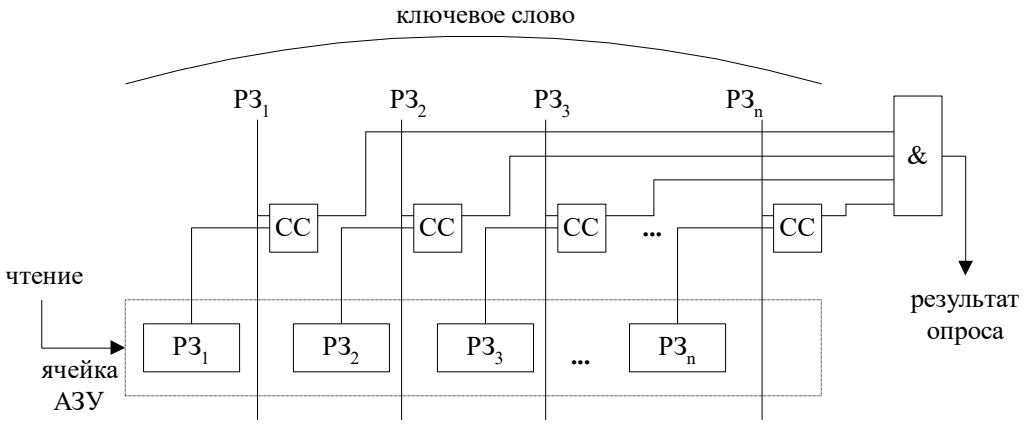


Рисунок 3.6.1- Устройство ячейки АЗУ

Структура АЗУ имеет следующий вид:

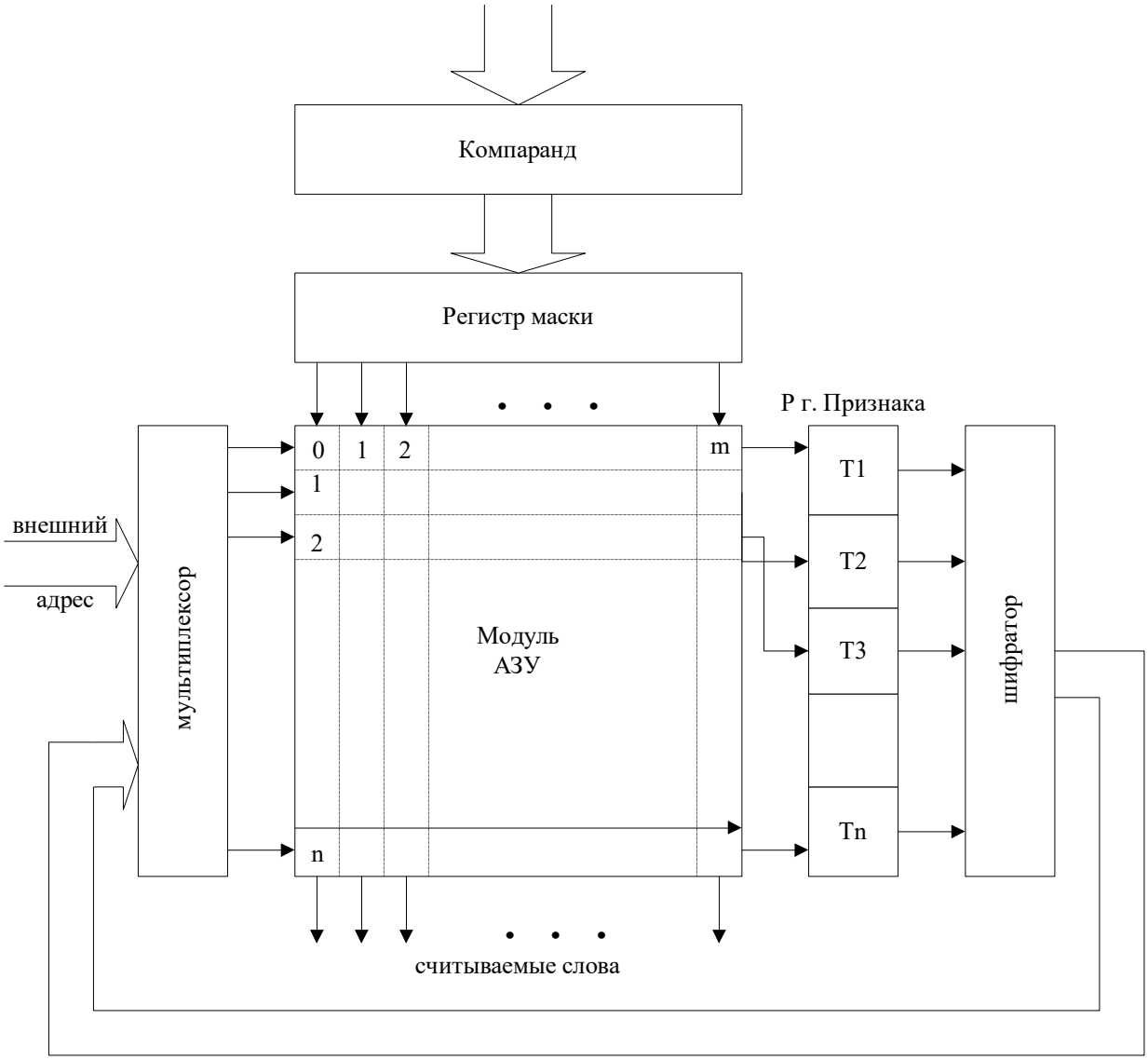


Рисунок 3.6.2- Структура АЗУ

Перед началом работы информация заносится в регистр, называемый компарандом (операнд в операции сравнения). Каждая ячейка связана с процессором через регистр признаков (Рг.Пр) с помощью разряда Tj. Регистр признака называют памятью отклика. Регистр маски маскирует те разряды компаранда, которые не должны участвовать в операции сравнения.

Перед началом работы все разряды Рг.Пр устанавливаются в состояние “Лог. 0”. По команде процессора “Сравнить” любая ячейка, содержащая слово, которое совпадает с компарандом, формирует сигнал, устанавливающий соответствующий разряд в Рг.Пр в состояние “Лог. 1”. Эта информация является адресной для линейной выборки.

4. ПРИНЦИПЫ-ОРГАНИЗАЦИИ ПРОЦЕССОРОВ

4.1 Обобщенные структуры процессоров с непосредственными и магистральными связями

Основными функциями процессора являются:

- организация обращений в ОП;
- дешифрация и выполнение команд;
- инициация работы периферийных устройств;
- обработка запросов прерываний, поступающих из устройств машины.

Обобщенная структура процессора с непосредственными (локальными) связями между его блоками приведена на рисунке 4.1.1.

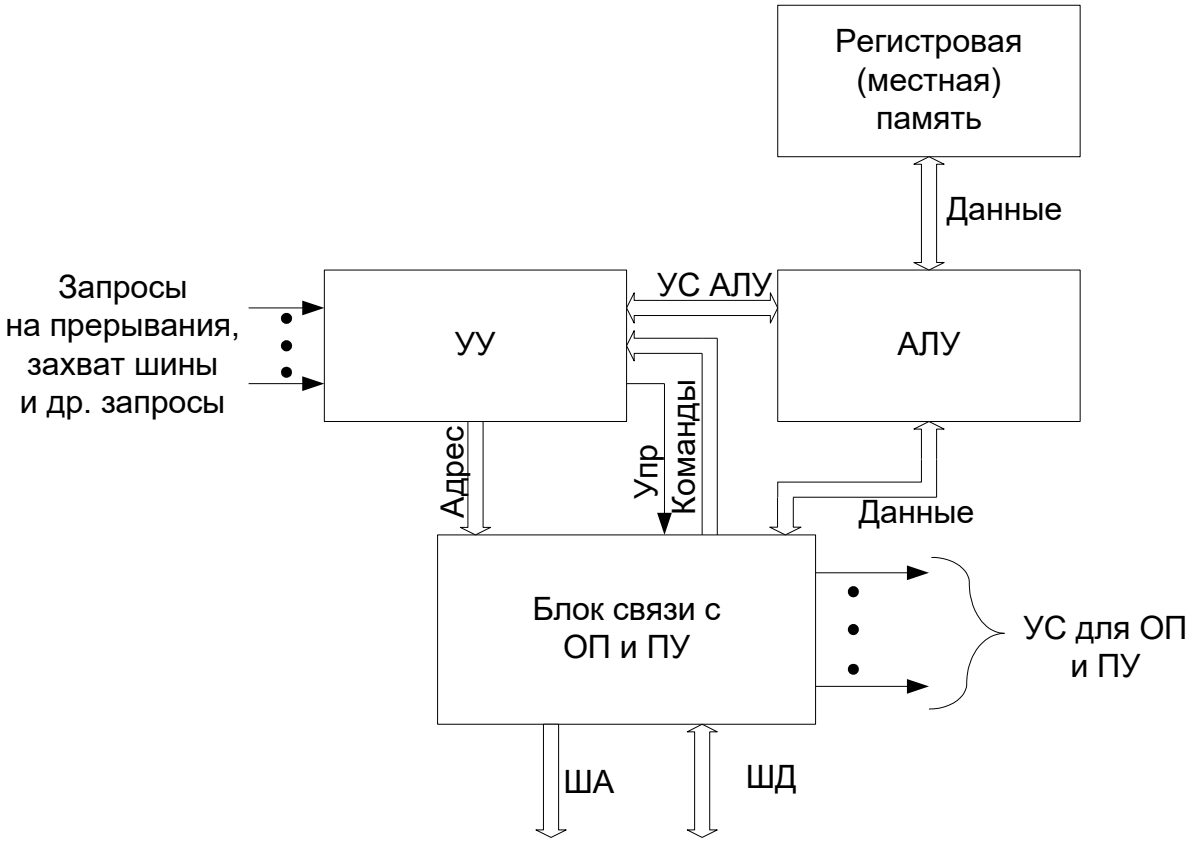


Рисунок 4.1.1 – Структура процессора с непосредственными связями

Структура процессора с магистральными связями между его блоками приведена на рисунке 4.1.2. Используется в большинстве современных МП.

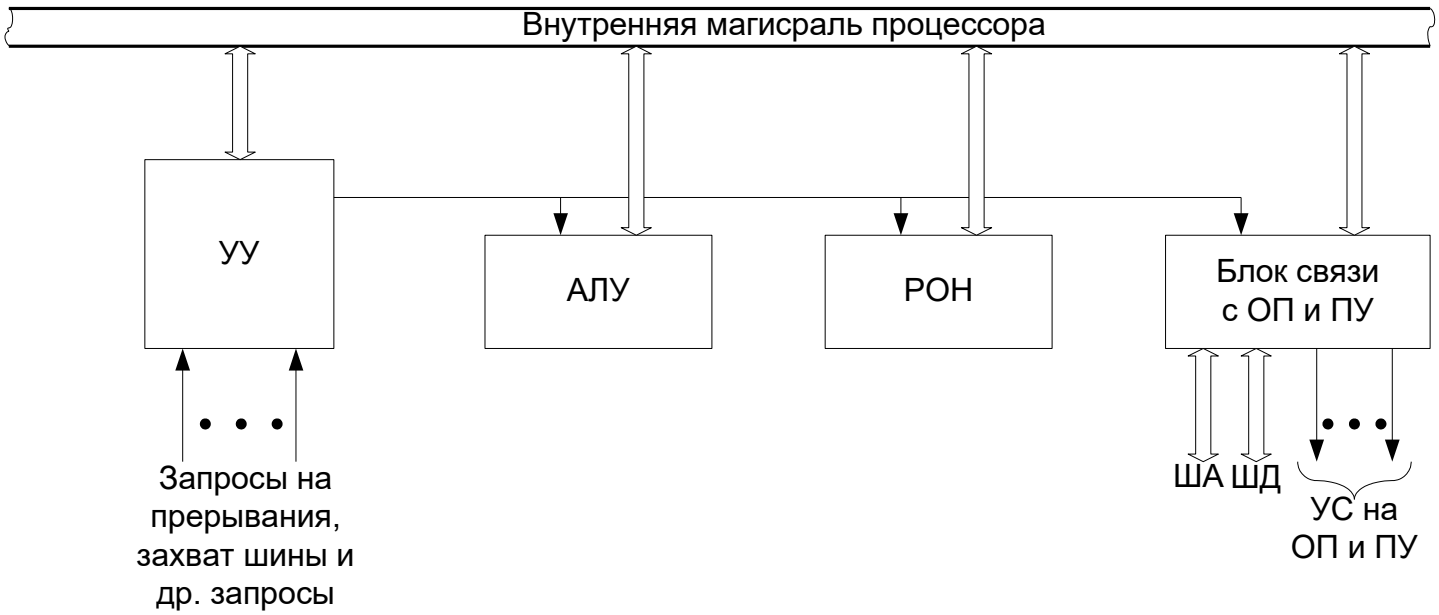


Рисунок 4.1.2 – Структура процессора с магистральными связями

В обеих структурах в УУ входят регистр команд, счетчик команд, схема управления прерываниями, регистр состояния процессора, дешифратор команд, устройство выработки последовательности сигналов, управляющих ходом выполнения команд в АЛУ и другие устройства.

АЛУ предназначается для выполнения арифметических и логических операций, т.е. в АЛУ происходит преобразование над логическими кодами фиксированной и переменной длины (над битами, байтами, словами), арифметические операции над числами с фиксированной и плавающей запятой, обработка алфавитно-цифровых слов переменной длины, а также операции преобразования кодов из одной системы счисления в другую.

Блок РОН позволяет увеличить производительность процессора и расширить его функциональные возможности. Обычно местная память имеет небольшой объем (8-16 байт), но выполняется на быстрых регистрах (на элементной базе самого процессора). Для адресации регистров используются укороченные команды, что сокращает объем программы и, следовательно, время ее выполнения. Расширение функциональных возможностей осуществляется путем введения в РОНы базовых и индексных регистров, а также указателей стека, что позволяет увеличить возможности адресации.

Блок связи с ОП и ПУ организует обмен с ОП и ПУ и обеспечивает защиту участков ОП от несанкционированного доступа .

Часто в состав процессора вводят блок контроля и диагностики, который служит для обнаружения и отказов аппаратуры процессора.

4.2 Декомпозиция процессора на УА и ОУ

При разделении процессора на управляющий автомат УА(УУ) и операционное устройство ОУ(АЛУ), его можно представить в следующем виде.(см. рисунок 4.2)

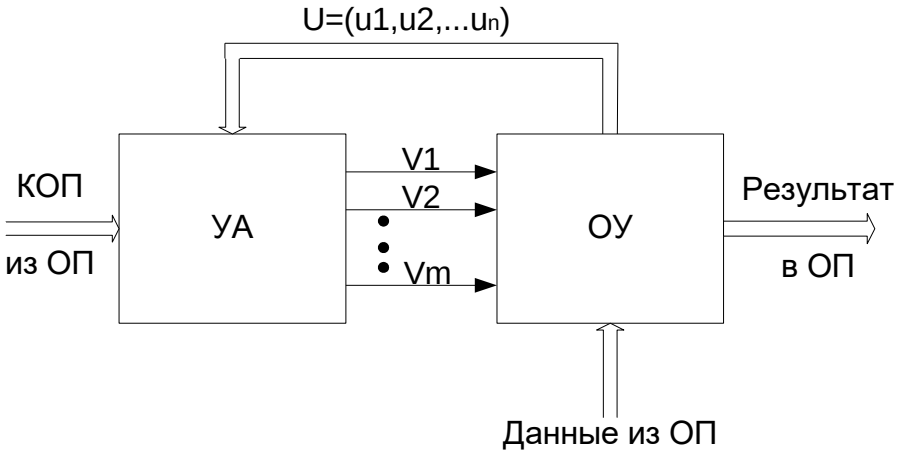


Рисунок 4.2 – Структура процессора при декомпозиции его на УА и ОУ

На приведенном рисунке $Y=\{y_1,y_2 \ldots y_n\}$-множество функциональных сигналов, управляющих ходом выполнения операций. Каждый функциональный сигнал соответствует выполняемой микрооперации на некотором такте работы ЭВМ. $U=\{u_1, u_2 \ldots u_n\}$- сигналы, оповещающие о ходе выполнения операции.

Управляющее устройство представляет собой УА, который может быть задан как автомат Мура или Мили.

4.3 Классификация УУ

Существуют два основных типа УУ:

- с жесткой или схемной логикой (аппаратные);
- с хранимой в памяти логикой (микропрограммного управления).

В аппаратных УУ для каждой операции, задаваемой КОП, строится набор схем, которые в нужных тактах формируют соответствующие управляющие сигналы.

В УУ с микропрограммным управлением каждой операции соответствует набор микрокоманд, хранимых в памяти микрокоманд. Каждая микрокоманда несет информацию о микрооперациях, подлежащих выполнению в течение машинного такта и указания, какая микрокоманда должна быть выбрана из памяти следующей. Последовательность микрокоманд, выполняющая одну машинную команду или некоторую процедуру, образует микропрограмму.

4.4 Микропрограммные УУ

4.4.1 Принцип микропрограммного управления Уилкса

Идея микропрограммного управления была высказана Уилксом в 1951 г. Она заключается в том, что управляющие сигналы “прошиваются” в памяти (ПЗУ, ППЗУ). Схема блока микропрограммного управления БМУ Уилкса следующая (смотри рисунок):

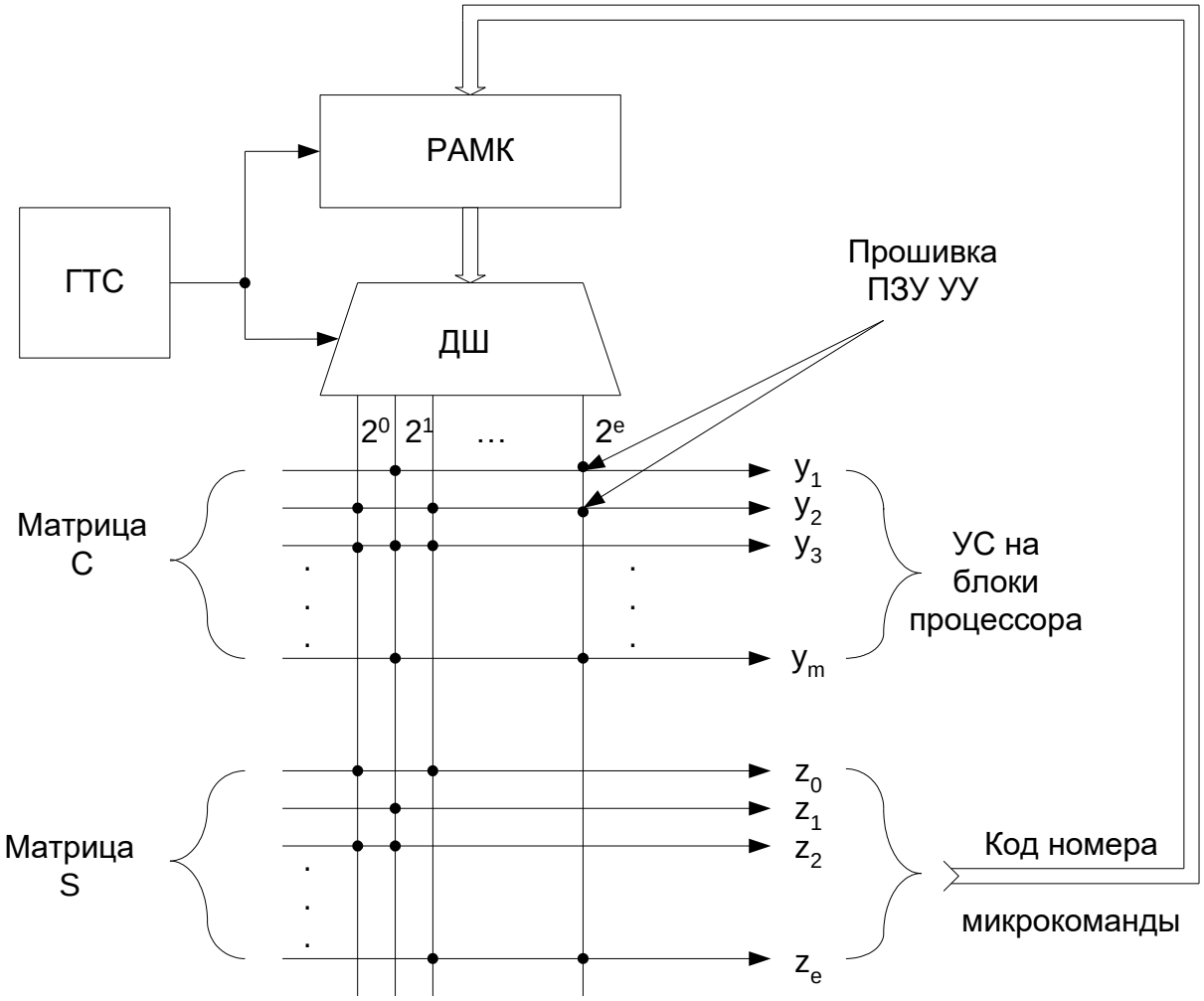


Рисунок 4.4.1 – Упрощенная структура блока микропрограммного управления

Схема содержит две матрицы С и S, Матрица С – управляющая, вырабатывает управляющие сигналы. Матрица S определяет последовательность выборки микрокоманд. Точками указаны места прошивки ПЗУ. Функционирует схема следующим образом.

В момент действия синхросигнала, выдаваемого генератором тактовых импульсов ГТИ, дешифратор (Дш) выбирает одну из l вертикальных линий. Возбуждение передается на те горизонтальные линии, которые соединены (электрически) с данной вертикальной прошивкой. Выработанные управляющие сигналы по шине $Y=\{y_1, y_2 \ldots y_m\}$ поступают в АЛУ и регистры процессора, а по шине $R=\{r_0, r_1 \ldots r_l\}$ передается адрес следующей микрокоманды в регистр адреса микрокоманды РАМК.

Микропрограммный способ управления удобен при разработке или дополнении любого набора команд, что достигается заменой одной “прошивки” ПЗУ на другое.

4.4.2 Структура блока микропрограммного управления

Структура блока микропрограммного управления (БМУ) приведена на рисунке 4.4.2.

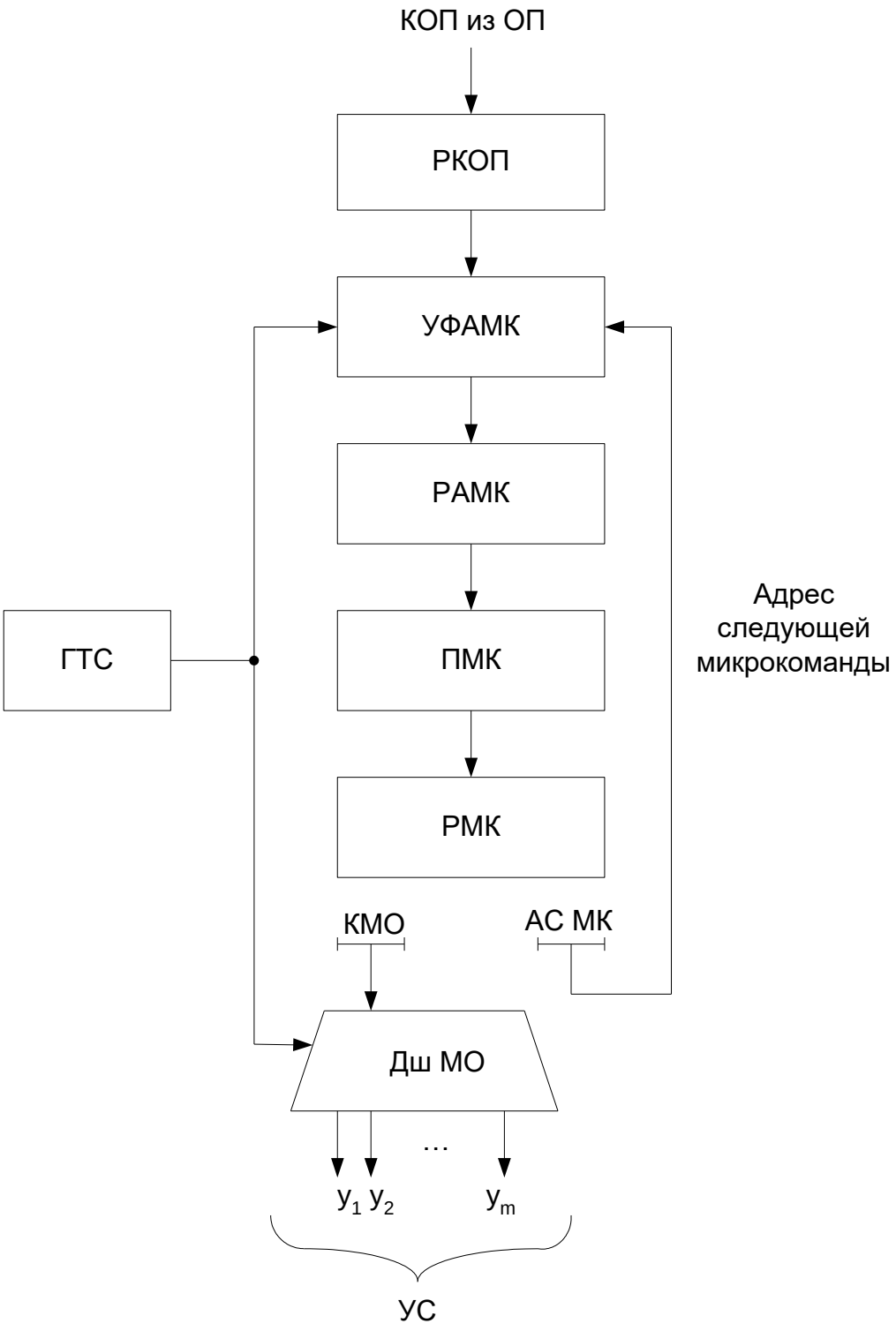


Рисунок 4.4.2 – Структура БМУ

В состав БМУ входят память микрокоманд (ПМК), регистр адреса микрокоманд (РАМК), регистр микрокоманд (РМК), дешифратор микроопераций (ДшМО), генератор тактовых сигналов (ГТС).

Код операции (КОП) поступает из ОП системы на регистр кода операции (РКОП), который задает начальный адрес микропрограммы. Адрес микропрограммы формируется устройством формирования адреса МК (УФАМК) и хранится в РАМК. По этому адресу из

памяти микрокоманд (ПМК) БМУ считывается микрокоманда и фиксируется в регистре МК (РМК).

Микрокоманда содержит два основных поля:

Код микрооперации (КМО).	Адрес следующей МК (АСМК)

КМО дешифрируется и преобразуется в набор управляющих сигналов $y_1 \ldots y_m$, управляющих функционированием процессора. Адрес следующей микрокоманды поступает в УФАМК, в ркезультате чего производится выборка следующей МК.

Рассмотренный БМУ использует принудительную адресацию МК. БМУ такого типа используются чаще всего. Кроме них используются и БМУ с естественной адресацией, в которых для задания адресов МК используется счетчик микрокоманд Сч МК.

4.5 Развернутая структура процессора и его функционирование

4.5.1 Обобщенная структура процессора с микропрограммным управлением

В состав процессора (см. рисунок 4.5.1) входят блоки АЛУ, РОН, УУ, интерфейсы памяти и УВВ. Данные из памяти и ПУ последовательно передаются через двунаправленную магистраль данных, БРД и внутреннюю магистраль данных на входы АЛУ или РОН. Команды поступают в РК по той же магистрали.

Выполнение некоторой программы начинается с загрузки СчК начальным адресом. Содержимое СчК передается в буферный регистр адреса (БРА) для адресации памяти. Команда, считанная из памяти по входной магистрали, через БРД поступает в РК. КОП команды используется для выборки микропрограммы из ПЗУ и формирования сигналов, управляющих ходом выполнения команды. Адресная часть команды передается в БРА для выборки операндов. Операнды заносятся в аккумулятор А или один из регистров РОН.

Результаты выполнения команд с выхода сумматора ($\sum$) поступают в магистраль данных, с которой они могут быть пересланы в ОП или любой регистр (А или РОН).

После завершения процесса исполнения текущей команды, содержимое СчК модифицируется и производится выборка следующей команды.

В качестве внешних управляющих сигналов используются выходные сигналы чтения (Чт) и записи (Зп) для управления памятью (формируются при выполнении команд обращения к памяти), сигналы ввода (Вв) и вывода (Выв) (формируются при выполнении команд обращения к УВВ); входной сигнал запрос прерывания ЗПр, обеспечивающий прерывание выполнения основной программы и переход к выполнению подпрограммы, соответствующей внешнему запросу. Часто в процессорах формируют сигналы внутренних прерываний (например, при попытке деления на нуль или при недопустимых переполнениях).

Указатель стека УС предназначен для адресации стековой памяти, которая чаще всего реализуется в некоторой области оперативной памяти. Эта область определяется либо операционной системой, либо программистом путем загрузки начального адреса области стека в УС.

4.5.2 Рабочий цикл процессора

Функционирование процессора состоит из повторяющихся рабочих циклов, каждый из которых соответствует выполнению одной команды. Завершив рабочий цикл процессор переходит к выполнению следующего рабочего цикла.

Предположим, что процессор может выполнять четыре типа команд:

- основные (арифметические, логические, пересылочные операции);
- передачи управления;
- ввода-вывода;
- системные (установка маски прерываний, состояния процессора и др.).

Рассмотрим рабочий цикл, выполняемый покомандно (существуют и рабочие циклы, выполняемые по машинным циклам).

Рабочий цикл начинается (см. рисунок 4.5.2) с определения состояния процессора - счет или ожидание . Из состояния ожидания процессор может выйти только по сигналу ЗПр (или “Запрос захвата шины” для некоторых процессоров). Процессор в этом состоянии никаких действий не выполняет.

В состоянии “Счет” (счет- последовательная выборка и выполнение команд), если поступил запрос прерывания, процессор сбрасывает триггер прерывания ТгПр и переходит к выполнения подпрограммы обработки прерывания путем передачи адреса подпрограммы в СчК. Если сигнал ЗПр отсутствует, последовательно выполняются следующие этапы рабочего цикла выполнения основной команды: формирование исполнительных адресов операндов, выборка операндов, выполнение операций и запоминание результата. После этого процессор переходит к выборке следующей команды и цикл повторяется.

При выполнении большинства команд формируются признаки операций, которые используются в командах условного перехода.

При выполнении команд передачи управления проверяется условие перехода по вышеуказанным признакам для команд условных переходов. Если условие не выполняется, то выбирается следующая по порядку команда по подвинутому адресу, хранящемуся в СчК. Если условие выполняется, то в СчК заносится адрес перехода.

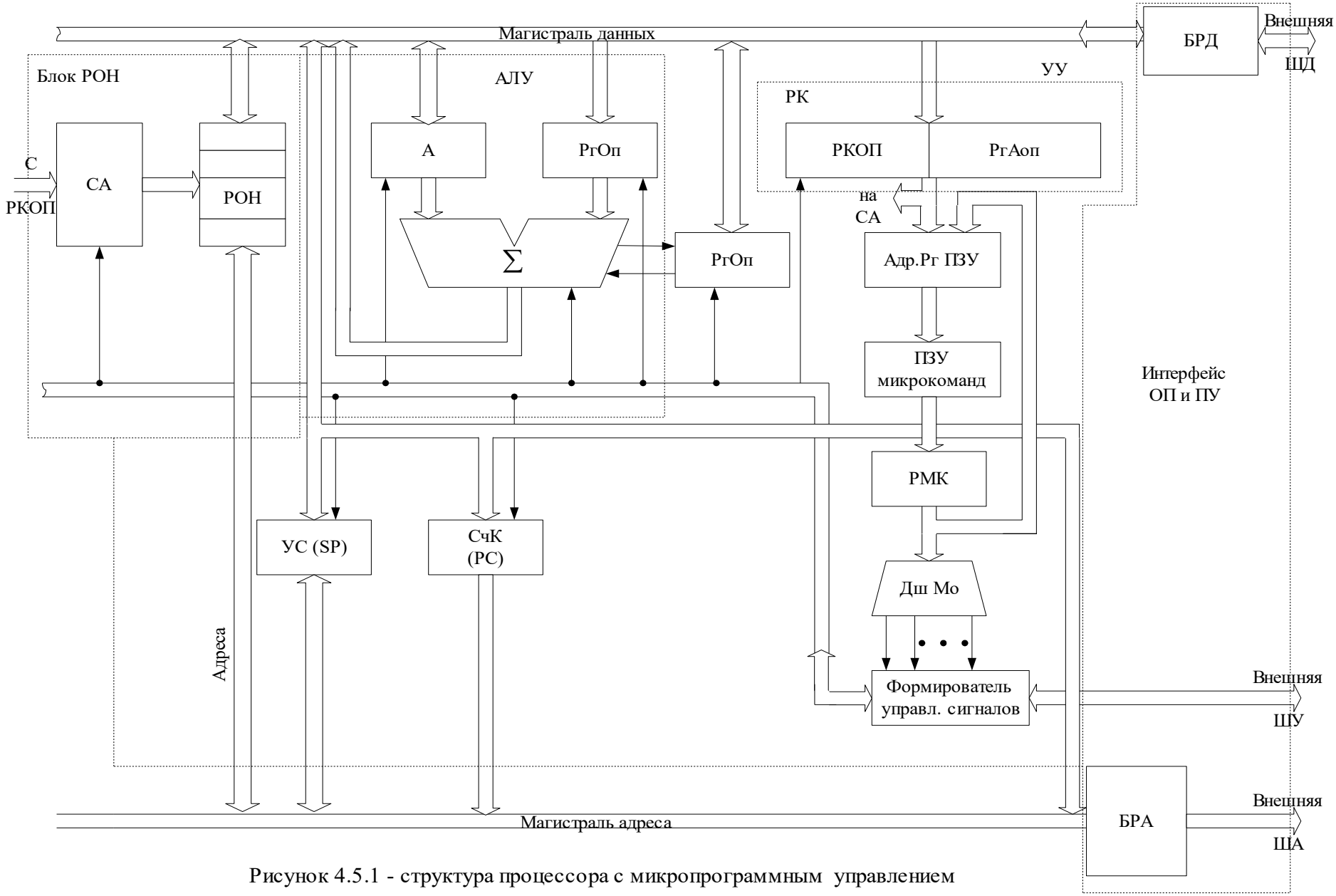


Рисунок 4.5.1 - структура процессора с микропрограммным управлением

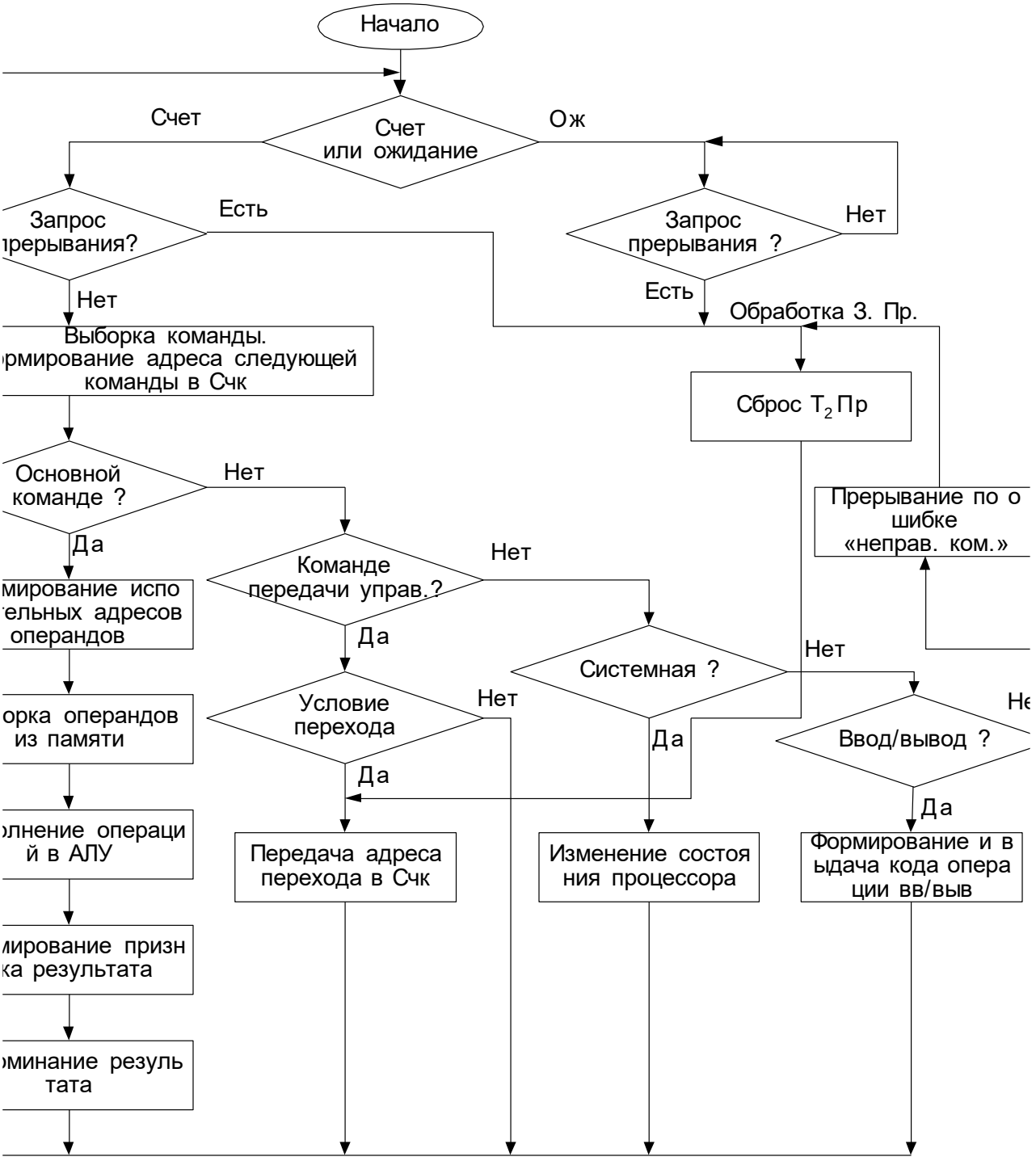


Рисунок 4.5.2 – Рабочий цикл процессора

Команда вызова подпрограмм и переход к подпрограмме выполняется так же, как и команда перехода, но при этом дополнительно запоминается состояние процессора.

Системные команды производят переключение состояния процессора (программы).

Команды ввода-вывода инициируют обращение процессора к УВВ (ПУ).

4.5.3 Понятие о слове состояния процессора

В ходе функционирования процессора постоянно меняется состояние его внутренних регистров. Если поступит запрос на прерывание выполнения основной программы или команда перехода к подпрограмме, которые приводят к изменению порядка выполнения программы, то для корректного возврата из подпрограммы в основную программу необходимо запомнить состояние процессора.

Содержимое регистров, обеспечивающих восстановление состояния вычислительного процесса, составляет слово состояния программы или процессора ССП (PSW- Program status word).

Чаще всего в информацию о состоянии процессора включают содержимое счетчика команд, содержимое регистра признаков и аккумулятора.

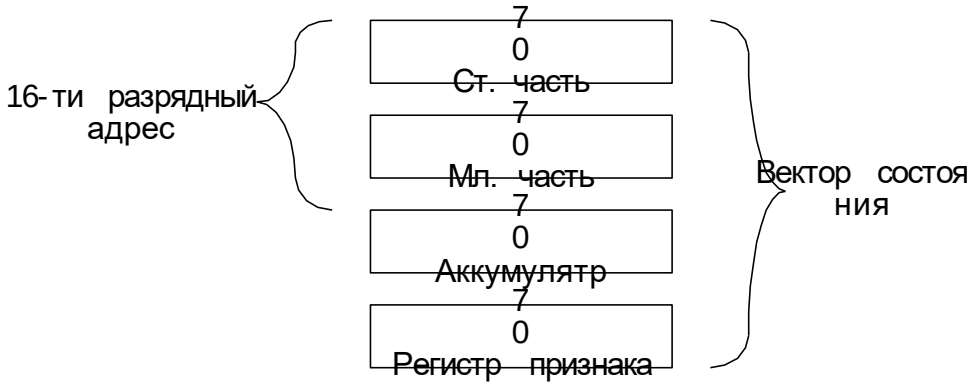


Рисунок 4.5.3 – Структура ССП

Слово состояния обычно сохраняют в специально отведенной области памяти ЭВМ или стековой памяти. Сохранение производится автоматически (т.е. аппаратно) в начале обслуживания запроса на прерывание программы. Другие регистры процессора могут быть сохранены и восстановлены программным путем.

4.5.4 Процедура выполнения команд перехода (условного и безусловного)

При естественной адресации адрес следующей команды получается из адреса выполняемой команды увеличением его на 1, 2, 3 и т.д. (в зависимости от количества байт в команде), т. к. команды располагаются в смежных ячейках памяти. Для этого содержимое СчК автоматически модифицируется после выполнения текущей команды.

Для управления ходом выполнения программ и организации ветвлений в систему команд процессоров с естественной адресацией были введены команды условных и безусловных переходов.

Рисунок 4.5.4 – Формат команд условного и безусловного переходов

В разных машинах реализация этих команд различная, однако общий подход следующий: содержимое поля адреса перехода команды загружается в СчК, после чего процессор продолжает выполнение программы с нового адреса.

Команды безусловного перехода предписывают совершать переход по программе независимо от каких-либо условий. Существуют команды безусловного перехода по косвенному адресу. В этом случае в коде команды указывают адрес ячейки, в которой хранится адрес перехода.

При условном переходе адрес следующей команды зависит от некоторого условия, полученного в результате выполнения предыдущей. Если условие выполняется, то процессор переходит к выполнению программы по адресу, указанному в адресной части команды, если нет, то к команде, следующей непосредственно за командой условного перехода.

4.5.5 Процедура выполнения команд вызова подпрограмм

Другим типом команд передачи управления являются команды вызова подпрограмм. Их особенность заключается в том, что по окончанию выполнения подпрограммы они должны обеспечить возврат к выполнению программы, из которой подпрограмма была вызвана. Для этого адрес возврата должен быть запомнен, для чего в СчК формируется продвинутый адрес, который затем сохраняется в памяти (или в стеке). Для перехода к выполнению подпрограммы в СчК заносится адресная часть команды ее вызова. По окончанию выполнения подпрограммы адрес следующей команды основной программы, ранее сохраненный в стеке, вызывается из него, заносится в СчК и выполнение программы продолжается. Для организации возврата в основную программу подпрограмма должна оканчиваться командой “Возврат” (“RETURN”). Кроме нее существует также и команда “Условного возврата”.

Формат команды “Перехода к подпрограмме” приведен на рисунке 4.5.5.1.

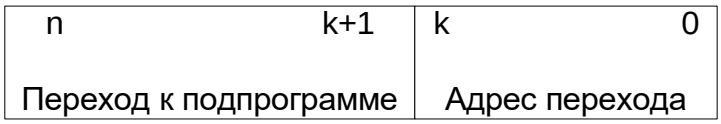


Рисунок 4.5.5.1 – Формат команды “Перехода к подпрограмме”

Процесс выполнения команд “Вызов подпрограмм” проиллюстрирован на рисунке 4.5.5.2. Короткий отрезок прямой на этом рисунке соответствует одной команде, длинный-переходу к выполнению подпрограммы или возврату из нее.

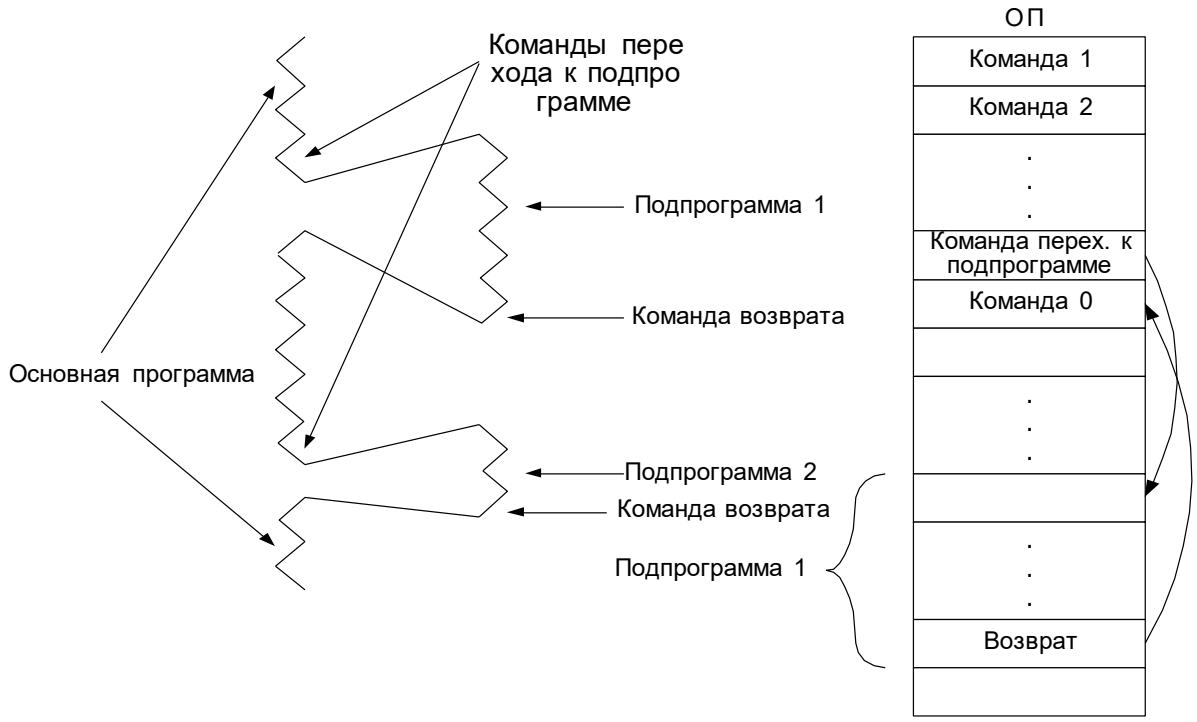


Рисунок 4.5.5.2 – Процесс выполнения команды “Вызов подпрограммы”

ПРИЛОЖЕНИЕ 2

2 ПРИЛОЖЕНИЕ 2

3 Практическая работа № 1

4 “ИССЛЕДОВАНИЕ АРХИТЕКТУРЫ ПК”

Задание

Составить отчет по практической работе, используя приведенный шаблон.

1. Определить состав ПК и составить ее техническую характеристику подобно приведенной ниже.
2. Краткую характеристику ПЭВМ составить, используя программы:
   a. из группы Стандартные ->.Служебные->Сведения о системе
   b. программу AIDA32.
3. Составить структурную схему ПК, указав основные характеристики ее составных частей. Используйте приведенную в задании схему для ее модификации.
4. Определить состав системного программного обеспечения ПК и составить схему их взаимодействия.

Отчет по работе

1. Состав ПК на основе прилагаемой документации

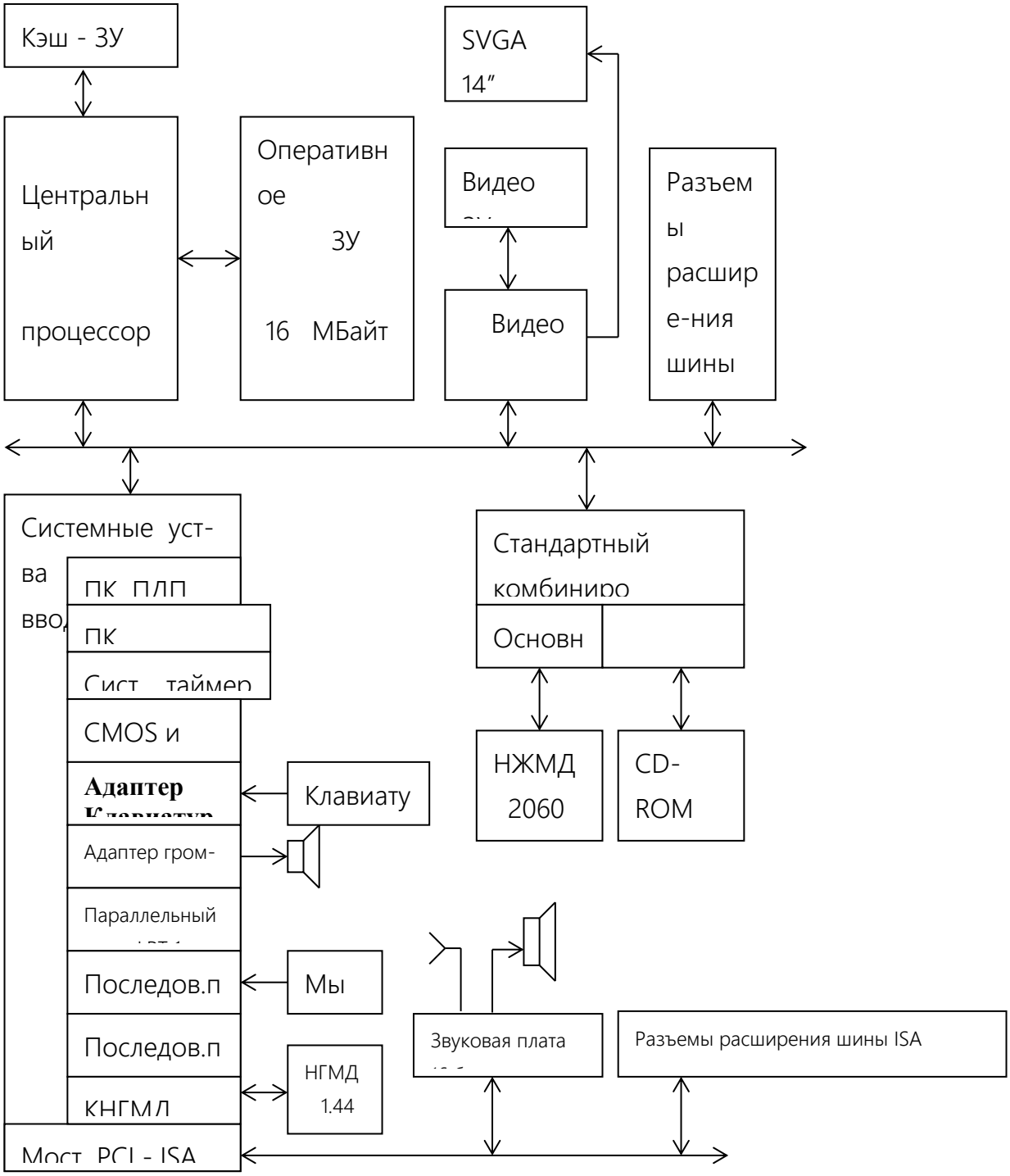
Центральный процессор:
Системная плата:
Оперативная память:
Разъемы:
Кэш - память:
Разъемы расширения:
Параллельные порты:
Последовательные порты:
Разъемы для подключения дисковых накопителей:
Дисковые накопители:
Видеоадаптер:
Монитор:
Звуковая плата:
Мышь:
Клавиатура:

2. Краткая характеристика ПК, полученная с использованием ее программных средств

Центральный процессор:
Системная плата:
Оперативная память:
Кэш - память:
BIOS :
Параллельные порты:
Последовательные порты:
Накопитель на жестком магнитном диске:
Устройство чтения компакт-дисков:

Накопитель на гибком магнитном диске:
Видеоадаптер:
Монитор:
Звуковая плата:
Мышь :
Клавиатура:

3. ***Структурная схема ПК***



4. Состав системного программного обеспечения ПК.

Используйте ниже приведенную схему программного обеспечения для определения состава системного программного обеспечения персонального компьютера.

Практическая работа №2

«Работа с машинными командами и командами ассемблера с помощью отладчика DEBUG»

Цель работы

Знакомство с отладчиком debug.exe. Получение практических навыков работы с данной программой.

Программное обеспечение:

Программа Debug.

Задание:

1. Изучить **теоретический материал,** проделав поэтапно описанные по ходу текста упражнения. Затем выполнить приведенные ниже задания.

2.Найдите сумму и разность 2-х чисел: 1-е число – количество букв в вашей фамилии+ количество букв в вашем имени (переведенное в шестнадцатиричную форму), 2-е – ваш день рождения. Сумму и разность переведите вручную в десятичную форму, результат и само решение напишите в отчете.

3.Просмотрите содержимое регистров микропроцессора, а также флагов и вставьте в отчет в виде скриншота. Объясните какую функцию выполняет каждый из регистров? Диапазон области памяти задайте используя для начала диапазона порядковый номер в журнале студенческой группы, для конца диапазона количество букв в вашей фамилии. Например студент Алексеев имеет порядковый номер в журнале 1, вводим D 0001:0008.

4.Запишите в регистр AX первое число (из задания 2), а в регистр BX - второе (из задания 2). Введите в оперативную память в сегмент кода (смещение 100) машинную команду сложения регистров AX и BX. Просмотрите на экране ее ассемблерную форму. Выполните эту команду, результат переведите в десятичную форму.

5.Введите в оперативную память в сегмент кода (смещение 100) набор команд ассемблера для распечатки символа на экране - первой буквы вашей фамилии. Проверьте программу в DEBUG. Затем запишите ее на диск в виде .COM-файла. Чему равен размер программы? Запустите ее на выполнение из DOS.

6.Все действия опишите в отчете.

Содержание отчета:

1.Название работы.

2.Цель работы.

3.Приборы и оборудование.

4.Краткие теоретические сведения.

5.Описание проделанных действий. (**привести скриншоты**).

6.Текст созданной программы.

7. Ответы на контрольные вопросы.

Контрольные вопросы

1. Что называется смещением сегмента?

2. Какой формат имеют машинные команды МП?

3. Как называется программа, позволяющая записывать коды чисел и команд в ячейки памяти?

4. Как осуществляется запуск программы Debug в среде ОС Windiws?

5. Содержимое каких регистров МП используется для формирования адреса ячейки сегментов кодов?

6. Как можно вывести содержимое всех регистров МП на экран монитора?

7. Укажите порядок изменения содержимого регистра ip.

8. Укажите порядок изменения содержимого регистра ax.

9. Что означает выражение: «выполнение программы в режиме трассировки»?

10. Какая команда программы выполняется при вводе команды и программы Debug.

11. Какая команда производит ввод в память данные или инструкции машинного кода?

12. Как сравнить содержимое двух областей памяти?

13. Каким образом заполнить область памяти данными из списка?

14. Какая команда производит выполнение отлаженной программы на машинном языке?

15. Какая команда записывает файл из Debug?

16. Как производится ассемблирование и дизассемблирование?

5 Краткие теоретические сведения

Программа DEBUG используется для тестирования и отладки исполняемых программ. Программа DEBUG показывает код и данные программы в шестнадцатеричном формате, и любые данные, которые вводятся в память, также должны быть в этом формате. DEBUG также реализует пошаговый режим исполнения, позволяющий выполнять инструкции программы по отдельности одну за другой и наблюдать результат выполнения каждой

инструкции в памяти и регистрах.

Команды DEBUG

Таблица 1. Краткая таблица всех команд debug.exe

Команда	Описание	Формат
A (Assemble)	Транслирование команд ассемблера в машинный код; адрес по умолчанию - CS:0100h.	A [<адрес_начала_кода>]
C (Compare)	Сравнение содержимого двух областей памяти; по умолчанию используется DS. В команде указывается либо длина участков, либо диапазон адресов.	C <начальный_адрес_1> L<длина> <начальный_адрес_2> C <начальный_адрес_1> <конечный_адрес_1> <начальный_адрес_2>
D (Display/Dump)	Вывод содержимого области памяти в шестнадцатеричном и ASCII-форматах. По умолчанию используется DS; можно указывать длину или диапазон.	D [<начальный_адрес> [L<длина>]] D [начальный_адрес конечный_адрес]
E (Enter)	Ввод в память данные или инструкции машинного кода; по умолчанию используется DS.	E [<адрес> [<инструкции/данные>]]
F (Fill)	Заполнение области памяти данными из списка; по умолчанию используется DS. Использовать можно как длину, так и диапазон.	F <начальный_адрес_1> L<длина> '<данные>' F <начальный_адрес> <конечный_адрес> '<данные>'
G (Go)	Выполнение отлаженной программы на машинном языке до указанной точки останова; по умолчанию используется CS. При этом убедитесь, что IP содержит корректный адрес.	G [=<начальный_адрес>] <адрес_останова> [<адрес_останова> ...]
H (Hexadecimal)	Вычисление суммы и разности двух шестнадцатеричных величин.	H <величина_1> <величина_2>
I (Input)	Считывание и вывод одного байта из порта.	I <адрес_порта>

L (Load)	Загрузка файла или данных из секторов диска в память; по умолчанию - CS:100h. Файл можно указать с помощью команды N или аргумента при запуске debug.exe.	L [<адрес_в_памяти_для_загрузки>] L [<адрес_в_памяти_для_загрузки> [<номер_диска> <начальный_сектор> <количество_секторов>]]
M (Move)	Копирование содержимого ячеек памяти; по умолчанию используется DS. Можно указывать как длину, так и диапазон.	M <начальный_адрес> L<длина> <адрес_назначения> M <начальный_адрес> <конечный_адрес> <адрес_назначения>
N (Name)	Указание имени файла для команд L и W.	N <имя_файла>
O (Output)	Отсылка байта в порт.	O <адрес_порта> <байт>
P (Proceed)	Выполнение инструкций CALL, LOOP, INT или повторяемой строковой инструкции с префиксами REPnn, переходя к следующей инструкции.	P [=<адрес_начала>] [<количество_инструкций>]
Q (Quit)	Завершение работы debug.exe.	Q
R (Register)	Вывод содержимого регистров и следующей инструкции.	R <имя_регистра>
S (Search)	Поиск в памяти символов из списка; по умолчанию используется DS. Можно указывать как длину, так и диапазон.	S <начальный_адрес> L<длина> '<данные>' S <начальный_адрес> <конечный_адрес> '<данные>'
T (Trace)	Пошаговое выполнение программы. Как и в команде P, по умолчанию используется пара CS:IP. Замечу, что для выполнения прерываний лучше пользоваться командой P.	T [=<адрес_начала>] [<количество_выполняемых_команд>]
U (Unassemble)	Дизассемблирование машинного кода; по умолчанию используется пара CS:IP. К сожалению, debug.exe некорректно дизассемблирует специфические команды процессоров 80286+,	U [<начальный_адрес>] U [<начальный_адрес конечный_адрес>]

	хотя они все равно выполняются корректно.	
W (Write)	Запись файла из debug.exe; необходимо обязательно задать имя файла командой N, если он не был загружен. А программы записываются только в виде файлов .COM!	W [<адрес> [<номер_диска> <начальный_сектор> <количество_секторов>]]

Основные особенности программы DEBUG.

- DEBUG не различает строчные и заглавные буквы.
- Все вводимые числа задаются в шестнадцатеричной форме.
- Пробелы используются в командах только для разделения параметров.
- Сегмент и смещение указываются с использованием символа двоеточия, т.е. в форме сегмент:смещение.

После загрузки отладчика на экране появится приглашение, выглядящее в виде дефиса. Регистры CS, DS, ES, SS в этот момент инициализированы адресом 256-байтного префикса сегмента программы, а рабочая область в памяти будет начинаться с адреса этого префикса + 100h. Команды debug.exe вводятся сразу после приглашения на месте, которое отмечено курсором. Каждая команда состоит из идентификатора и параметров, идентификатор состоит из одной буквы.

Команда D (Display - показать) в программе DEBUG

Эта команда выводит содержимое указанной области памяти на экран. Следующие три примера используют команду D для просмотра одного и того же участка памяти, начинающегося со смещения 3C1H в сегменте кода (CS):

D CS:3C1 (команда набрана заглавными буквами с пробелом);

DCS:3C1 (команда набрана заглавными буквами без пробела);

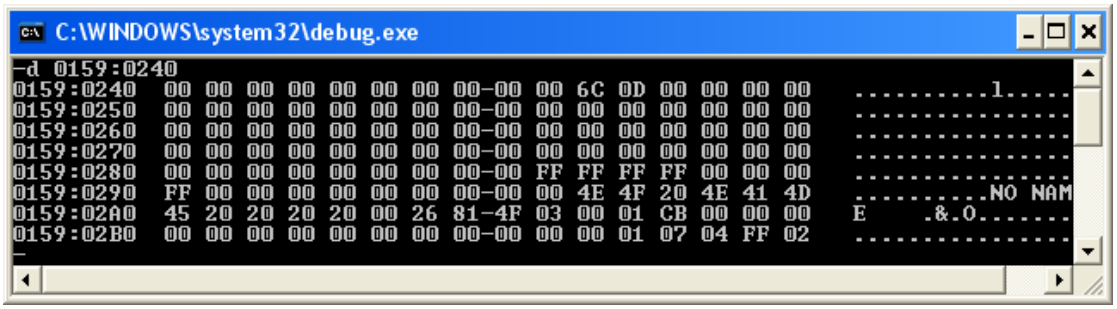
dcs:3cl (команда набрана строчными буквами без пробела);

В результате выполнения команды получается восемь строк. В левой части каждой из них будет указан адрес первого слева показанного байта в форме сегмент: смещение. Основную часть строки, в центре, занимает шестнадцатеричное представление параграфа, начинающегося с указанного в начале строки байта. Справа на экран выведены в ASCII-форме символы этого параграфа, которые можно вывести на экран, для облегчения интерпретации шестнадцатеричной записи.

Команда D выводит 8 строк данных, в каждой из которых по 16 байт (32 шестнадцатеричных разряда), всего – 128 байт, начиная с указанного адреса.

Адрес слева относится только к первому байту в строке, адреса последующих байтов могут быть легко найдены простым счетом: например, если адрес первого байта – 0159:0240H, то одиннадцатый байт в строке имеет адрес 0159:024AH. Шестнадцатеричное представление содержит два знака для каждого байта, байты разделяются пробелами для улучшения читаемости. Кроме того, восьмой и девятый байты разделяет дефис. Поэтому, например, если требуется найти байт со смещением xx13H, начните с байта xx10H и найдите третий после него байт.

Команда D также показывает содержимое регистров и состояние флагов в регистре Flags.



Упражнение 1: просмотр области данных BIOS

Первое упражнение показывает содержимое области данных BIOS в памяти, начиная с адреса 400H или, более точно, с адреса сегмента 40[0]H. BIOS инициализирует значения в этой области памяти при включении компьютера и меняет их в ходе выполнения программ.

Просматривайте эти значения при помощи адреса из двух частей: в качестве адреса сегмента (то есть 400, с отброшенным младшим разрядом), и nn в качестве смещения от начала сегмента. Воспринимайте адрес 40:nn как сегмент 40[0]H плюс смещение nnH.

Проверка параллельных и последовательных портов

Первые 16 байт области данных BIOS содержат адреса параллельных и последовательных портов. Введите следующую команду:

D 40:00 (и нажмите <Enter>)

Первые четыре выведенных слова указывают на адреса портов от COM1 до COM4. Если на компьютере два последовательных порта, первые два слова, вероятно, содержат F803 и F802 в обращенной (с переставленными байтами) последовательности. Последовательные порты имеют адреса 03F8 и 02F8. Следующие 4 слова указывают на параллельные порты от LPT1 до LPT4. Для системы с одним параллельным портом первое слово, вероятно, содержит 7803, т.е. адрес порта – 0378.

Проверка оборудования системы

Слово состояния оборудования в области данных BIOS предлагает базовую информацию о присутствующих в системе устройствах. Это слово, расположенное по адресу 410H - 411H, можно просмотреть командой

D 40:10 (и нажмите <Enter>)

Выведенные строки должны начинаться так:

0040:0010 xx xx ...

Допустим, что слово содержит 23 44 в шестнадцатеричной форме. Чтобы интерпретировать его, переставим байты (44 23) и преобразуем в двоичную форму:

Двоичное значение:	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	1
Позиция бита:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Эти биты означают:

Биты	Устройство
15, 14	Число параллельных (принтерных) портов = 1 (двоичное 01)
11 – 9	Число последовательных портов = 2 (двоичное 010)
7,6	Число дисководов = 1 (00 = 1,01 = 2, 10 = 3, 11 = 4)

5,4	Начальный видеорежим = 10 (01 = 40x25 цветной, 10 = 80x25 цветной, 11 = 80x25 монохромный)
1	1 = присутствует математический сопроцессор
0	1 = присутствует привод для дискет

Неописанные биты не используются.

Проверка состояния регистра клавиатуры (клавиша Shift)

В области данных BIOS по адресу 417H хранится первый байт состояния регистра клавиатуры. Убедитесь, что Num Lock и Caps Lock выключены, и просмотрите содержимое байта по этому адресу командой D 40:17.

Результат будет похож на

0040:0017 00 00 ...

Теперь включите Num Lock и Caps Lock и введите вновь команду D 40:17. Результат должен начинаться с 60 00.

Проверка состояния видеоустройства

В области данных BIOS по адресу 449H находится первая область видеоданных (Video Data Area). Введите команду D 40:49. Первый показанный байт указывает на текущий видеорежим (например, 03 - цветной), а второй – число столбцов на экране (например, 50 означает режим с 80 столбцами). Число строк хранится по адресу 40:84H.

Упражнение 2: Просмотр ROM BIOS

Сведения об авторском праве BIOS системы встроены в ROM BIOS no адресу FE00:0. В зависимости от производителя компьютера будут выведены различные строки, после которых будет указан семизначный серийный номер. Строка, указывающая на авторство BIOS, легко читается в виде ASCII-последовательности, а серийный номер – в виде шестнадцатеричного числа. Строка с указанием авторских прав может быть длиннее, чем показанный участок памяти; в этом случае для просмотра не показанной части снова введите D и нажмите <Enter>.

Проверка даты производства BIOS

Эта дата, записанная в виде мм/дд/гг, начинается с адреса FFFF5H. Для ее просмотра введите команду D FFFF:5. Знание этой даты полезно при определении возраста и модели компьютера.

Используя команду D, вы можете просмотреть содержимое любой области памяти. Можно также последовательно просматривать память, просто повторно вводя D – DEBUG будет выводить 128 байт, следующих за последними просмотренными.

Для завершения работы с DEBUG введите Q.

Упражнение 3: Использование непосредственных данных

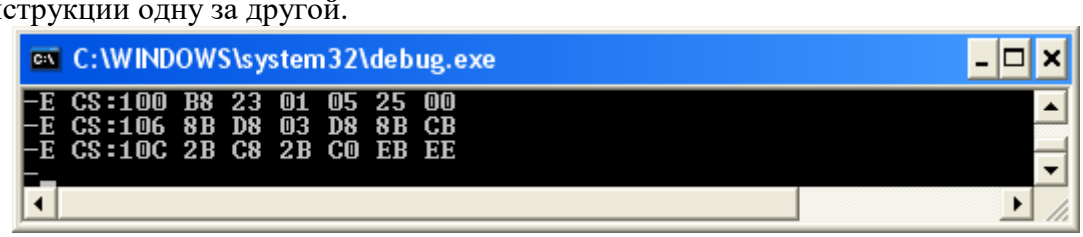
Воспользуемся DEBUG для ввода первой из двух программ непосредственно в память и наблюдения за ее выполнением. Обе программы включают простые инструкции машинного языка в виде, в котором они находятся в памяти, и позволяют продемонстрировать эффект их выполнения. Выполнение упражнения начинается с команды E (Enter, ввести).

Первая программа использует непосредственные данные - данные, определенные в теле инструкций. Далее показан как машинный код и соответствующий символьный код с комментариями для улучшения восприятия. Первой инструкции, B82301, соответствует символьный код MOV AX,0123, заносящий (копирующий) значение 0123H в регистр AX

(непосредственные данные записываются в прямой, с непереставленными байтами, форме). MOV – инструкция, AX – ее первый операнд, непосредственное значение 0123H – второй.

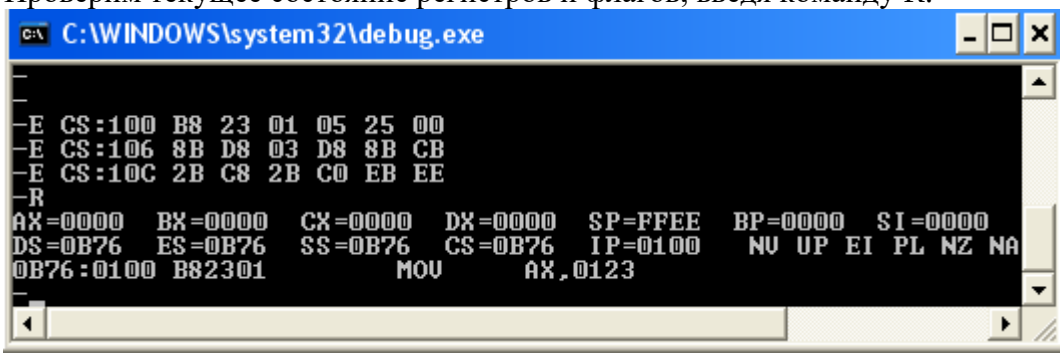
Машинная инструкция	Символьный код	Пояснение
B82301	MOV AX,0123	Переслать значение 0123h в AX
052500	ADD AX,0025	Прибавить значение O025h к AX
8BD8	MOV BX,AX	Переслать содержимое AX в BX
03D8	ADD BX,AX	Прибавить содержимое AX к BX
8BCB	MOV CX,BX	Переслать содержимое BX в CX
2BC8	SUB CX,AX	Вычесть содержимое AX из CX
2BC0	SUB AX,AX	Вычесть содержимое AX из AX (очистка AX)
90	NOP	Нет операции
EBEE	JMP 100	Переход к началу программы.

Из примера видно, что машинные инструкции имеют длину 1, 2 или 3 байта. Первый байт указывает, собственно, операцию, а все последующие – ее операнды (непосредственные значения, ссылки на регистры или адреса в памяти). Исполнение программы начинается с первой машинной инструкции и последовательно проходит через все инструкции одну за другой.



В примере машинный код разделен на три части по шесть байт и каждая вводится, используя команду E и начиная с адреса CS:100.

Проверим текущее состояние регистров и флагов, введя команду R.



Итак, как можно видеть, debug.exe инициализировал сегменты DS, ES, SS, CS одним и тем же адресом. Регистр IP содержит 0100, указывая на то, что инструкции выполняются со смещения 100h относительно CS (а мы, вводя инструкции в память, как раз указали этот адрес).

Здесь же указаны и значения флагов переполнения, направления, прерывания, знака, нуля, дополнительного переноса, четности и переноса:

Значение	**Описание**
NV	Отсутствие переполнения
UP	Направление вверх или вправо
EI	Разрешение прерываний
PL	Положительный знак
NZ	Ненулевое значение

NA	Отсутствие дополнительного переноса
PO	Нечетное слово
NC	Отсутствие переноса

Ввод инструкций программы

Для того чтобы ввести свою программу непосредственно в память, вводите только машинный код, символьный код и комментарии водить не нужно. Введите команду E и следующие символы, как показано ниже.

E CS:100 B8 23 01 05 25 00 (и нажмите <Enter>)

CS:100 указывает начальный адрес области, в которой будет храниться введенный код (обычный начальный адрес для машинного кода в программе DEBUG). Команда E сохранит следующие за ней байты в адресах от CS:100 до CS:105.

Следующая команда E сохранит еще 6 байт в адресах от CS:106 до CS:10B.

E CS:106 8B D8 03 D8 8B CB (нажмите <Enter>)

Последняя команда E сохраните 6 байт в адресах от CS:10C до CS:111.

E CS:10C 2B C8 2B C0 EB EE (нажмите <Enter>)

Если вы введете неправильную строку, просто повторите ввод правильно.

Выполнение инструкций программы

Теперь можно выполнить введенные инструкции по очереди. Можно также просматривать содержимое регистров после выполнения каждой инструкции с помощью команд R (registers - регистры) и T (trace - отследить).

Для просмотра начального состояния регистров и флагов введите команду R и нажмите <Enter>. DEBUG показывает содержимое регистров в шестнадцатеричной форме в виде

AX=0000 BX=0000 ...

DEBUG проинициализировал DS, CS, SS и ES одним и тем же адресом, xxxx[0]. IP должен показывать 0100, указывая, что выполнение инструкций начинается со смещения 100H относительно начала сегмента кода.

Регистр флагов показывает следующие значения флагов переполнения, направления, прерывания, знака, нуля, дополнительного переноса, четности и переноса:

NV UP El PL NZ NA PO NC

Эти значения соответствуют слева направо: отсутствию переполнения, направлению вверх (или вправо), разрешению прерываний, положительному знаку, ненулевому значению, отсутствию дополнительного переноса, нечетному слову, и отсутствию переноса. В данный момент эти значения не важны.

Сразу после регистров командой R показана первая, подлежащая выполнению инструкция:

xxxx:0100 B82301 MOV AX,0123

- xxxx означает начало сегмента кода по адресу xxxx[0]H. Значение xxxx:0100 означает смещение 100H от начала сегмента кода, начинающегося по адресу xxxx[0]H.
- B82301 – машинный код, введенный по адресу CS:100.
- MOV AX,0123 – символьная инструкция на языке Ассемблера, определенная DEBUG из машинного кода. Эта инструкция перемещает (копирует) непосредственное значение 0123H в регистр AX. DEBUG дизассемблировал машинный код для лучшего его понимания.

Для выполнения инструкции MOV введите команду T и нажмите <Enter>. Машинный код команды – B8 (поместить в AX), операнд – 2301. Операция помещает 23 в младшую половину регистра (AL) и 01 – в старшую (AH).

DEBUG выводит результаты выполнения – эффект, оказанный операцией на содержимое регистров. Регистр IP содержит теперь 0103H (0100H плюс 3 байта – длина выполненной инструкции). Это значение указывает на адрес, по которому расположена следующая подлежащая выполнению инструкция, то есть:

```
xxxx:0103 052500  ADD AX,0025
```

Для выполнения этой инструкции вновь введите T. Инструкция добавит 25H к младшему байту (AL) регистра AX и 00H – к старшему (AH), то есть добавит 0025H к содержимому AX. Теперь AX содержит 0148H, а IP – 0106H, указывая на следующую инструкцию, которую необходимо выполнить:

```
xxxx:0106 8BD8  MOV AX,BX
```

Вновь введите команду T. Инструкция MOV заносит содержимое регистра AX в регистр BX. После выполнения инструкции BX содержит 0148H. AX все еще хранит 0148H, поскольку инструкция MOV копирует данные, не удаляя их из источника.

Теперь последовательно вводите команды T для выполнения оставшихся инструкций. Инструкция ADD суммирует содержимое регистров AX и BX, записывая значение 0290H в регистр BX. Затем программа копирует содержимое BX в CX, вычитает AX из CX и вычитает AX из самого себя. Последняя инструкция очищает AX, устанавливая его в 0, и устанавливает флаг ZF из NZ (nonzero – не нуль) в ZR (zero – нуль), отображая результат операции.

Инструкция JMP устанавливает IP в 100H, и обработка возвращается обратно к началу программы. Эта инструкция введена из соображений предосторожности, поскольку за последней введенной инструкцией следует "мусор", который при попытке выполнения мог бы вызвать останов процессора или другие нежелательные результаты.

Регистры DS, ES, SS и CS содержат один и тот же адрес сегмента. Это потому, что DEBUG рассматривает весь введенный код как программу .COM, хранящую данные, стек и код в одном сегменте, хотя вы и храните их по отдельности внутри сегмента. При написании программы .EXE стек, данные и код хранятся в отдельных сегментах с разными адресами.

Для повторного выполнения программы введите еще раз команду T, DEBUG выполнит инструкцию JMP и перейдет к началу введенной программы.

Просмотр содержимого памяти

Для просмотра программы на машинном языке в сегменте кода запросите вывод информации командой D CS:100. Результаты выполнения этой команды представляются строками с 16 байтами (32 шестнадцатеричными разрядами) в каждой строке. Справа – ASCII представление (если соответствующий символ можно вывести) каждого байта. В случае машинного кода ASCII представление бессмысленно и может игнорироваться.

Первая строка начинается со смещения 100H в сегменте кода и содержит байты с адресами от CS:100H до CS:10FH. Вторая строка выводит байты с адресами от CS:110H до CS:11FH. Хотя введенная программа заканчивается байтом с адресом CS:111H, DEBUG автоматически выводит восемь строк – от CS:100H до CS:170H. В этом примере все данные, следующие за CS:111H – "мусор".

Введите Q для завершения работы с DEBUG или начните выполнение следующего упражнения.

Использование ранее определенных данных

Предыдущий пример использовал непосредственные данные, определенные прямо в директивах MOV и ADD. В этом примере данные (числовые константы 0123H и 0025H) определены в виде отдельных элементов данных в программе. Инструкции программы должны работать с ячейками памяти, содержащими эти значения.

Этот пример показывает, как компьютер получает доступ к данным с помощью адреса в регистре DS и смещений. В примере, начиная с адреса DS:0200H, определены следующие элементы данных.

Смещение в сегменте DS	Шестнадцатеричный код содержимого
0200H	2301H
0202H	2500H
0204H	0000H
0206H	2A2A2AH

Шестнадцатеричная цифра занимает полбайта, поэтому, например, 23H хранится в байте с адресом 200H, а 01H – в следующем (со смещением 201H). Ниже приведены машинные инструкции, обрабатывающие эти данные, со значениями, введенными в обратной (с переставленными байтами) форме.

Инструкция	Объяснение
A10002	Поместить слово (2 байта), начинающееся с адреса DS:0200H, в регистр AX
03060202	Добавить содержимое слова (2 байта), начинающегося с адреса DS:0202H, к содержимому регистра AX
A30402	Поместить содержимое AX в слово, начинающееся с адреса DS:0204H
EBF4	Перейти к началу программы

Две инструкции по перемещению данных имеют разные машинные коды: A1 и A3. Действительный машинный код инструкции MOV зависит от регистров, на которые ссылается инструкция, размера элемента данных (байт или слово), направления передачи данных (в регистр или из регистра) и того, ссылается инструкция на непосредственные данные, адрес в памяти или регистр.

Ввод инструкций и данных программы

Используйте программу DEBUG для ввода и выполнения программы этого примера. Сначала с помощью команды E введите инструкции, начиная с адреса CS:0100:

E CS:100 A1 00 02 03 06 02 02 (и нажмите <Enter>)

E CS:107 A3 04 02 EB F4 (и нажмите <Enter>)

Теперь (также при помощи команды E) введите соответствующие значения в ячейки сегмента данных:

E DS:0200 23 01 25 00 00 00 (и нажмите <Enter>)

E DS:0206 2A 2A 2A (и нажмите <Enter>)

Первая команда E сохраняет три слова (6 байт) в начале области данных – начиная со смещения 0200H. Все эти слова должны быть введены в форме с переставленными байтами, то есть 0123 как 2301 и 0025 как 2500. Когда в дальнейшем инструкция MOV запросит эти слова из памяти, байты будут переставлены опять, образуя исходные значения.

Вторая команда E вводит в память три символа звездочки (***) в виде кода 2A2A2A. Вы сможете их увидеть в виде символов с помощью команды D.

Для просмотра данных в сегменте (со смещениями от 0200H до 0208H) и инструкций в сегменте CS (от 0100H до 010AH) введите следующие команды D:

Для просмотра кода: 100,10A <Enter>

Для просмотра данных: 200,208 <Enter>

Выполнение инструкций программы

Введя инструкции, можно выполнять их тем же способом, что и в предыдущем примере. Сначала убедитесь, что IP содержит 0100H. После этого командой R просмотрите содержимое регистров, флагов и код первой инструкции. Хотя AX может все еще содержать значение, присвоенное ему в предыдущем примере, оно скоро будет заменено новым. Первая показанная инструкция

xxxx:0100 A10002 M0V AX,[0200]

CS:0100 указывает на первую инструкцию, A10002. DEBUG интерпретирует эту инструкцию как MOV и определяет, что она ссылается на ячейку со смещением [0200H] в сегменте DS. Квадратные скобки указывают, что это не непосредственное значение, а адрес в памяти. (Непосредственное значение для записи в AX выглядело бы как MOV AX,0200.)

Теперь введите команду T. Инструкция выполняется и помещает значение слова со смещения 0200H в регистр AX. Содержимое этого слова – 2301H, а инструкция переставляет байты и помещает его в регистр AX в виде 0123H, стирая любое предыдущее значение.

Введите вторую команду T. Будет выполнена следующая инструкция, т.е. ADD. Операция добавляет значение из слова по адресу DS:0202H к значению регистра AX. Результат – сумма 0123H и 0025H, то есть 0148H.

Следующая инструкция – MOV [0204],AX. Введите T для ее исполнения. Инструкция копирует значение из регистра AX в слово в памяти, занимающее ячейки с адресами 0204H и 0205H. При этом байты будут переставлены, и слово будет содержать значение 4801H. Для просмотра изменившегося содержимого ячеек памяти введите команду

D DS:200,208 <Enter>

Выведенные значения должны быть такими:

```
Значение в ячейке: 23  01  25  00  48  01  2A  2A  2A
Смещение:       200 201 202 203 204 205 206 207 208
```

Левая сторона дисплея показывает действительный машинный код так, как он хранится в памяти. Правая сторона помогает легко находить символьные данные. Эти шестнадцатеричные значения представлены в правой части экрана соответствующими им символами ASCII. Коды 23H и 25H выводятся как # (символ номера) и % (символ процента) соответственно. Три байта 2AH выводятся как три звездочки (*).

Можно завершить работу с DEBUG командой Q или перейти к выполнению следующего примера.

Повторное выполнение инструкций

1. Иногда необходимо вручную установить значение в регистре IP. Делается это так.
2. Введите команду R IP для вывода содержимого регистра IP.
3. Введите значение 100 (или адрес другой инструкции), а затем нажмите <Enter>.

Эта процедура возвращает к началу программы (или к инструкции внутри программы), и можно повторно выполнить уже пройденные шаги. Введите команду R (без IP). DEBUG выведет на экран содержимое регистров, флаги и следующую подлежащую выполнению инструкцию. Теперь можно использовать команду T для повторного выполнения инструкций. Если программа накапливает значения, используйте команду E для очистки ячеек памяти и команду R для очистки регистров.

Сохранение программы в DEBUG

Можно использовать DEBUG для сохранения программы на диске в двух случаях.

1. Для получения с диска существующей программы, внесения в нее изменений и последующего сохранения.
2. Для создания при помощи DEBUG маленькой программы в машинных кодах, которую нужно сохранить.

За детальным объяснением обращайтесь к описанию команды W (write -записать) На данном этапе может оказаться полезной команда H, складывающая и вычитающая шестнадцатеричные числа. Максимальная длина чисел – 4 шестнадцатеричных разряда. Например, введите команду H 3443 2A2B. Команда выведет сначала сумму (5E6E), а затем – разность (0A18).

DEBUG также можно использовать для ввода программ на языке Ассемблера.

Для сохранения программы на диске сначала задается имя файла:

```
-N my_file.com
```

Затем в регистр CX необходимо поместить размер программы в байтах. Он будет равен разности конечного и начального смещений. Теперь остается только осуществить запись на диск командой W и в результате увидеть записанное количество байтов. В итоге мы получаем программу, готовую к исполнению.

Выход осуществляется командой q. Пример:

```
-A
0B3B:0100 mov ax,1234
0B3B:0103 mov ah, 4c
0B3B:0105 int 21
0B3B:0107
-u CS:100, 106
0B3B:0100 B83412 MOV AX,1234
0B3B:0103 B44C MOV AH,4C
0B3B:0105 CD21 INT 21
-r cx
CX 0000
:7
-r
AX=0000 BX=0000 CX=0007 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=0B3B ES=0B3B SS=0B3B CS=0B3B IP=0100 NV UP EI PL NZ NA PO NC
0B3B:0100 B83412 MOV AX,1234
-N my.com
-W
Запись 00007 байт
-q
```

Для выполнения этой лабораторной работы понадобится несколько команд ассемблера:

- MOV AH,<шестнадцатиричное число> - запись в регистр AH числа 02 для указания системной функции - вывод символа на экран;

- MOV DL,<шестнадцатиричное число> - запись в регистр DL кода символа;

- INT 21 - основное прерывание DOS (процедура), реализующее много различных функций; номер функции записывается предварительно в регистр AH; для распечатки

символа на экране - в регистре AH функция 02, при этом в DL записывают предварительно код символа.

- INT 20 - прерывание DOS, осуществляющее выход из программы (из .COM-программы).

Пример:

Вывести символ "*" на экран.

```
mov AH,02 ; системная функция 02 - вывод символа на экран
mov DL,2A ; ASCII-код звездочки
int 21h ; прерывание для вывода "*"
int 20h ; выход из программы
```

ПРИЛОЖЕНИЕ III

Тест по предмету «Организация ЭВМ»

ФИО_____

1. **В цифровой обработке выделяют следующие основные этапы**:
   1.1. Преобразование цифрового сигнала в другой цифровой сигнал по заданному алгоритму, формирование результатирующего аналогового сигнала.
   1.2. Формирование цифрового сигнала, формирование результатирующего аналогового сигнала.
   1.3. Формирование цифрового сигнала, преобразование цифрового сигнала в другой цифровой сигнал по заданному алгоритму, формирование результатирующего аналогового сигнала.
2. **Классы сигналов:**
   2.1. Аналоговый, дискретный, цифровой
   2.2. Цифровой, дискретный.
   2.3. Аналоговый, цифровой.
3. **Цикл фон-Неймана включает следующую последовательность команд:**
   3.1. Выборка команды из памяти и загрузка ее в регистр команд, выполнение команд , дешифрование команды, увеличение содержимого счетчика команд на 1
   3.2. Выборка команды из памяти и загрузка ее в регистр команд, увеличение содержимого счетчика команд на 1, дешифрование команды, выполнение команд
   3.3. Выборка команды из памяти и загрузка ее в регистр команд, дешифрование команды, увеличение содержимого счетчика команд на 1, выполнение команд
4. **Какую операцию выполняет следующая команда:**
   END BEGIN_____
5. **Микросхемы постоянной флэш-памяти относятся** :
   5.1. Масочным
   5.2. Однократно программируемым
   5.3. Многократно программируемым
6. **Основой центрального процессора являются:**
   6.1. Блок управления, регистровая память
   6.2. Блок микрокоманд, кэш-память 1-го уровня.
   6.3. Блок управления, арифметико-логическое устройство
7. **Какую операцию выполняет следующая команда:**
   MOV AX,0134H_____
8. **Внешние запоминающие устройства предназначены, для:**
   8.1. Для записи, хранения и считывания информации.
   8.2. Для записи, считывания информации.
   8.3. Хранения и считывания информации.
9. **Взаимодействие между устройствами и оперативной памятью осуществляется посредством**
   9.1. Конроллера, шин
   9.2. Контроллера, буферной памяти
   9.3. Шин, драйверов
10. **К устройствам ввода относятся:**
    10.1. Монитор, мышь, джойстик, сканер
    10.2. Клавиатура, мышь, джойстик, сканер

10.3. Джойстик, динамики, клавиатура, мышь

11. Какую операцию выполняет следующая команда:
SUB CX,AX_____

12. Различают следуюшие шины:
12.1. Данных , управления, адреса
12.2. Адреса, данных
12.3. Адреса, управления

13. **Последовательность из 32 бит называют** (подчеркнуть): байт, слово, двойное слово
14. **Длина одного сегментного регистра** (подчеркнуть):
8 бит, 16 бит, 32 бита
15. **Напишите битовое представление числа** 62
_____
16. **Сложите следующие дв. числа:** 10011011,
00101101:_____
17. **Сложите следующие шест. числа**: 5CDD, AA7
_____
18. Какую операцию выполняет следующая команда:
ADD BX,AX_____
19. **Определите шестнадцатеричное представление следующих десятичных чисел:** 27, 157 _____
20. **Определите положительные значения для следующих отрицательных двоичных чисел:** 110010111, 10011101 _____
21. **Какой регистр адресует сегмент кода** (подчеркнуть): CS, DS, SS, ES
22. Регистр сегмента данных содержит 0BD5h и некоторая команда обращается к ячейке памяти внутри сегмента данных со смещение 0C3Dh. **Какой будет реальный адрес ячейки памяти? _____**
23. **Напишите машинные команды** для пересылки значения 02945h в регистр AX:
_____
24. **Какую операцию выполняет следующая команда**:
BEGIN ENDP_____
25. **Напишите машинные команды** для сложения 0D45h с содержимым регистра AX
_____
26. **Какие регистры можно использовать для сложения и вычитания** (подчеркнуть):
AX, BX, CX, DX, SI
27. **Какие регистры можно использовать для адресации выполняемой команды:** IP, CX, BP
28. **Процессор хранит двухбайтовые числовые данные** (слова) в памяти :
28.1. в обратной последовательности
28.2. в прямой последовательности
28.3. случайной последовательности

29. Какую операцию выполняет следующая команда:
SUB AX,AX_____

Количество правильных ответов:_____