Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

Н.В. Замятин

ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ

**Учебное методическое пособие
для студентов специальности 23.01.02
«Автоматизированные системы обработки
информации и управления»**

2005

Корректор: Осипова Е.А.

Замятин Н.В.
Организация ЭВМ и систем: Учебное методическое пособие. – Томск: Томский межвузовский центр дистанционного образования, 2005. – 215 с.

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ. МЕТОДИКА РАБОТЫ С ПОСОБИЕМ

Расположение материала представлено следующим образом. Всего имеется 8 разделов (включая литературу). 1, 2 и 3 контрольные работы приведены в разделе 3 «Контрольные работы с методическими указаниями». Необходимый материал для выполнения контрольных работ приведен в 6 разделе, состоящем из 11 глав. Содержание каждой главы соответствует главе учебно-методического пособия [1]. Для удобства выполнения контрольных работ теоретический материал каждой главы третьего раздела в данном учебно-методическом пособии представлен в виде рабочих тетрадей (структурированная форма).

Рекомендуется сначала изучить материал, подробно изложенный в учебном пособии [1], а затем при выполнении контрольных работ пользоваться материалом учебно-методического пособия, изложенным в структурированном виде в рабочих тетрадях, что позволяет ускорить обучение и повысить его качество.

Контрольные работы (3 раздел) приведены с примерами выполнения заданий. Представлены методические указания для выполнения контрольных работ и требования, указаны принципы формирования вариантов контрольных работ и пример их выполнения.

В 4 разделе приведены методические указания для выполнения контрольных работ с использованием пакета Electronics Workbench (EWB).

В 5 разделе приведены методические указания по выполнению контрольных работ. Представлены основные сведения для построения программ на ассемблере и примеры.

2 РАБОЧАЯ ПРОГРАММА ПО ДИСЦИПЛИНЕ «ОРГАНИЗАЦИЯ ЭВМ И СИСТЕМ»

1. ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ

Целью изучения дисциплины является получение знаний о технических средствах автоматизированных систем обработки информации и управления, структур и принципов функционирования вычислительных систем различного назначения, принципов организации вычислительного процесса.

В результате изучения дисциплины студент должен знать:

- современную аналоговую и цифровую элементную базу средств вычислительной техники, методы проектирования и расчета элементов и узлов электронных средств обработки информации;
- основные принципы организации и функционирования отдельных устройств и ЭВМ в целом, а также вычислительных систем и комплексов, характеристики и возможности в области применения наиболее распространенных классов и типов ЭВМ;
- принципы построения архитектуры вычислительных систем.

Уметь использовать:

- формальный аппарат для анализа технической структуры автоматизированных систем;
- методы анализа и синтеза электронных схем микропроцессорных устройств при создании АСОИУ;
- возможности вычислительных систем при построении АСОИУ.

Иметь опыт:

- выполнения схемотехнических расчетов электронных элементов и устройств ЭВМ, проектирования микропроцессорных контроллеров;
- комплексирования ЭВМ, систем, комплексов и сетей, анализа и оценки архитектур вычислительных систем.

Иметь представление:

- об основных закономерностях функционирования вычислительных комплексов и возможностях их системного анализа;

- о тенденциях развития микроэлектроники, о перспективных схемотехнических решениях в области цифровой и аналоговой техники;
- о современном состоянии и тенденциях развития архитектур ЭВМ, вычислительных систем, комплектов и сетей;
- об архитектуре и возможностях микропроцессорных средств.

Дисциплина связана с предшествующими ей курсами: «Операционные системы», «Системное программирование», «Электротехника и электроника» «Информационные технологии».

2. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

2.1 Лекции:

Введение

Технические средства автоматизированных систем обработки информации и управления. Классификация средств преобразования информации.Обобщенная структурная схема и принципы функционирования универсальных преобразователей информации. Состояние и перспективы их развития.

1. Общие сведения и принципы работы ЭВМ

Структура ЭВМ, поколения ЭВМ. Классификация ЭВМ, основные типы (микропроцессоры и микроЭВМ, персональные, суперЭВМ, нейрокомпьютеры). Основные устройства ЭВМ и их назначения. Принципы программного управления, представления данных, машинные операции, адресация и команды.

2. Арифметические основы ЭВМ

Системы счисления. Форма представления чисел в ЭВМ. (Числа с фиксированной точкой, числа с плавающей точкой). Прямой, обратный и дополнительный коды. Двоичная арифметика. Кодирование десятичных чисел и алфавитно-цифровой информации. Десятичная арифметика.

3. Основы логического проектирования цифровых устройств

Конечные автоматы комбинированного и последовательного типа. Синхронные и асинхронные элементы. Абстрактный и

структурный синтез конечных автоматов. Основные теоремы булевой алгебры для одной, двух и более переменных. Булева функция и способы ее представления.

Способы минимизации функций. Функционально полные системы логических элементов. Основные понятия логического синтеза комбинационных и последовательностных устройств. Автоматы Мили и Мура.

4. Элементы и функциональные узлы ЭВМ

Физические основы представления информации в ЭВМ. Системы цифровых элементов, основные характеристики, классификация. Логические элементы и элементы с памятью. Функциональные узлы без памяти и с памятью. Операционные устройства. Принцип построения операционных устройств В.М. Глушкова.

5. Принципы построения устройств памяти

Классификация и основные параметры ЗУ. Иерархическая структура ЗУ современных ЭВМ. Методы доступа. Способы организации памяти (адресная, магазинная, стековая, ассоциативная).

Оперативная память ЭВМ. Основные характеристики и классификация. Системы организации оперативной памяти. Многоуровневая оперативная память. Структурные схемы и принципы работы ЗУ, структуры 3D, 2 D и 2.5.D.

Полупроводниковая и интегральная оперативная память на биполярных и МОП транзисторах.

Постоянные ЗУ (ПЗУ). Полупроводниковые интегральные ПЗУ с программированием и перепрограммированием информации. ППЗУ и флеш-память.

Внешние запоминающие устройства. Накопители на магнитных лентах, накопители на магнитных дисках, накопители на ЦМД.

6. Принципы организации процессоров

Назначение и структура процессора. Процессорные устройства. Характеристика основных блоков процессора. Работа процессора. Арифметико-логические устройства (АЛУ). Алгоритм выполнения логических и арифметических операций с фиксиро-

ванной запятой в АЛУ в двоичной и десятичных системах счисления. Структура операционного устройства для выполнения операций над числами с фиксированной запятой. Особенности взаимодействия узлов и блоков. АЛУ при выполнении арифметических операций с плавающей запятой.

7. Устройства управления. Основные функции

Понятие операции, такта, микрооперации, микропрограммы. Операционные устройства процессоров для обработки командной информации. Организация естественного порядка выборки команд. Выполнение команд условного и безусловного переходов. Выборка операндов при непосредственной и косвенной адресации. Кодирование команд. Форматы команд.

Устройства управления процессом схемно-логического типа (Жесткая логика). Способы формирования сигналов управления микрооперациями.

Микропрограммный принцип построения УУ. Структура микропрограммного УУ. Организация чтения микропрограммы из памяти. Использование программируемых логических матриц для построения устройств управления. Средства мультипрограммной работы процессора. Принципы организации прерываний. Многоуровневые системы прерываний. Приоритеты. Слово состояния программы (ССП). Структурные схемы блока прерываний.

8. Принцип организации систем ввода-вывода. Интерфейсы

Организация ввода-вывода информации в ЭВМ. Система прерываний. Иерархия команд ввода-вывода. Состав команд и их форматы. Функции каналов ввода-вывода. Последовательные и параллельные порты ввода-вывода информации.

Интерфейс. Концепция интерфейсов. Интерфейс программного обмена данными. Интерфейс прямого доступа в память. Единый интерфейс общая шина. Организация обмена данными. Современные интерфейсы обмена данными. Алфавитно-печатающие устройства. Дисплеи.

9. Микропроцессоры и ПЭВМ

Место ПЭВМ в АСОИУ. Классификация и основные характеристики микроЭВМ и ПЭВМ. Перспективы развития. Особенности логической организации микроЭВМ. Базовая система машинных команд. Основные типы адресации. Форматы команд и данные.

Многопроцессорные наборы. Состав, классификация и основные характеристики. Архитектура микропроцессорного набора. Системная магистраль. Программируемые интерфейсы. Построения микропроцессорной системы на базе микропроцессорных наборов. Микропроцессоры с микропрограммным управлением.

Однокристальные микроЭВМ. Структурная схема, принцип работы и характеристики.

Программируемый интерфейс. Назначение, логическая организация. Персональные ЭВМ (ПЭВМ). Перспективные микропроцессоры. Новые тенденции в архитектуре и проектировании.

10. Организация вычислительного процесса в мультипроцессорных вычислительных системах

Параллельная обработка информации. Классификация систем параллельной обработки. Типы структурной организации многопроцессорных вычислительных систем. Способы обмена информацией между процессорами. Обмен через общее поле памяти. Обмен по магистралям доступа к индивидуальной памяти. Буферизация межпроцессорного обмена информацией. Математические модели межмашинного обмена информацией. Протоколы распределенных вычислительных систем, их уровни и назначения.

Организация вычислительного процесса в многопроцессорных мультисистемах. Реализация синхронного и асинхронного режимов функционирования вычислительных систем. Примеры построения мультипроцессорных вычислительных систем.

11. Нейрокомпьютерная техника

Основные понятия теории нейронных сетей. Классификация и парадигмы нейронных сетей. Методы обучения нейронных сетей. Аппаратная реализация нейронных сетей. Аналоговые и

цифровые представления. Оптические нейронные сети. Понятие нейрокомпьютер. Нейрокомпьютеры и их реализации. Коммерческие изделия и их приложения. Числовые, аналоговые, гибридные, оптоэлектронные и оптические нейрокомпьютеры.

2.2 Перечень возможных тем лабораторных работ для комплексного лабораторного практикума

Из приведенного примерного перечня тем лабораторные работы выполняются выборочно.

1. Исследование организации памяти ПЭВМ (4 часа).
2. Конфигурирование компьютера (4 часа).
3. Исследование и программирование последовательного порта (4 часа).
4. Исследование и программирование параллельного порта (4 часа).
5. Исследование и программирование таймера (4 часа).
6. Исследование и программирование прерываний (4 часа).
7. Исследование и программирование прямого доступа в память (4 часа).
8. Исследование и программирование персептрона (4 часа).
9. Исследование и программирование нейронной сети Хемминга (4 часа).
10. Исследование и программирование генетического алгоритма (4 часа).
11. Исследование и программирование нейронной сети с обучением обратным распространением ошибки (4 часа).

2.3 Перечень тем контрольных работ

1. Команды и адресация микропроцессоров.
2. Оценка производительности микропроцессорных систем.
3. Нейроматематика.

3 КОНТРОЛЬНЫЕ РАБОТЫ С ПРИМЕРАМИ ВЫПОЛНЕНИЯ ЗАДАНИЙ

Состоят из трех видов работ. Тема каждой контрольной работы соответствует определенным разделам учебного пособия, а в совокупности они охватывают весь материал по изучаемой дисциплине.

- Контрольная работа №1. Синтез логических узлов ЭВМ (логические основы ЭВМ).
- Контрольная работа №2. Программирование вычислительных систем (базовые основы функционирования компьютеров, процессоры, организация памяти и ввода-вывода, многопроцессорные и нейрокомпьютерные системы, программирование процессоров).
- Контрольная работа №3. Синтез операционного устройства для понимания организации микропроцессора.

3.1 Методические указания для выполнения контрольной работы №1

Контрольная работа посвящена выполнению задания по синтезу логических узлов, необходимых для построения устройств преобразования информации.

Цель задания: изучить основные функции логических узлов. Для этого предлагается синтезировать логические узлы для выполнения заданной логической операции (счет в прямом или обратном направлении, сдвиг кода вправо или влево).

Параметры задания:

- вид логического узла,
- вид операции,
- вид триггерной логики для реализации логического узла.

Для выбора варианта необходимо воспользоваться иерархической схемой, начиная движение снизу (от номера варианта) вверх, получая параметры варианта.

Задание: необходимо для определенного варианта синтезировать логический узел в виде регистра сдвига влево (вправо) или счетчика с прямым (обратным) счетом на D-триггерах (RS-

триггерах). Реализовать логический узел в среде EWB и доказать его работоспособность записыванием и считыванием трехразрядных кодов.

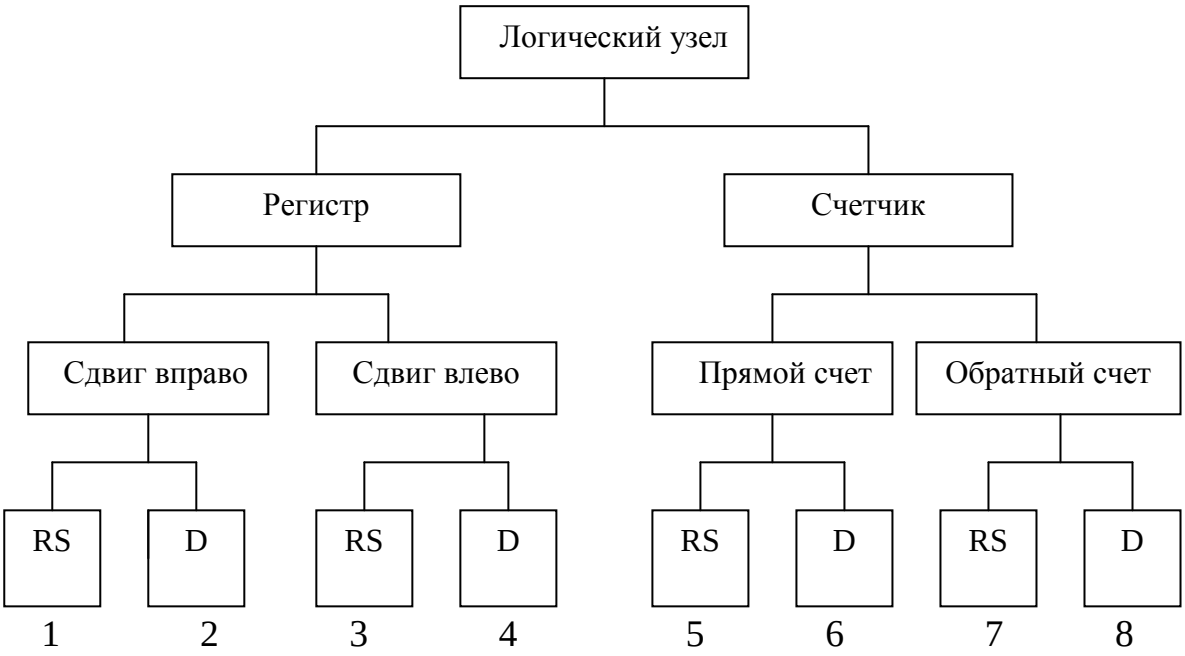
Последовательность действий при выполнении контрольной работы:

1. Уяснить задачу.
2. Выбрать номер варианта (рис. 3.1).
3. Составить граф переходов.
4. Составить таблицу истинности для указанного триггера.
5. Составить таблицу истинности для указанного логического узла.
6. Минимизировать полученные булевы функции.
7. Выбрать логический базис (И-НЕ либо ИЛИ-НЕ).
8. Минимизировать полученные булевы функции.
9. Привести к единому логическому базису.
10. Синтезировать схему логического узла.
11. Собрать логическую схему в среде EWB.
12. Проверить функционирование.
13. Подготовить отчет по контрольной работе.

Содержание отчета по контрольной работе:

1. Номер варианта.
2. Граф переходов.
3. Теорема истинности выбранного триггера.
4. Таблица истинности с отображением склеенных минтермов (макстермов).
5. Преобразованные по теореме Де Моргана булевы функции.
6. Скопированная и распечатанная из EWB схема логического узла.
7. Проверка функционирования узла в отображения содержания значений элементов и переходов триггеров в виде 0 и 1 для какой-либо строки таблицы истинности синтезируемого узла.
8. Выводы.

Схема формирования вариантов



Номера вариантов

Рис. 3.1 – Схемы формирования вариантов для контрольной работы №1

Пример выполнения контрольной работы №1

Задание:

- синтезировать логическую схему регистра;
- уяснить задачу;
- составить граф переходов;
- составить таблицу истинности;
- получить булевы функции;
- минимизировать булевы функции;
- взять заданный базис в виде заданных триггеров;
- составить логическую схему;
- произвести проверку;
- изобразить эпюры сигналов.

Наименование регистра	Реверсивный сдвиг
RS-триггер	5 вариант

Ответ:

Составляем граф переходов.

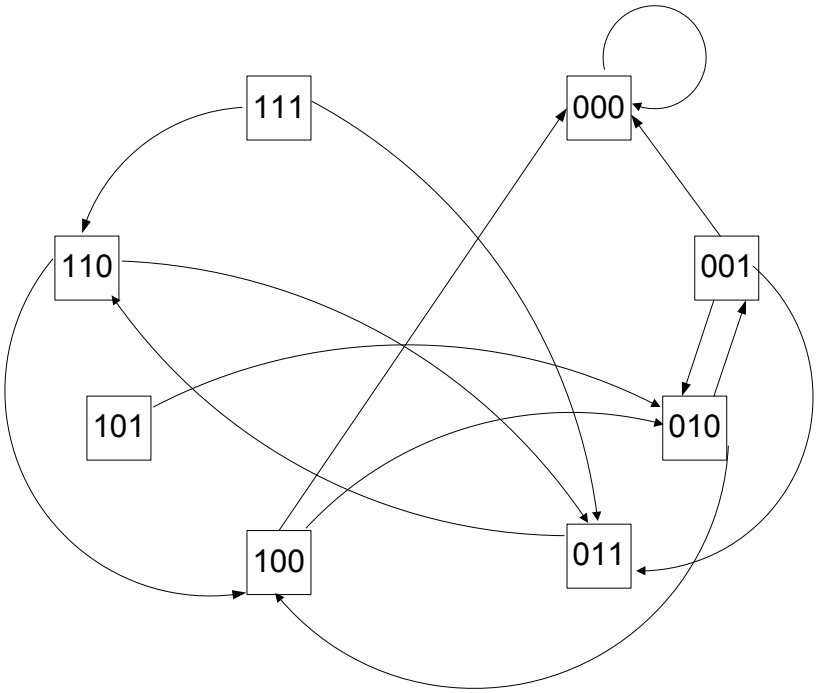


Рис. 3.2 – Граф переходов для функционирования регистра

Составляем таблицу истинности.

X	$Q_2(t)$	$Q_1(t)$	$Q_0(t)$	$Q_2(t+1)$	$Q_1(t+1)$	$Q_0(t+1)$	T_2		T_1		T_0	
							R_2	S_2	R_1	S_1	R_0	S_0
0	0	0	0	0	0	0	*	0	*	0	*	0
0	0	0	1	0	0	0	*	0	*	0	1	0
0	0	1	0	0	0	1	*	0	1	0	0	1
0	0	1	1	0	0	1	*	0	1	0	0	*
0	1	0	0	0	1	0	1	0	0	1	*	0
0	1	0	1	0	1	0	1	0	0	1	1	0
0	1	1	0	0	1	1	1	0	0	*	0	1
0	1	1	1	0	1	1	1	0	0	*	0	*
1	0	0	0	0	0	0	*	0	*	0	*	0
1	0	0	1	0	1	0	*	0	0	1	1	0
1	0	1	0	1	0	0	0	1	1	0	*	0
1	0	1	1	1	1	0	0	1	0	*	1	0
1	1	0	0	0	0	0	1	0	*	0	*	0
1	1	0	1	0	1	0	1	0	0	1	1	0
1	1	1	0	1	0	0	0	*	1	0	*	0
1	1	1	1	1	1	0	0	*	0	*	1	0

По таблице истинности получим булевы функции

$$R_2 = \overline{X}Q_2\overline{Q_1}\overline{Q_0} + \overline{X}Q_2\overline{Q_1}Q_0 + \overline{X}Q_2Q_1\overline{Q_0} + \overline{X}Q_2Q_1Q_0 + XQ_2\overline{Q_1}\overline{Q_0} + XQ_2\overline{Q_1}Q_0$$

$$S_2 = X\overline{Q_2}Q_1\overline{Q_0} + X\overline{Q_2}Q_1Q_0$$

$$R_1 = \overline{X}\overline{Q_2}Q_1\overline{Q_0} + \overline{X}\overline{Q_2}Q_1Q_0 + X\overline{Q_2}Q_1\overline{Q_0}$$

$$S_1 = \overline{X}Q_2\overline{Q_1}\overline{Q_0} + \overline{X}Q_2\overline{Q_1}Q_0 + X\overline{Q_2}\overline{Q_1}Q_0 + XQ_2\overline{Q_1}Q_0$$

$$R_0 = \overline{X}Q_2\overline{Q_1}Q_0 + XQ_2\overline{Q_1}Q_0 + XQ_2Q_1\overline{Q_0} + XQ_2\overline{Q_1}Q_0$$

$$S_0 = \overline{X}\overline{Q_2}Q_1Q_0 + \overline{X}Q_2Q_1\overline{Q_0}$$

С помощью карт Карно минимизируем булевы функции

R_2	$\overline{X}\overline{Q_2}$	$\overline{X}Q_2$	XQ_2	$X\overline{Q_2}$
$\overline{Q_1}\overline{Q_0}$	*	1	1	*
$\overline{Q_1}Q_0$	*	1	1	*
Q_1Q_0	*	1	0	0
$Q_1\overline{Q_0}$	*	1	0	0

$$R2 = \overline{X} + \overline{Q1} = \overline{X * Q1}$$

S_2	$\overline{X}\overline{Q_2}$	$\overline{X}Q_2$	XQ_2	$X\overline{Q_2}$
$\overline{Q_1}\overline{Q_0}$	0	0	0	0
$\overline{Q_1}Q_0$	0	0	0	0
Q_1Q_0	0	0	*	1
$Q_1\overline{Q_0}$	0	0	*	1

$$S2 = X * Q1 = \overline{\overline{X * Q1}}$$

R_1	$\overline{X}\overline{Q_2}$	$\overline{X}Q_2$	XQ_2	$X\overline{Q_2}$
$\overline{Q_1}\overline{Q_0}$	*	0	*	*
$\overline{Q_1}Q_0$	1	0	1	1
Q_1Q_0	*	0	0	0
$Q_1\overline{Q_0}$	1	0	0	0

$$R1 = \overline{X} * \overline{Q2} + X * \overline{Q0} = \overline{\overline{\overline{X} * \overline{Q2}} * \overline{X * \overline{Q0}}}$$

S_1	$\overline{X}\ \overline{Q_2}$	$\overline{X}Q_2$	XQ_2	$X\overline{Q_2}$
$\overline{Q_1}\overline{Q_0}$	0	1	0	0
$\overline{Q_1}Q_0$	0	*	0	0
Q_1Q_0	0	1	1	1
$Q_1\overline{Q_0}$	0	*	*	*

$$S1 = \overline{X} * Q2 + X * Q0 = \overline{\overline{\overline{X} * Q2} * \overline{X * Q0}}$$

R_0	$\overline{X}\ \overline{Q_2}$	$\overline{X}Q_2$	XQ_2	$X\overline{Q_2}$
$\overline{Q_1}\overline{Q_0}$	*	*	*	*
$\overline{Q_1}Q_0$	1	1	1	1
Q_1Q_0	0	0	1	1
$Q_1\overline{Q_0}$	0	0	1	1

$$R0 = \overline{Q1} + X = \overline{Q1 * \overline{X}}$$

S_0	$\overline{X}\ \overline{Q_2}$	$\overline{X}Q_2$	XQ_2	$X\overline{Q_2}$
$\overline{Q_1}\overline{Q_0}$	0	0	0	0
$\overline{Q_1}Q_0$	0	0	0	0
Q_1Q_0	1	1	0	0
$Q_1\overline{Q_0}$	*	*	0	0

$$S0 = \overline{X} * Q1 = \overline{\overline{X} * Q1}$$

Составим логическую схему (рис. 3.3):

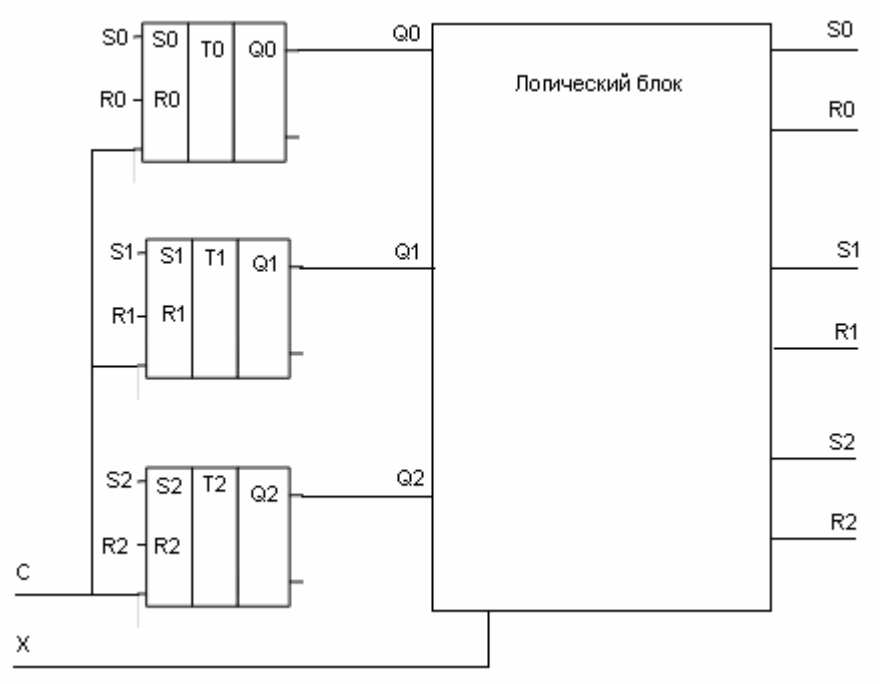


Рис. 3.3 – Логическая схема

Схема логического блока приведена ниже, также приведены сигналы для проверки работы схемы (согласно таблице истинности).

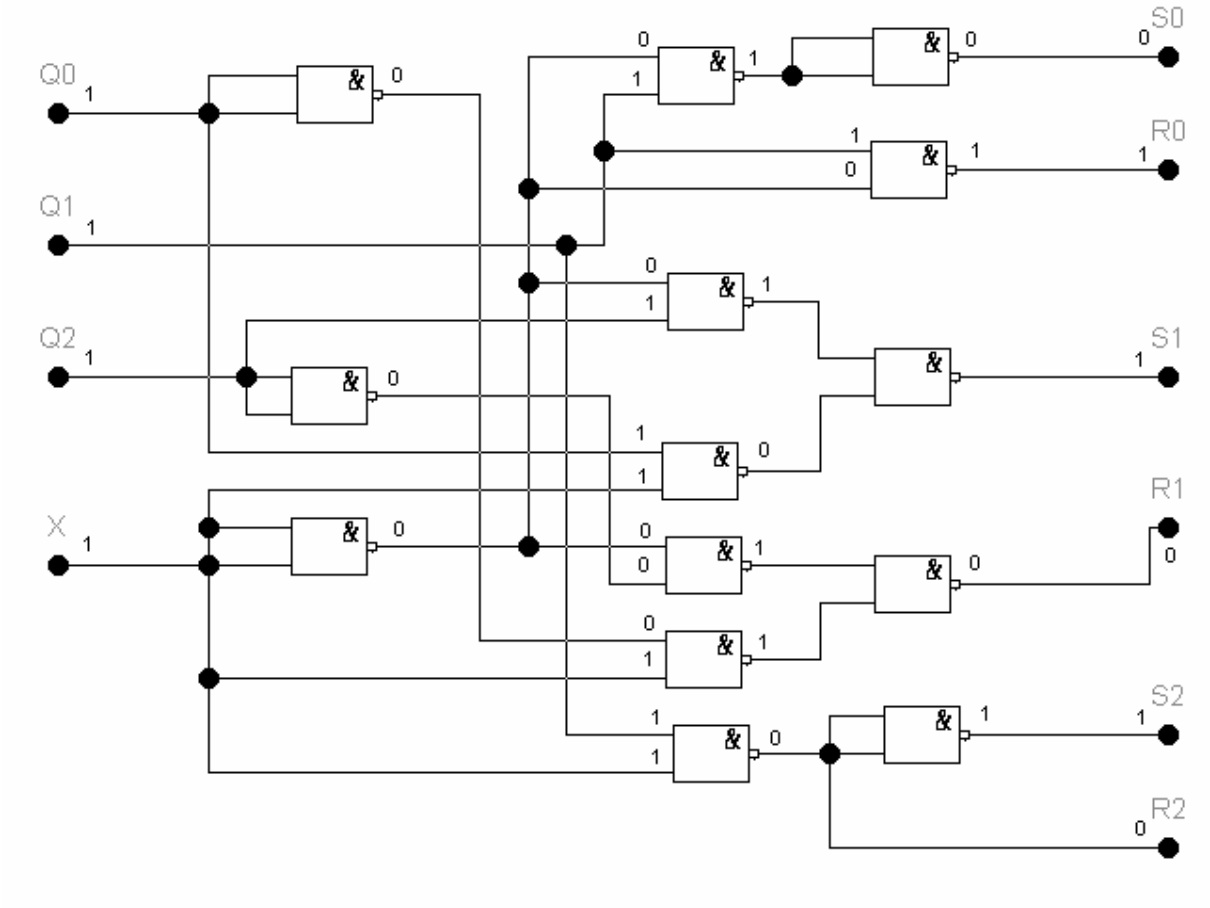


Рис. 3.4 – Схема логического блока

Изобразим эпюры сигналов:

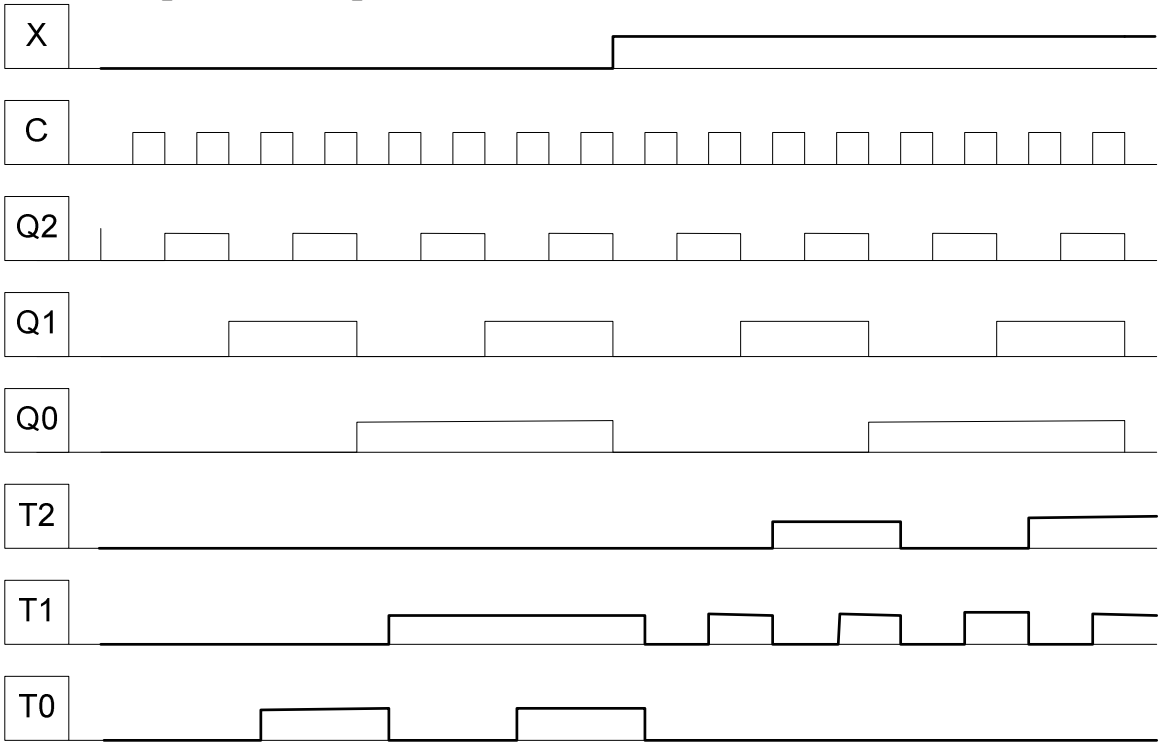


Рис. 3.5 – Эпюры сигналов

Вывод:

В результате выполненной работы была синтезирована логическая схема на RS-триггерах. Функционирование синтезированной схемы подтверждает анализ готового логического блока при подстановке значений из таблицы истинности для текущего состояния Q0=Q1=Q2=X=«1». При этом получены соответствующие сигналы на входах RS-триггеров (последняя строка таблицы истинности).

3.2 Методические указания для выполнения контрольной работы №2

Целью выполнения контрольной работы является закрепление и проверка знаний по разделу базовые структуры ЭВМ, процессоры, организация памяти, ввод-вывод путем программирования на ассемблере задач, близких к вычислительной системе.

Для выполнения контрольной работы необходимо выбрать вариант индивидуального задания: составить программу управления звуком с использованием подсистемы ввода-вывода и микросхемы системного таймера.

Системный таймер.

Практически все компьютеры содержат устройство, называемое системным таймером. Это устройство подключено к линии запроса на прерывание IRQ0 и вырабатывает прерывание INT 8H приблизительно 18,2 раза в секунду (точное значение – 1193180/65535 раза в секунду).

При инициализации BIOS устанавливает свой обработчик для прерывания таймера. Эта программа каждый раз увеличивает на 1 текущее значение 4-байтовой переменной, располагающейся в области данных BIOS по адресу 0040:006CH – счетчика тиков таймера. Если этот счетчик переполняется (т.е. прошло более 24 часов с момента запуска таймера), в ячейку 0040:0070H заносится 1.

Стандартный обработчик прерывания таймера осуществляет также контроль за работой двигателей НГМД. Если после последнего обращения к НГМД прошло более 2 секунд, обработчик прерывания выключает двигатель (ячейка с адресом 0040:0040H содержит время, оставшееся до выключения двигателя).

Последнее действие, которое выполняет обработчик прерывания таймера, это вызов прерывания INT 1CH. После инициализации системы вектор INT 1CH указывает на команду IRET, т.е. ничего не выполняется. Программа пользователя может установить собственный обработчик этого прерывания для того, чтобы выполнить какие-либо периодические действия.

Необходимо отметить, что прерывание INT 1CH вызывается до сброса контроллера прерываний, поэтому во время выполнения прерывания INT 1CH все аппаратные прерывания запрещены. Для сброса контроллера прерываний надо в порт 20H записать значение 20H.

Состав и режимы работы микросхемы таймера.

Таймеры 8253 и 8254 состоят из трех независимых каналов. Каждый канал содержит регистры:

- состояния канала RS (8 разрядов);
- управляющего слова PSW (8 разрядов);
- буферный регистр OL (16 разрядов);
- регистр счетчика CE (16 разрядов);
- регистр констант пересчета CR (16 разрядов).

Каналы таймера подключаются к внешним устройствам при помощи трех линий:

GATE – управляющий вход;

CLOCK – вход тактовой частоты;

OUT – выход таймера.

Регистр счетчика CE работает в режиме вычитания. Его содержимое уменьшается по заднему фронту сигнала CLOCK при условии, что на входе GATE установлен уровень логической 1. В зависимости от режима работы таймера при достижении счетчиком CE нуля тем или иным способом меняется выходной сигнал OUT.

Буферный регистр OL предназначен для запоминания текущего содержимого регистра счетчика CE без остановки процесса счета. После запоминания буферный регистр доступен программе для чтения.

Регистр констант пересчета CR может загружаться в регистр счетчика, если это требуется в текущем режиме работы таймера.

Возможны 6 режимов работы таймера. Они разделяются на три типа:

-режимы 0 и 4 – однократное выполнение функций;

-режимы 1 и 5 – работа с перезапуском;

-режимы 2 и 3 – работа с автозагрузкой.

В режиме однократного выполнения функций перед началом счета содержимое регистра констант CR переписывается в регистр счетчика CE по сигналу CLOCK, если сигнал GATE установлен в 1. В дальнейшем содержимое регистра CE уменьшается по мере прихода сигналов CLOCK. Процесс счета можно приостановить, если подать на вход GATE уровень логического 0. Если затем на вход GATE подать 1, счет будет продолжен дальше. Для повторения выполнения функции необходима новая загрузка регистра CR, т.е. повторное программирование таймера.

При работе с перезапуском не требуется повторного программирования таймера для выполнения той же функции. По фронту сигнала GATE значение константы из регистра CR вновь переписывается в регистр CE, даже если текущая операция не была завершена.

В режиме автозагрузки регистр CR автоматически переписывается в регистр CE после завершения счета. Сигнал на выходе OUT появляется только при наличии на входе GATE уровня логической 1. Этот режим используется для создания программируемых импульсных генераторов и генераторов прямоугольных импульсов (меандров).

Обычно задействованы все три канала таймера.

Канал 0 используется в системных часах времени суток. Этот канал работает в режиме 3 и используется как генератор импульсов с частотой примерно 18,2 Гц. Именно эти импульсы вызывают аппаратное прерывание INT 8H.

Канал 1 используется для регенерации содержимого динамической памяти компьютера. Выходная линия канала OUT связана с микросхемой прямого доступа к памяти (DMA), и ее импульс заставляет DMA регенерировать память.

Канал 2 связан с громкоговорителем (динамиком) компьютера и может быть использован для генерации различных звуков или музыки либо как генератор случайных чисел. Канал использует режим 3 таймера.

Программирование таймера.

Таймеру соответствуют 4 порта ввода/вывода со следующими адресами: 40H – канал 0; 41H – канал 1; 42H – канал 2; 43H – управляющий регистр. Формат байта управляющего регистра (управляющего слова) приведен в таблице 1.

Таблица 1 – Управляющий регистр

Биты	Значение
0	0 – двоичные данные; 1 – данные в форме BCD (двоично-десятичные)
3–1	номер режима: 000 – режим 0 (прерывание от таймера); 001 – режим 1 (программируемый ждущий мультивибратор); X01 – режим 2 (программируемый генератор импульсов); X11 – режим 3 (генератор меандра); 100 – режим 4 (программно-запускаемый одновибратор);

Окончание табл. 1

Биты	Значение
	101 – режим 5 (аппаратно-запускаемый одновибратор)
5–4 тип	операции: 00 – передать значения счетчика в буфер; 01 – читать/писать только старший байт; 10 – читать/писать только младший байт; 11 – читать/писать сначала младший байт, потом – старший
7–6 номер канала:	00 – канал 0; 01 – канал 1; 10 – канал 2; 11 – код команды RBC (чтение состояния канала)

Младший разряд байта определяет формат константы, использующейся для счета. В двоично-десятичном режиме константа задается в диапазоне 1 – 9999.

Для программирования канала таймера необходимо выполнить следующую последовательность действий:

- записать в управляющий регистр по адресу 43H управляющее слово;

- требуемое значение счетчика посылается в порт канала (адреса 40H – 42H), причем сначала посылается младший байт, а затем старший байт значения счетчика.

Сразу после этого канал таймера начинает выполнять требуемую функцию.

Генерация звуков и музыки

Звуки и музыку на ПЭВМ можно воспроизводить двумя способами:

- с использованием таймера;

- без использования таймера.

При поступлении на динамик электрического сигнала некоторого уровня происходит втягивание мембраны динамика, а при изменении уровня электрического сигнала – выталкивание. Таким образом периодический электрический сигнал вызывает вибрацию мембраны динамика, т.е. генерацию звука той же частоты.

Управление динамиком с использованием таймера.

Одно из наиболее распространенных применений таймера – генерация звуковых сигналов и воспроизведение музыки. Таймер позволяет воспроизводить музыку в фоновом режиме, т.е. во время работы программы может звучать музыка.

Канал 2 микросхемы 8254 связан с динамиком. Однако динамик не просто соединен с выходом OUT канала 2. Порт 61H также используется для управления динамиком (рис. 2).

Младший бит этого порта при установке в 1 разрешает работу канала, т.е. генерацию импульсов для динамика. Дополнительно для управления динамиком используется бит 1 порта 61H. Если этот бит установлен в 1, импульсы от канала 2 таймера смогут проходить на динамик.

Системный генератор импульсов (СГИ) независимо от типа и производительности компьютера IBM вырабатывает импульсы одной и той же частоты – 1 193180 Гц.

Таким образом, для включения звука надо выполнить следующие действия:

- запрограммировать канал 2 таймера на нужную частоту (т.е. загрузить в регистр счетчика канала значение, равное отношению частоты СГИ и требуемой частоты звука);

- для включения звука установить в 1 два младших разряда порта 61H.

Поскольку остальные разряды порта 61H используются в машине для других целей, они должны быть оставлены без изменения.

Для выключения звука надо сбросить два младших разряда порта 61H (не изменяя опять-таки состояния остальных разрядов).

Управление динамиком без таймера.

Для генерации звука без таймера надо сбросить младший бит порта 61H (запретив тем самым работу канала 2 таймера) и, управляя битом 1 этого порта, т.е. устанавливая этот бит то в 1,

то в 0, формировать импульсы для динамика. Высота генерируемого звука будет соответствовать периоду импульсов. Отметим, что этим способом можно генерировать импульсы любой скважности, что дает больше возможностей для создания различных звуковых эффектов.

Воспроизведение музыки

Мелодия, как известно, состоит из нот, разделенных или не разделенных паузами. При проигрывании мелодии необходимо для каждой ноты программировать соответствующим образом канал 2 таймера и включать динамик (с помощью порта 61H) на определенное время, равное длительности ноты. Затем программа должна выключить динамик и выдержать паузу, если потребуется, перед проигрыванием следующей ноты. В таблице 2 приведены частоты для нот 2-й октавы. Для других октав при понижении или повышении тона надо значения частот делить или умножать на два.

Таблица 2 – Частоты нот

Нота	Частота, Гц
До	267,7
До-диез	277,2
Ре	293,7
Ре-диез	311,1
Ми	329,6
Фа	349,2
Фа-диез	370,0
Соль	392,0
Соль-диез	415,3
Ля	440,0
Ля-диез	466,2
Си	493,9

Плавные переходы тонов производятся за счет непрерывного изменения частоты. Такой звуковой эффект можно сделать более выразительным, если немного уменьшить длительность каж-

дого тона при повышении звука или слегка увеличивать длительность при понижении.

Порядок выполнения контрольной работы

Воспроизвести «мелодию» согласно индивидуальному заданию. Для этого необходимо:

1. Программировать канал 2 таймера в режиме 3 на нужную частоту (определяемую нотой, см. табл. 2).
2. С помощью порта 61H задавать требуемую длительность звучания ноты и длительность паузы (длительность ноты в секундах задана в индивидуальном задании после наименования ноты).
3. Воспроизвести звуковой эффект без использования таймера.

Варианты задания

1. Си(0,5), до-диез(3), ля-диез(0,5).
2. До(2,5), фа(1), соль(2).
3. Фа(2), фа-диез(2), соль(2).
4. Ре-диез(2,5), ре(2,5), до-диез(2,5).
5. Ми(1), ре(2), до(1).
6. Ре(1,5), ля-диез(2), ре-диез(1).
7. Соль(2), ля(1), до-диез(0,5).
8. Ля(4), ре-диез(1), си(0,5).

Содержание отчета по контрольной работе

Тема контрольной работы.
Цель работы.
Индивидуальное задание.
Текст программы.
Результаты работы программы
Выводы.

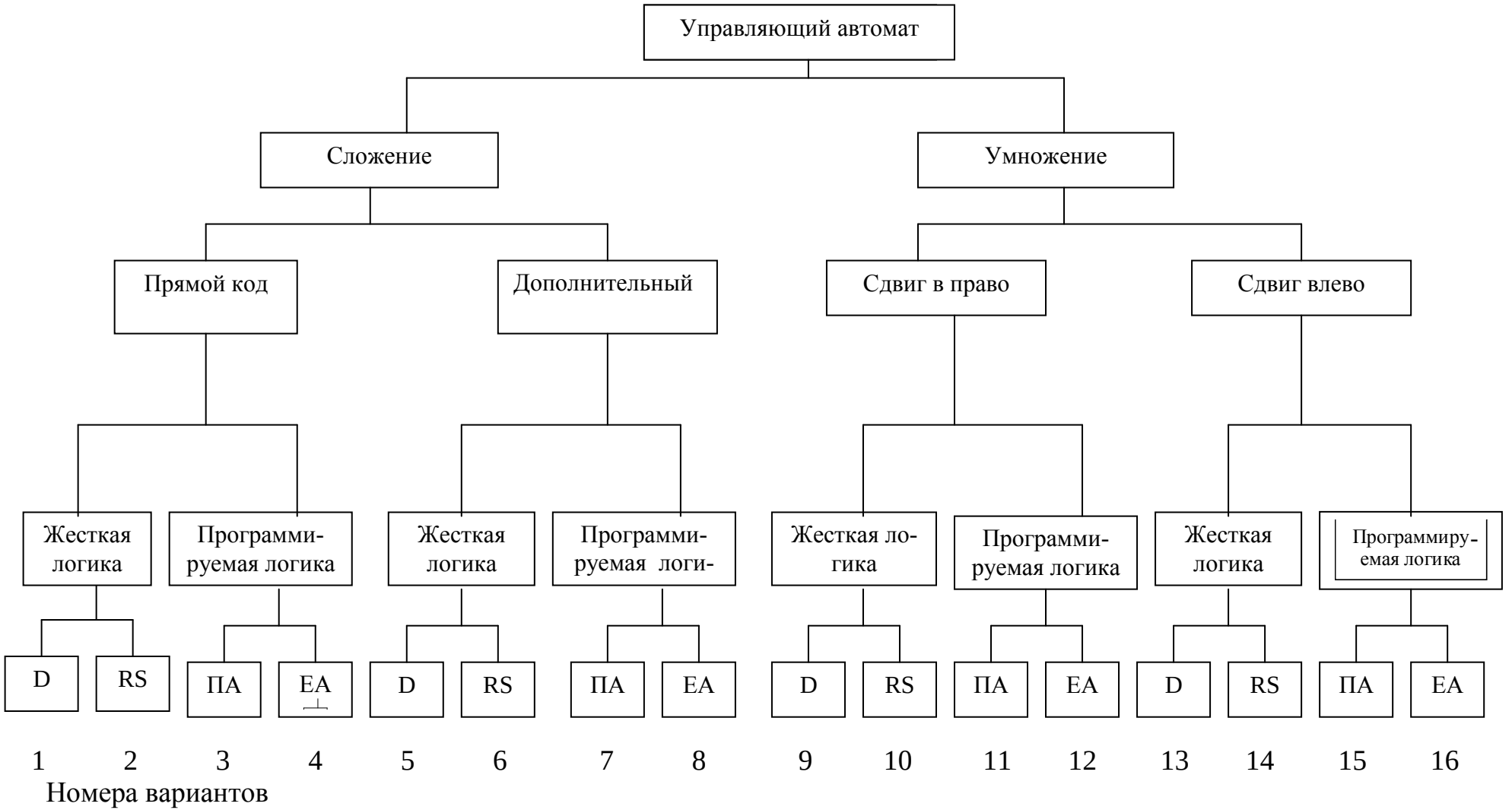


Рис. 3.6. – Иерархическая схема формирования вариантов контрольной работы № 3:
ЕА – естественная адресация, ПА – принудительная адресация, D – D-триггер, RS – RS-триггер

3.3 Методические указания для выполнения контрольной работы №3

Целью контрольной работы является необходимость научить студентов проектированию логических устройств для выполнения арифметическо-логических операций. Операционное устройство отображает принципы работы любого элемента ЭВМ, является составной частью любого микропроцессора, и поэтому проектирование ОУ для выполнения арифметическо-логических операций позволяет понять принцип работы и программирования микропроцессора.

Согласно концепции академика Глушкова В.М. любое операционное устройство состоит из операционного автомата, предназначенного для выполнения микроопераций, и управляющего автомата для формирования управляющих воздействий. Управляющий (микропрограммный) автомат (МПА) – конечный автомат, обеспечивающий выполнение микропрограммы (МП) операционным устройством (ОУ), состоящим из операционного и управляющего автоматов (ОА и УА).

Микропрограмма описывает алгоритм выполнения какой-либо сложной операции в терминах микроопераций (МО).

К микрооперациям относятся простейшие действия, выполняемые элементами операционного автомата (инверсия слова, сложение двух слов, сдвиг влево или вправо). МП наглядно представляется в виде ГСА, в вершинах которой записываются микрооперации или логические условия. Идентификаторами этих операций, условий являются обозначения элементов операционных автоматов.

Например:

RgA – регистр А.

RgA [0] – нулевой разряд регистра А.

СТ – содержимое счетчика и т.д.

В микрооперациях возможны следующие действия над переменными.

Присваивание:

RgA:= A

RgA:= RgA. – сдвиг вправо на 1 разряд;

.RgA:= .RgA – сдвиг влево на 1 разряд и т.д.

Микрооперации, которые можно выполнять одновременно, называются совместимыми и могут быть записаны в одной операторной вершине. МП, записанная в терминах микрооперации и логических условий, называется функциональной или содержательной.

Каждая МО в операционном автомате инициируется микрокомандой, которую вырабатывает управляющий автомат в зависимости от значений логических условий. По МП строится граф-схема алгоритма (ГСА) управляющего автомата. По топологии ГСА УА совпадает с МП ОА. Отличием являются дополнительная условная вершина в начале ГСА «пуск автомата» и дополнительная операторная – «операция выполнена».

В операционных вершинах ГСА УА записываются Y_i – микрокоманды, а в условных X_i – логические условия.

Существуют управляющие автоматы с жесткой (ЖЛ) и программируемой логиками (ПЛ). УА с ПЛ имеют широкое применение.

Для ЖЛ наибольшее распространение получили автоматы двух типов: Мили и Мура.

Различие автоматов состоит в том, что выходной сигнал у автомата Мили зависит не только от его состояния a_m, но и от входного сигнала x_t. У автомата Мура сигнал зависит только от a_m.

Обычно в автоматах используют синхронные триггеры, у которых реакция на входные сигналы приходит только при наличии сигнала синхронизации С (его положительным или отрицательным фронтом).

Последовательность действий при синтезе ОУ с ЖЛ:

1. Выбрать вариант задания (рис. 3.6).
2. Уяснить задачу по разработке ОУ для выполнения заданной арифметическо-логической операции.
3. Выполнить арифметическо-логическую операцию на содержательном уровне, (как это делает человек).
4. Выполнить арифметическо-логическую операцию так, как это производит компьютер.
5. Построить граф-схему алгоритма (ГСА).

6. Разметить ГСА для построения автомата Мили.

7. Построить граф переходов.

8. Сформировать таблицы управляющих и условных (осведомительных) сигналов.

9. Синтезировать операционный автомат с указанием управляющих и условных сигналов.

10. Выбрать триггеры, согласно варианту задания, для синтеза управляющего автомата.

11. Построить таблицу истинности для УА.

12. Выбрать базис (И-НЕ, ИЛИ-НЕ) для формирования сигналов управления для триггеров.

13. Получить СДНФ для булевых функций сигналов управления триггерами.

14. Минимизировать полученные СДНФ.

15. Синтезировать логическую схему управляющего автомата на заданных триггерах и по полученным булевым функциям.

16. Выполнить проверку функционирования УА на одном из переходов, обязательно с наличием условия.

17. Оформить контрольную работу.

При синтезе УА с ПЛ пункты 1–9 сохраняются.

1–9 так же, как и для ЖЛ.

10. Выбрать вариант построения УА с принудительной или естественной адресацией.

11. Заполнить по ГСА ячейки ПЗУ АУ.

12. Выполнить проверку функционирования УА на одном из переходов, обязательно с наличием условия.

Задание: Спроектировать операционное устройство (ОУ), состоящее из ОА и УА, выполняющее арифметическую операцию. Для выбора номера варианта нужно воспользоваться иерархической схемой, приведенной на рис. 3.6. Зная свой номер, начинают движение по схеме снизу. Двигаясь по ветвям схемы, определяют параметры варианта. Например, номер 5: D-триггер, жесткая логика, дополнительный код, сложение.

Пример выполнения контрольной работы №3

Содержание.

1. Задание.
2. Алгоритм в содержательной форме.
3. Граф-схема алгоритма.
4. Синтез операционного автомата.
5. Синтез управляющего автомата.

Любое операционное устройство, согласно концепции академика Глушкова В.М., состоит из операционного автомата (А0), предназначенного для выполнения простейших операций (микроопераций) и устройства управления (УА), формирующего управляющие сигналы в зависимости от результатов выполнения операций.

Операционный автомат (ОА) предназначен для выполнения арифметических и логических операций под управлением управляющего автомата (УА). ОА включает в себя регистры, сумматор, каналы передачи информации. Процесс функционирования операционного автомата распадается на определенную последовательность элементарных действий в его узлах. Перечень элементарных действий включает в себя:

1) установку регистров в некоторое состояние;
2) инвертирование содержимого регистров;
3) пересылку содержимого одного узла в другой узел;
4) сдвиг содержимого регистров влево или вправо;
5) счет, при котором число в счетчике возрастает или убывает на единицу;
6) выполнение операции сложения;
7) сравнение на равенство содержимого регистра с некоторым числом;
8) логические действия (поразрядная дизъюнкция, конъюнкция).

Каждое такое элементарное действие, выполняемое из узлов операционного автомата в течение одного тактового периода, называется микрооперацией. Из запоминающего устройства в операционные устройства поступает операнд, над которым выполняется некоторая операция. Результат операций обычно сохраняется в операционном устройстве и может быть использован в каче-

стве операнда в следующей операции. Наименование выполняемой операции задается устройством управления (ЦУ), из которого на операционное устройство поступает код операции. Из операционного устройства в устройство управления посылаются сигналы об окончании операции, о переполнении разрядной сетки операционного устройства и т.д.

Поскольку на автомат возлагается управление выполнением некоторого набора операций, то закон функционирования автомата должен определяться исходя из набора соответствующих микропрограмм. Микропрограммы, входящие в заданный набор, должны быть объединены между собой и иметь одну начальную и одну конечную вершины.

Вариант операционного автомата приведен в рабочих тетрадях, глава 4.

2. Содержательная форма алгоритма

Выполним операцию умножения в содержательной форме двух двоичных четырехразрядных чисел так, как это делает человек.

```
      1 1 1 0     – множимое
      1 0 1 0     – множитель
      0 0 0 0     –1-ая часть суммы
    1 1 1 0       – 2-ая часть суммы
  0 0 0 0         – 3-ая часть суммы
1 1 1 0           – 4-ая часть суммы
– – – – – – – – – – –
1 0 0 0 1 1 0 0   – результат выполнения операции  множения.
```

3. Выполнение арифметической операции в виде машинных процедур

В исходном состоянии регистры RgA и RgB хранят соответственно множимое и множитель. Регистр RgC, предназначенный для хранения и сдвига суммы частичных произведений, установлен в нулевое состояние.

Анализируется содержимое младшего разряда регистра RgB. Так как в рассматриваемом примере этот разряд содержит единицу, то 1-е частичное произведение равно множимому, и оно прибавляется к содержимому регистра RgC, используемому для

накопления суммы частичных произведений. Далее производится сдвиг на один разряд вправо содержимого регистра RgC3, причем выдвигаемый при сдвиге из регистра RgC младший разряд числа (не принимающий участия в последующем суммировании частичных произведений) может передаваться в освобождающийся при сдвиге старший разряд регистра RgB. В младшем разряде регистра RgB оказывается 2-й разряд множителя. Анализ младшего разряда регистра RgB вновь обнаруживает в нем единицу и производится прибавление множимого (2-го частичного произведения) к сдвинутому вправо 1-му частичному произведению в регистре RgC. В регистре RgC образуется сумма двух первых частичных произведений. При суммировании может возникнуть перенос из старшего разряда, его необходимо запомнить, вдвигая затем при сдвиге в старший разряд регистра RgC. Производится очередной сдвиг содержимого регистров RgC и RgB.

Таблица 1.1

Множимое (RgA)	Старшие разряды произведения (Кз)	Множитель и младшие разряды произведения (RgB)	Выполняемое действие
1101	0000	1011	Исходное состояние
	+		
	1101		
	0 1101		Суммирование
	0 0110	1101	Сдвиг (R3) и (R2)
	+		
	1101		
	10011		Суммирование
	01001	1110	Сдвиг (R3) и (R2)
	00100	1111	Сдвиг (R3) и (R2)
	+		
	1101		
	10001		Суммирование
	01000	1111	Сдвиг (R3) и (R2)
	П Р О И З В Е Д Е Н И Е		

В младшем разряде регистра RgB обнаруживается нуль (3-й разряд множителя), частичное произведение равно нулю, и суммирование не производится. Осуществляется очередной сдвиг.

В младшем разряде регистра RgB выявляется единица (4-й разряд множителя), прибавляется очередное частичное произведение, и после выполнения сдвига регистр RgC содержит старшие разряды произведения, а регистр RgB – младшие разряды произведения.

Из приведенного описания видно, что процессы при выполнении умножения носят циклический характер. В каждом повторении цикла выполняются следующие действия: анализируется содержимое младшего разряда регистра RgB; если оно равно единице, производится прибавление множимого к содержимому RgC; осуществляется сдвиг содержимого регистров RgC и RgB. Число повторений цикла равно числу разрядов множителя.

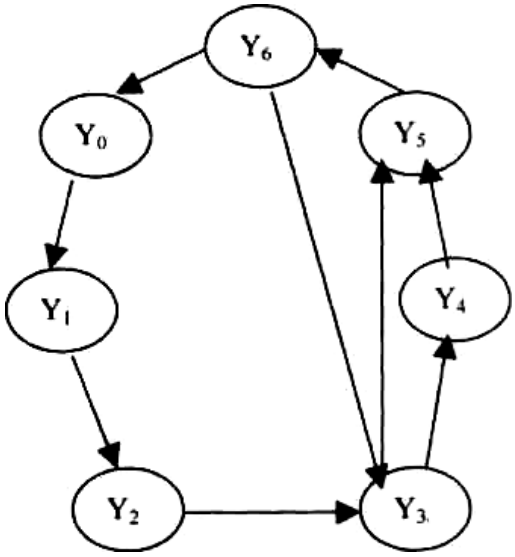


Рис. 3.7 – Граф переходов

4. Граф-схема алгоритма для операции умножения

В граф-схеме представлена структура алгоритма умножения двух чисел со сдвигом вправо.

На первой итерации обнуляем содержимое регистров.

На второй итерации присваиваем регистру A и B и счетчику значения.

На третей итерации сдвигаем значение регистра B вправо на один разряд.

На четвертой итерации анализируем младший разряд множителя: если 1, то к содержимому регистра C (сумме частичных произведений) прибавляем множимое, если нет, то к содержимому регистра C ничего не прибавляем и сдвигаем его на разряд вправо.

После чего вычитаем из счетчика единицу и проверяем, равен он нулю или нет, если равен, то конец, если нет, то сдвигаем регистр B и повторяем заново.

Граф-схема алгоритма и логические схемы управляющих автоматов с жесткой и программируемой логиками приведены в рабочих тетрадях, глава 4.

4 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО РАБОТЕ С ELECTRONICS WORKBENCH (EWB)

Программный комплекс EWB разработан фирмой Interactive Image Technologies (Канада) для схемотехнического моделирования цифровых и аналоговых радиоэлектронных устройств.

Предварительное исследование электронной схемы с применением компьютерного моделирования позволяет найти оптимальные параметры для работы исследуемого устройства, не прибегая к его практической реализации. Исследование на программной модели позволяет ознакомиться с возможностями проверки правильности построения схем. При разработке сложных схем физическое моделирование бывает просто невозможно из-за чрезвычайной сложности устройства.

Особенность программы EWB в наличии в ней контрольно-измерительных приборов, по внешнему виду, органам управления и характеристикам максимально приближенных к их промышленным аналогам.

Программа EWB 4.1 рассчитана для работы в среде Windows 3.xx или 95.98 и занимает около 5 Мбайт дисковой памяти, EWB 5.0 – в среде Windows 95/98 и NT 3.51, требуемый объём дисковой памяти – около 16 Мбайт. Для размещения временных файлов требуется дополнительно 10.....20 Мбайт свободного пространства.

В данном руководстве рассматривается версия EWB5PRO и EWB v.5.12.

Структура окна и система меню

Окно содержит строку команд меню, строку основных типовых электронных устройств, поле для составления исследуемой схемы и полосы управления прокруткой.

Основные команды меню:

Меню File:

первые четыре команды меню типовые и пояснений не требуют.

- **Revent to Saved** – стирание всех изменений, внесенных в текущем сеансе редактирования, и восстановление схемы в первоначальном виде.

- **Install –** установка дополнительных программ с жёстких дисков.

- **Import** – импорт текстовых файлов описания схемы.

- **Export –** составление текстового описания схемы и задания на моделирование в формате SPICE.

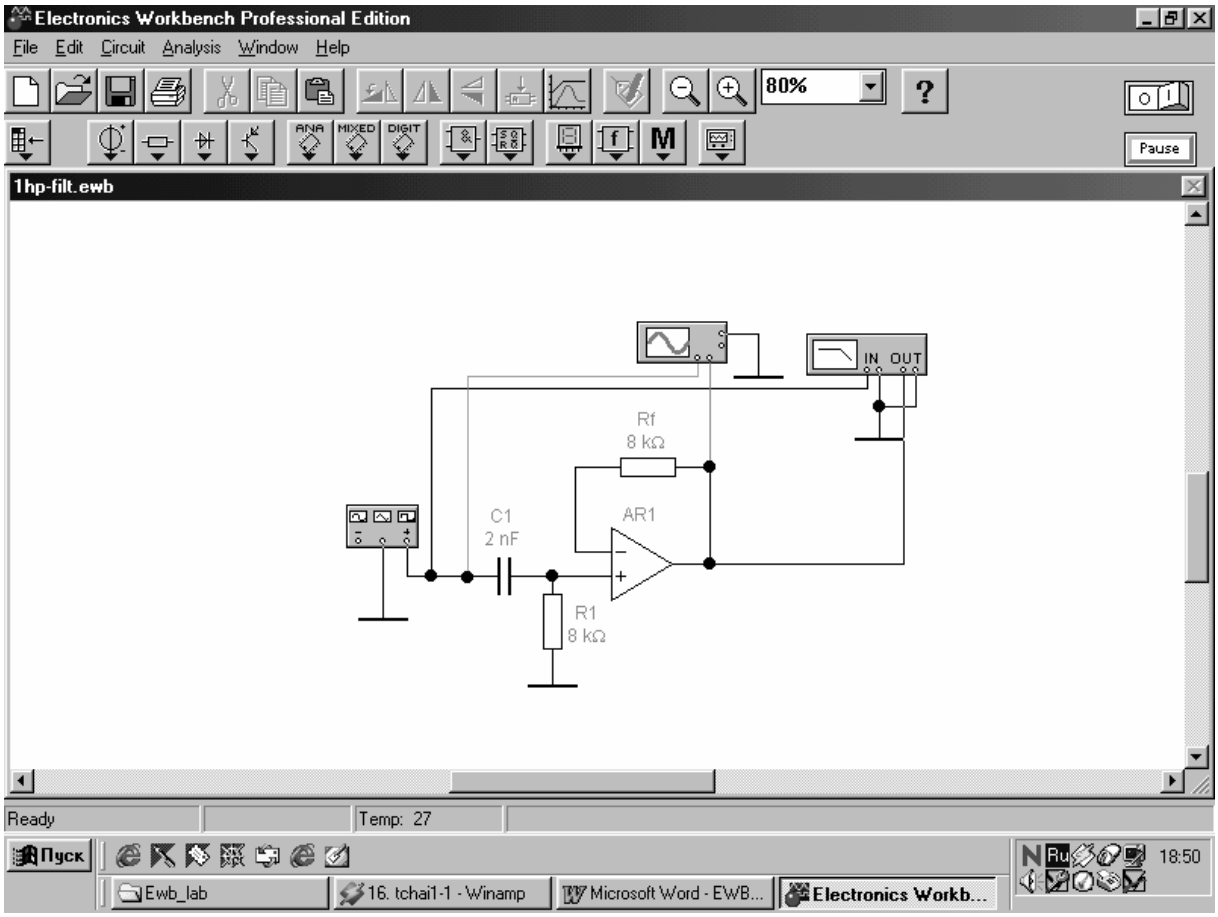


Рис. 4.1 – Окно программы EWB 5.x.

Меню Edit:.

- **CUT –** стирание (вырезание) выделенной части схемы с сохранением в буфере обмена. Выделение одного компонента производится щелчком мыши на изображении компонента. Для выделения части схемы или нескольких компонентов курсор мыши в левый угол воображаемого прямоугольника, охватывающего выделяемую часть, нажать левую кнопку мыши и, не

отпуская её, протянуть курсор по диагонали этого прямоугольника, контуры которого появляются уже в начале движения мыши, и затем отпустить кнопку. Выделенные компоненты окрашиваются в красный цвет.

- **COPY –** копирование выделенной части схемы в буфер обмена.

- **PAST –** вставка содержимого буфера обмена на рабочее поле программы. Фрагмент затем ещё будучи отмеченным перетаскивается с помощью мыши в нужное место.

- **DELETE –** стирание выделенной части схемы.

- **SELECT ALL –** выделение всей схемы.

- **COPYBITS –** команда превращает курсор мыши в крестик, которым по правилу прямоугольника можно выделить нужную часть экрана, после отпускания левой кнопки мыши выделенная часть копируется в буфер обмена, после чего его содержимое может быть импортировано в любое приложение Windows. Копирование всего экрана производится нажатием клавиш Print Screen; копирование активной в данный момент части экрана, например, диалогового окна - комбинацией Alt+Print Screen.

- **Show Clipboard –** показать содержимое буфера обмена.

- **Copy as Bitmap –** копирует выделенный участок в буфер обмена.

Меню Circuit – используется при подготовке схем, а также для задания параметров моделирования.

- **Activat** – запуск моделирования.

- **Stop** – остановка моделирования. Эти две команды дублируются нажатием кнопки выключателя, расположенного в правом верхнем углу экрана.

- **Pause –** прерывание моделирования.

- **Label –** ввод позиционного обозначения выделенного компонента с помощью диалогового окна.

- **Value** – изменение номинального значения параметра компонента с помощью диалогового окна.

- **Model** – выбор модели компонента, команда выполняется также двойным щелчком по компоненту. Работа с меню, как и во всех других подобных случаях, заканчивается нажатием кнопок

Accept или **Cancel –** с сохранением или без сохранения введённых изменений.

- **Zoom** – раскрытие (развёртывание) выделенной подсхемы или контрольно-измерительного прибора, команда выполняется также двойным щелчком мыши по иконке компонента или прибора.

- **Rotate** – вращение выделенного компонента.

- **Fault –** имитация неисправности выделенного компонента путём введения:

  - leakage – сопротивления утечки,
  - short – короткого замыкания,
  - open – обрыва,
  - none – отсутствие неисправности (включено по умолчанию).

- **Subcircuit** – преобразование предварительно выделенной части схемы в подсхему.

- **Wire Color** – изменение цвета предварительно выделенного проводника. Расцветка проводников важна в случае применения логического анализатора, – в этом случае цвет проводника определяет цвет временной диаграммы.

- **Preferences –** выбор элементов оформления схемы в соответствии с меню.

Для создания схем, рассматриваемых в рамках лабораторных работ по дисциплине «Организация ЭВМ и систем» достаточно воспользоваться имеющимися типовыми компонентами.

Для открытия нужной библиотеки компонентов нужно подвести курсор мыши к соответствующей иконке и нажать один раз её левую кнопку. В выпадающем множестве выбирается необходимый значок, и передвигается при удержании левой клавишей мыши на рабочее поле программы. Для установки параметров необходимо двойным нажатием левой кнопкой мыши раскрыть меню настройки параметров компонента. Выбор подтверждается нажатием кнопкой **Accept** и клавишей **Enter**.

После размещения компонентов производится соединение их выводов проводниками. При этом необходимо учитывать, что к выводу компонента можно подключить только один проводник.

Для выполнения подключения курсор мыши подводится к выводу компонента и после появления прямоугольной площадки

синего цвета, нажимается левая кнопка и появляющийся при этом проводник протягивается к выводу другого компонента до появления на нём такой же прямоугольной площадки, после чего кнопка мыши отпускается и соединение готово. При необходимости подключения к этим выводам других проводников в библиотеке Passive выбирается точка (символ соединения) и переносится на ранее установленный проводник. После удачной постановки точки к проводнику подсоединяется ещё два проводника.

Точка соединения может быть использована не только для подключения проводников, но и для введения надписей.

Если необходимо переместить отдельный сегмент проводника, к нему подводится курсор, нажимается левая кнопка и после появления в вертикальной или горизонтальной плоскости двойного курсора производятся нужные перемещения.

Подключение к схеме контрольно-измерительных приборов производится аналогично. Причём для таких приборов, как осциллограф или логический анализатор, соединения целесообразно проводить цветными проводниками, поскольку их цвет определяет соответствующую осциллограмму.

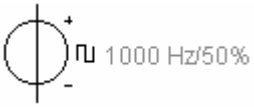
Основные компоненты EWB

Компонент **Выход из EWB.**

Вспомогательные компоненты **– группа SOURCES**:

- заземление (метка), точка нулевого потенциала в схеме;

- источник фиксированного напряжения +5 вольт;

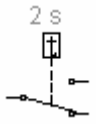


- генератор однополярных прямоугольных импульсов (амплитуда, частота, коэффициент заполнения).

Основные пассивные элементы – **группа BASIC**:

- точка соединения проводников, используется также для введения на схему надписей длиной не более 14 символов (других способов введения текста в EWB не существует);

- переключатель, управляемый нажатием задаваемой клавишей клавиатуры (в квадратных скобках), по умолчанию – клавиша пробела;



- переключатель, автоматически срабатывающий через заданное время на включение и выключение (время в секундах).

Индикаторные приборы – **группа INDICATORS**.

- светоиндикатор (свет свечения может быть настроен красным, зелёным и синим);

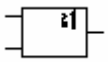
- семисегментный индикатор с дешифратором;

- семисегментный индикатор;

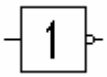
- лампа накаливания.

Логические элементы – **группа LOGIC GATES.**

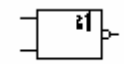
- логический элемент «И»;



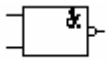
- логический элемент «ИЛИ»;



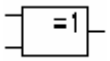
- логический элемент «НЕ»;



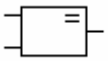
- логический элемент «ИЛИ-НЕ»;



- логический элемент «И-НЕ»;

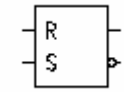


- логический элемент исключающее «ИЛИ»;

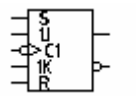


- логический элемент импликация.

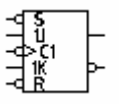
Комбинированные цифровые компоненты.



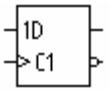
- асинхронный RS-триггер;



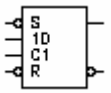
- универсальный JK-триггер с прямым тактовым входом и входами предустановки.



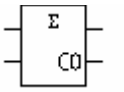
- универсальный JK-триггер с инверсным тактовым входом и инверсными входами предустановки;



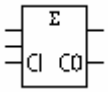
- D-триггер без предустановки;



- D- со входами предустановки;

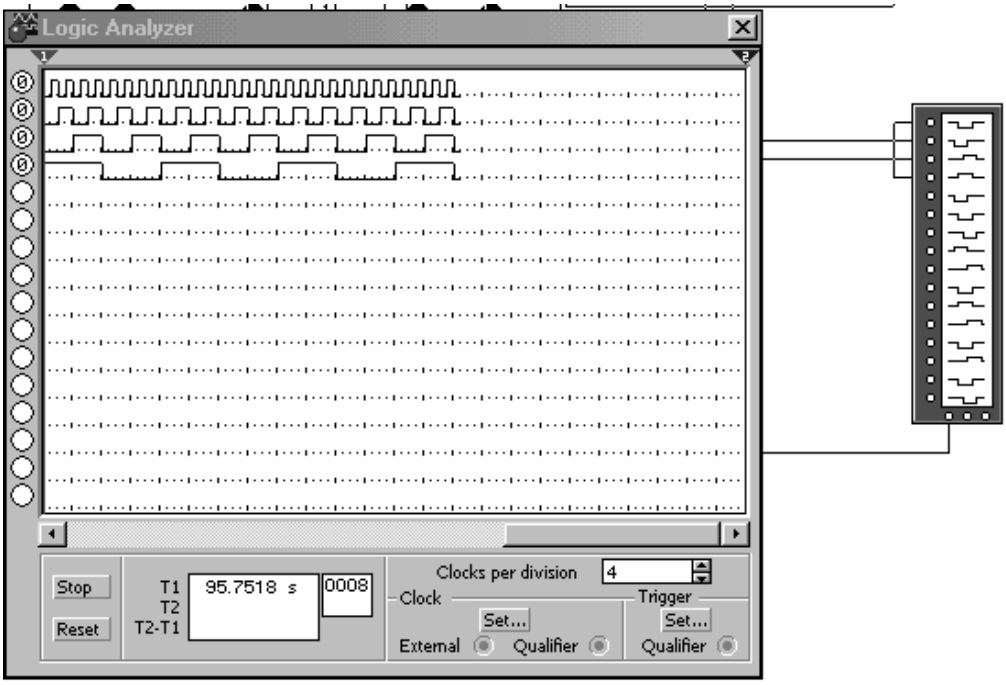


- полусумматор;

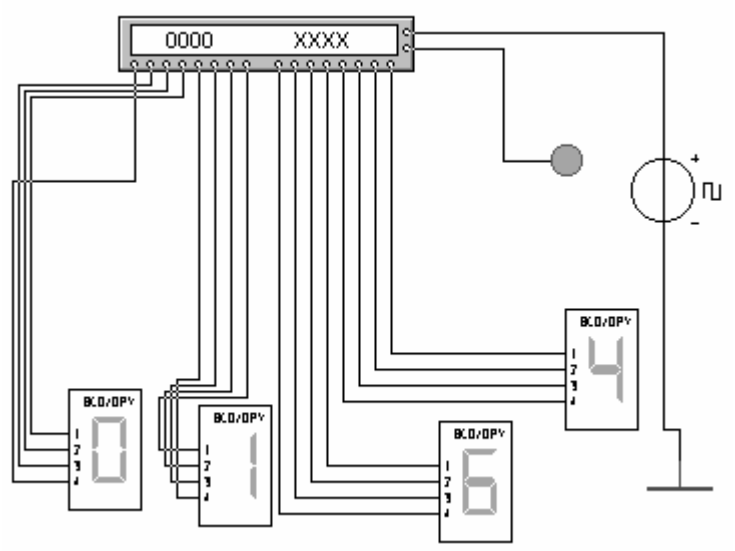


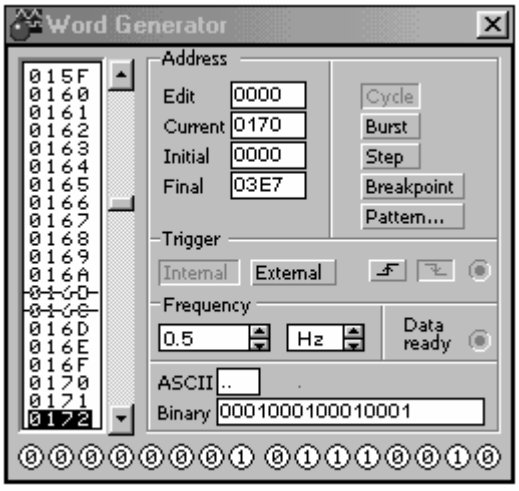
- полный сумматор.

Приборы, группа **INSTRUMENTS.**



- логический анализатор;





- генератор слова – **Word Generator.**

На первом рисунке показан генератор слова с подключенными семисегментными индикаторами и внешним генератором синхроимпульсов.

На втором рисунке генератор слова показан в развёрнутом виде.

Генератор (или кодовый генератор) предназначен для генерации 16-ти разрядных двоичных слов, которые набираются пользователем на экране, расположенным в левой части лицевой панели. Для набора двоичных комбинаций необходимо щёлкнуть мышью на соответствующем разряде и затем ввести с клавиатуры число в десятичном коде.

Сформированные слова выдаются на шестнадцать расположенных в нижней части прибора выходных клемм-индикаторов:

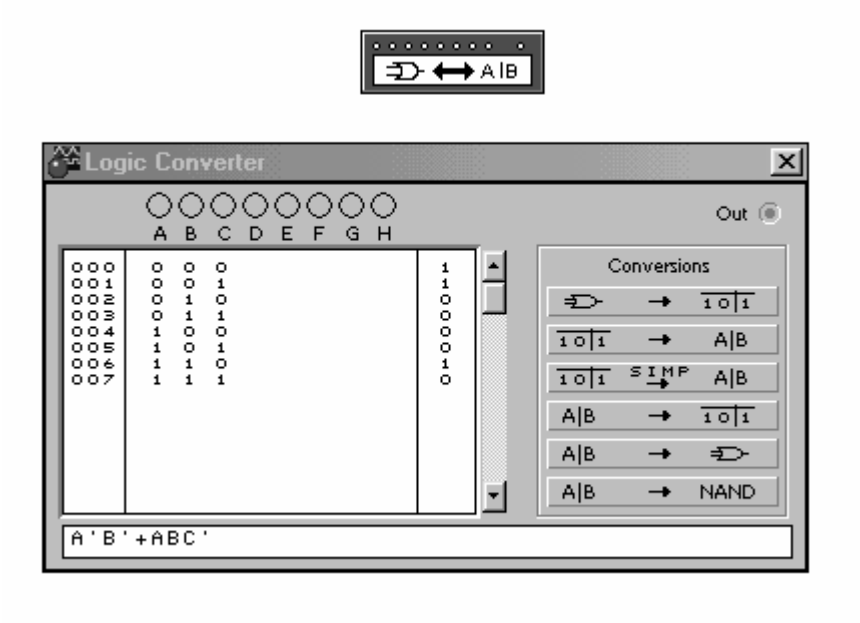
- с индикацией в двоичном коде в строке окна binary;

- в пошаговом (step), циклическим (cycle) или с выбранного слова до конца (при нажатии кнопки BURST) при заданной частоте посылок (установка – заданием частоты в окнах FREQUENCY);

- при внутреннем или внешнем запуске (при нажатии кнопки EXTERNAL, справа верхняя клемма служит для подключения сигнала синхронизации);

- призапуске по переднему или заднему фронту сигнала синхронизации служит кнопка

- на правую нижнюю клемму выдается выходной синхронизирующий импульс.



Логический преобразователь – **Logic Converter.**

На лицевой панели преобразователя показаны клеммы-индикаторы входов A,B,.....H и одного выхода OUT, экран для отображения таблицы истинности исследуемой схемы, экран-строка для отображения её булева выражения (в нижней части).

Логический анализ n-входового устройства с одним выходом может осуществлять следущюие действия, используя кнопки управления:

1.

- таблицу истинности исследуемого устройства;

2.

- булево выражение, реализуемое устройством;

3.

- минимизированное булево выражение;

4.

- схему устройства на логических элементах без ограничения их типа;

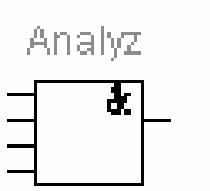
5.

- схему устройства только на логических элементах И-НЕ.

Пример составления исследуемой схемы.

Собрать схему логического элемента «И».

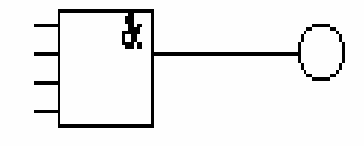
В группе Logic Gates, выбирается логический элемент «И».



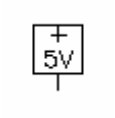
Двумя щелчками мыши на изображении логического элемента переходим к настройкам параметров логического элемента «И». Выбираем количество входов, например 4.

Можно присвоить название логическому элементу.

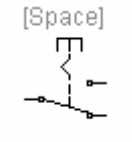
К выходу логического элемента присоединяем из группы **INDICATORS** красный светодиод.



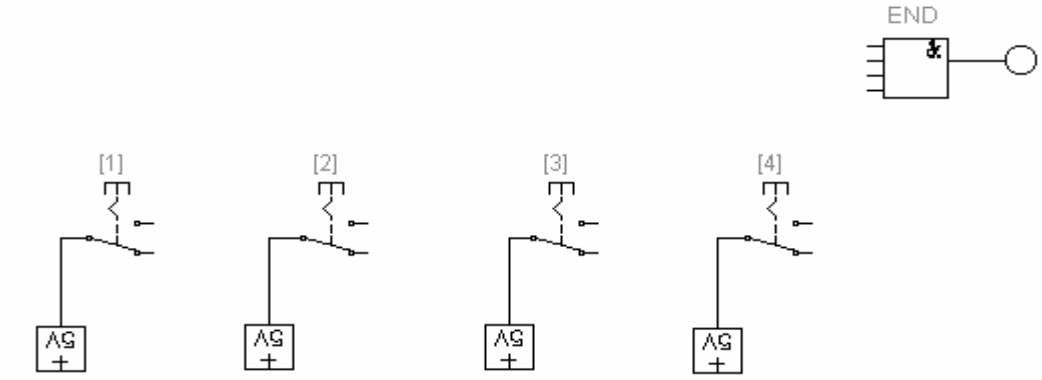
Для получения логического сигнала (0 или 1) удобно воспользоваться источником напряжения



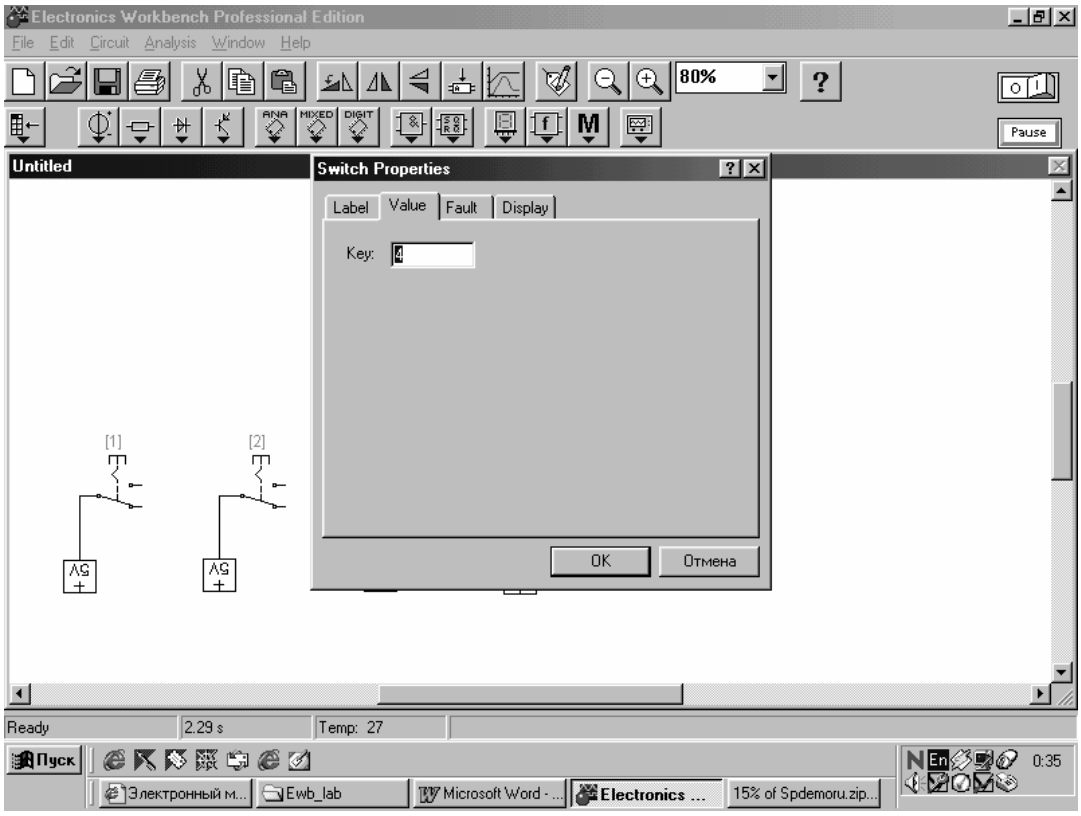
и переключателем



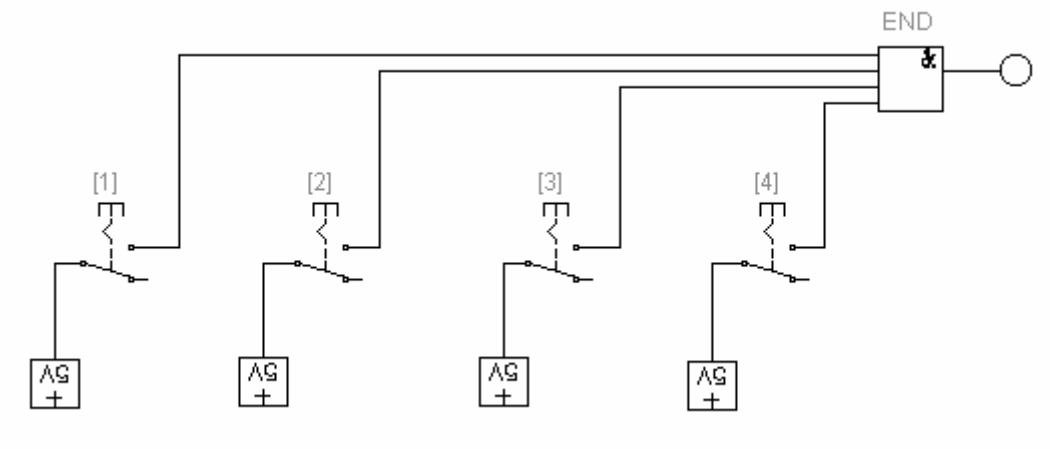
Затем набираем 4 источника и 4 переключателя



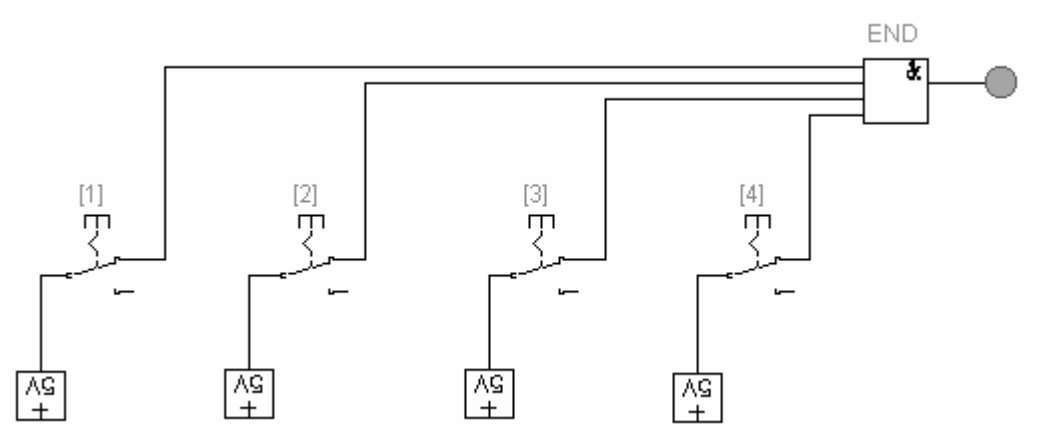
При этом присваиваем каждому переключателю клавишу переключения.



Затем соединяем входы логической схемы «И» с каждым из переключателей.



Проверка состоит в подаче различных кодовых комбинаций на вход логической схемы.



На выходе логической схемы «И» появляется логическая 1 (горит светодиод) только при подаче логических 1 (потенциал 5 вольт) на все четыре входа логической схемы «И».

5 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ВЫПОЛНЕНИЯ КОНТРОЛЬНЫХ РАБОТ НА АССЕМБЛЕРЕ

Структурная схема микропроцессора, представленная на рис. 5.1, включает: устройство управления (УУ), арифметико-логическое устройство (АЛУ), блок преобразования адресов и регистры.

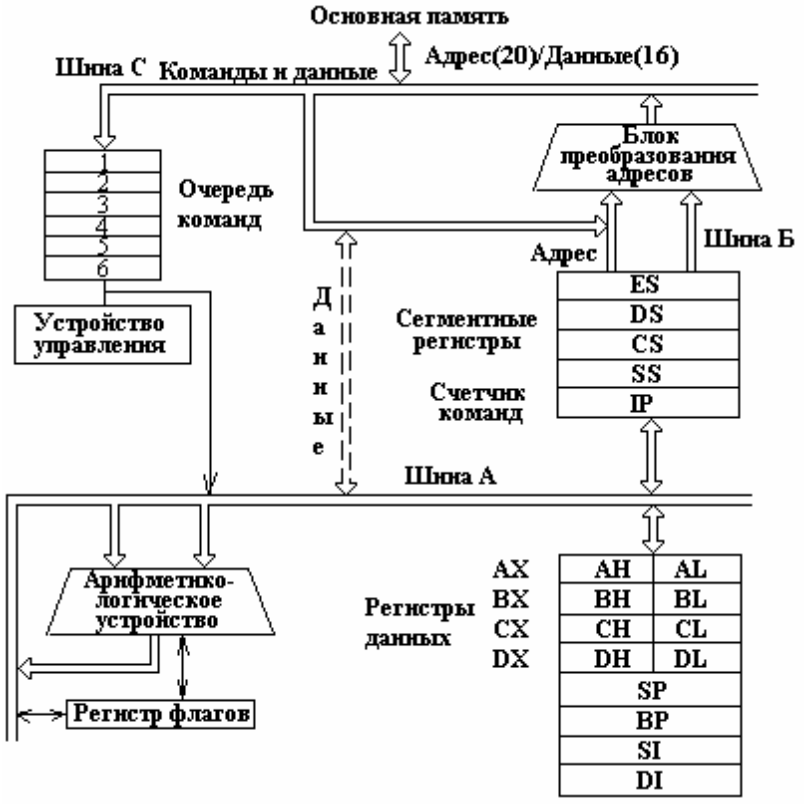


Рис. 5.1 – Структурная схема микропроцессора

Устройство управления дешифрирует коды команд и формирует необходимые управляющие сигналы. Арифметико-логическое устройство осуществляет необходимые арифметические и логические преобразования данных. В блоке преобразования адресов формируются физические адреса данных, расположенных в основной памяти. Наконец, регистры используются для хранения управляющей информации: адресов и данных.

Всего в состав микропроцессора семейства i8086 входит четырнадцать 16-битовых регистров:

а) четыре регистра общего назначения (регистры данных):

AX – регистр-аккумулятор,

BX – базовый регистр,

CX – счетчик,
DX – регистр-расширитель аккумулятора;

б) три адресных регистра:
SI – регистр индекса источника,
DI – регистр индекса результата,
BP – регистр-указатель базы;

в) три управляющих регистра:
SP – регистр-указатель стека,
IP – регистр-счетчик команд,
регистр флагов;

г) четыре сегментных регистра:
CS – регистр сегмента кодов,
DS – регистр сегмента данных,
ES – регистр дополнительного сегмента данных,
SS – регистр сегмента стека.

Минимальной адресуемой единицей основной памяти ПЭВМ является байт, состоящий из 8 бит. Доступ к байтам основной памяти осуществляется по номерам (номер байта является его физическим адресом в устройстве памяти).

Для адресации основной памяти в микропроцессоре i8086 предусматриваются 20-битовые адреса, что позволяет работать с основной памятью до 1 Мбайта.

Физический адрес формируется из 16-битового смещения и содержимого 16-битового сегментного регистра, сдвинутого влево на 4 бита (см. рис. 5.2).

Рис. 5.2 – Формирование физического адреса

Для размещения программ и данных в основной памяти выделяются специальные области – сегменты. Адреса этих областей хранятся в специальных сегментных регистрах.

Каждый из четырех сегментных регистров используется для хранения адреса определенного сегмента (см. рис. 5.3):

- сегмента кодов, т. е. области программ;
- сегмента данных, т. е. области размещения данных;
- дополнительного сегмента данных, используемого некоторыми командами;
- сегмента стека, т.е. области размещения стека.

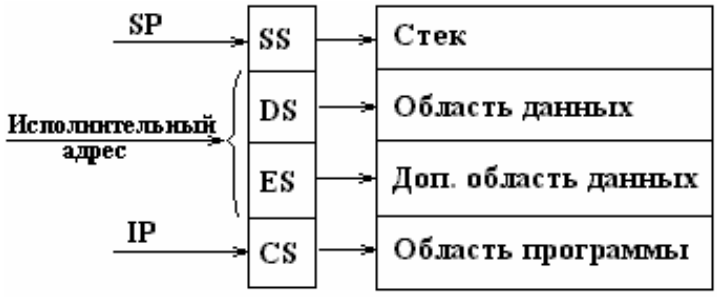


Рис. 5.3 – Сегментные регистры

Стек представляет собой специальным образом организованную область памяти, допускающую последовательную запись элементов данных длиной 2 байта (слово) и чтение их в порядке, обратном порядку записи. Для хранения адреса последнего слова, занесенного в стек, служит регистр-указатель стека **SP** (см. рис. 5.4, где а – текущее состояние стека, б – запись X, в – чтение X).

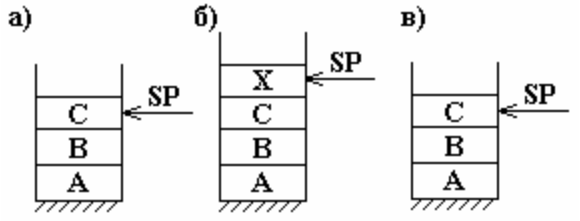


Рис. 5.4 – Организация стека

Стек используется для временного хранения данных и адресов, например при вызове подпрограмм, когда в стек заносится адрес возврата и значения параметров, передаваемых в подпрограмму.

Формат команд микропроцессора позволяет указывать в команде только один операнд, размещенный в основной памяти, т.е. одной командой нельзя, например, сложить содержимое двух ячеек памяти.

Принципиально допускается 8 способов задания смещения (исполнительного адреса) операндов, размещенных в основной памяти:

SI + <индексное смещение>
DI + <индексное смещение>
BP + <индексное смещение>
BX + <индексное смещение>
BP + **SI** + < индексное смещение>
BP + **DI** +< индексное смещение>
BX + **SI** + <индексное смещение>
BX + **DI** + <индексное смещение>

Во всех случаях исполнительный адрес операнда определяется как сумма содержимого указанных регистров и индексного смещения, представляющего собой некоторое число (одно- или двухбайтовое).

Содержимое регистров **CS** и **IP**, в которых хранится базовый адрес сегмента кодов и смещение очередной команды относительно начала сегмента, определяет физический адрес команды, которая должна быть выполнена на следующем шаге.

По указанному адресу из основной памяти считывается команда и пересылается в микропроцессор. Команда длиной от 1 до 8 байт помещается в очередь команд, откуда поступает в устройство управления, где дешифрируется.

Если при выполнении команды требуются данные, расположенные в основной памяти, то специальные полем кода команды определяется способ адресации и вычисляется исполнительный, а затем и физический адрес данных.

Данные, считанные из основной памяти по указанному адресу, пересылаются в регистр данных или в арифметико-логическое устройство и обрабатываются в соответствии с кодом команды. Результат помещается либо в регистры, либо (в соответствии с командой) в какую-нибудь область основной памяти.

Если выполненная команда не являлась командой передачи управления, то содержимое регистра **IP** увеличивается на длину выполненной команды, в противном случае в регистр **IP** заносится исполнительный адрес команды, которая должна выполняться следующей. Затем процесс повторяется.

На рис. 5.5 представлен флажковый регистр микропроцессора i8086, в котором в виде однобитовых признаков по принципу ДА – НЕТ (ВКЛЮЧЕНО – ВЫКЛЮЧЕНО) фиксируется информация о результатах выполнения некоторых команд, например арифметических.

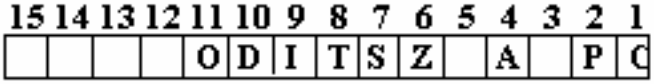


Рис. 5.5 – Регистр флагов:
O – признак переполнения;
D – признак направления;
I – признак прерывания;
T – признак трассировки;
S – признак знака: 1 – число < 0, 0 – число > 0
Z – признак нуля: 1 – число = 0
A – признак переноса из тетрады;
P – признак четности;
C – признак переноса

В последующем эта информация может использоваться, например, командами условной передачи управления.

Формат операторов ассемблера

Операторы языка ассемблера ПЭВМ имеют следующий формат:

[<метка>:] <код операции > [<список операндов >1 [<комментарии>].

Запись программы выполняется по свободному формату, т. е. специально не оговариваются правила заполнения каких бы то ни было позиций строки. Точка с запятой в начале строки означает, что данная строка является строкой комментария.

Программа может записываться как заглавными, так и строчными буквами. Метку произвольной длины следует записывать с начала строки и отдалять от кода операции двоеточием, за которым может следовать произвольное количество пробелов (вплоть до конца строки).

Код операции должен отделяться от списка операндов хотя бы одним пробелом. Операнды отделяются один от другого запятой.

Определение полей памяти для размещения данных

Для определения данных в основной памяти и резервирования полей памяти под данные, размещаемые в основной памяти в процессе выполнения программы, используются следующие операторы:

DB – определить однобайтовое поле, **DW** – определить слово (двухбайтовое поле), **DD** – определить двойное слово (четырехбайтовое поле).

Формат команды:

	DB	
[<имя поля>]	**DW** [< количество >	**DUP** (]{ <список чисел >}[)]
	DD	?

где <количество >– количество полей памяти указанной длины, которое определяется данной командой (указывается, если определяется не одно поле памяти); ? – используется при резервировании памяти.

Приведем примеры.

1. Записать в байт памяти десятичное число 23 и присвоить этому байту имя a:

a db 23.

2. Зарезервировать 1 байт памяти:**db ?**

3. Записать в слово памяти шестнадцатеричное число 1234:

dw 1234H.

4. Определить 31 байт памяти, повторяя последовательность 1, 2, 3, 4, 5, 1, 2, 3, 4,...:

db 31 dup (1,2,3,4,5)

Примечание. При записи слов в память младший байт записывается в поле с младшим адресом. Например, в примере 3, если запись выполнялась по адресу 100, то по адресу 100 будет записано 34H, а по адресу 101 – 12H.

Операнды команд ассемблера

Операнды команд ассемблера могут определяться непосредственно в команде, находиться в регистрах или в основной памяти.

Данные, непосредственно записанные в команде, называются литералами. Так, в команде

mov ah, 33 – литерал.

Если операнды команд ассемблера находятся в регистрах, то в соответствующих командах указываются имена регистров (если используемые регистры особо не оговариваются для данной команды). Например, в приведенном выше примере **ah** – имя регистра аккумулятора.

Адресация операндов, расположенных в основной памяти, может быть прямой и косвенной.

При использовании прямой адресации в команде указывается символическое имя поля памяти, содержащего необходимые данные, например:

inc OPND

Здесь **OPND** – символическое имя поля памяти, определенного оператором ассемблера

OPND dw ?

При трансляции программы ассемблер заменит символическое имя на исполнительный адрес указанного поля памяти (смещение относительно начала сегмента) и занесет этот адрес на место индексного смещения. Адресация в этом случае выполняется по схеме: **BP** + <индексное смещение>, но содержимое регистра **BP** при вычислении исполнительного адреса не используется (частный случай).

В отличие от прямого косвенный адрес определяет не местоположение данных в основной памяти, а местоположение компонентов адреса этих данных. В этом случае в команде указываются один или два регистра в соответствии с допустимыми схемами адресации и индексное смещение, которое может задаваться числом или символическим именем. Косвенный адрес заключается в квадратные скобки весь или частично, например:

[OPND +SI]
OPND [SI]
OPND + [SI]
[OPND] +[SI]

Приведенные выше формы записи косвенного адреса интерпретируются одинаково.

При трансляции программы ассемблер определяет используемую схему адресации и соответствующим образом формирует машинную команду, при этом символическое имя заменяется

смещением относительно начала сегмента так же, как в случае прямой адресации.

При использовании косвенной адресации по схеме BP + <индексное смещение> индексное смещение не может быть опущено, так как частный случай адресации по данной схеме с нулевой длиной индексного смещения используется для организации прямой адресации. Следовательно, при отсутствии индексного смещения в команде следует указывать нулевое индексное смещение, т.е. [**BP** + **0**] .

Приведем два примера: [**a** + **bx**] и [**bp**]+[**si**] +**6**.

В первом случае исполнительный адрес операнда определяется суммой содержимого регистра **bx** и индексного смещения, заданного символическим именем «**a**», а во втором – суммой содержимого регистров **bp**, **si** и индексного смещения, равного 6.

Длина операнда может определяться:

а) кодом команды – в том случае, если используемая команда обрабатывает данные определенной длины, что специально оговаривается;

б) объемом регистров, используемых для хранения операндов (1 или 2 байта);

в) специальными указателями **byte ptr** (1 байт) и **word ptr** (2 байта), которые используются в тех случаях, когда длину операнда нельзя установить другим способом. Например,

mov byte ptr x, 255

т.е. операнд пересылается в поле с именем «**x**» и имеет длину I байт.

Команды пересылки / преобразования данных

Команда пересылки данных

MOV <адрес приемника>,< адрес источника>

используется для пересылки данных длиной I или 2 байта из регистра в регистр, из регистра в основную память, из основной памяти в регистр, а также для записи в регистр или основную память данных, непосредственно записанных в команде. Все возможные пересылки представлены на рис. 5.6.

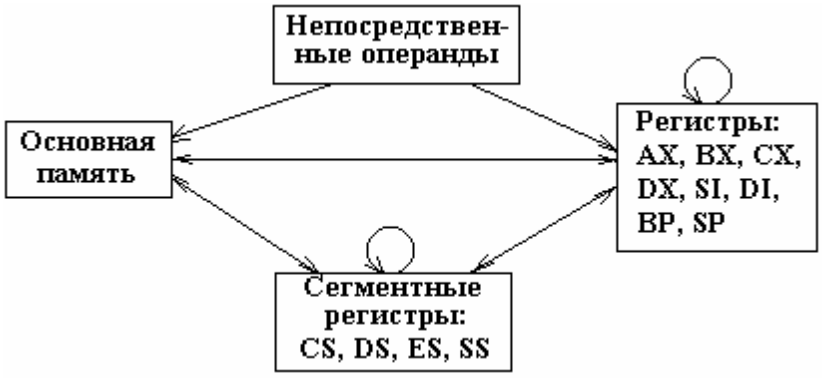


Рис. 5.6 – Команды пересылки

Примеры:

а) **mov ax, bx**– пересылка содержимого регистра **bx** в регистр **ax**;

б) **mov cx, exword** – пересылка 2 байт, расположенных в поле **exword**, из основной памяти в регистр **cx**;

в) **mov si, 1000** – запись числа 1000 в регистр **si**;

г) **mov word ptr [di+515], 4** – запись числа 4 длиной 2 байта в основную память по адресу **[di+515]**.

Для загрузки «прямого» адреса в сегментный регистр используются две команды пересылки:

mov ax, code

mov ds, ax

Команда обмена данных.

XCHG <операнд 1>, <операнд 2>

организует обмен содержимого двух регистров (кроме сегментных) или регистра и поля основной памяти. Например:

xchg bx, cx – обмен содержимого регистров **bx** и **cx**.

Команда загрузки исполнительного адреса

LEA < операнд l >, < операнд 2 >

вычисляет исполнительный адрес второго операнда и помещает его в поле, на которое указывает первый операнд. Приведем примеры:

а) **lea bx, exword** – в регистр **bx** загружается исполнительный адрес **exword**;

б) **lea bx, [di+10]** – в регистр **bx** загружается адрес l0-го байта относительно точки, на которую указывает адрес в регистре **di**.

Команды загрузки указателя

LDS < регистр >,<операнд 2>

LES < регистр >,< операнд 2>

Команда **LDS** загружает в регистры **DS**:< регистр> указатель (< адрес сегмента >: < исполнительный адрес >), расположенный по адресу, указанному во втором операнде.

Команда **LES** загружает указатель по адресу, расположенному во втором операнде, в регистры **ES**:< регистр>.

Например:

lds si, **exword**

т.е. слово (2 байта) по адресу **exword** загружается в **si**, а по адресу **exword+ 2 –** в **ds**.

Команда записи в стек

PUSH < операнд>

организует запись в стек слова, адрес которого указан в операнде. Например;

push dx – запомнить содержимое регистра **dx** в стеке.

Команда восстановления из стека

POP < операнд>

организует чтение из стека последнего слова и помещает его по адресу, указанному во втором операнде. Например:

pop dx – восстановить содержимое регистра **dx** из стека.

Команды сложения

ADD <операнд 1>, <операнд 2>

ADC <операнд 1>, <операнд 2>

устанавливают флаги четности, знака результата, наличия переноса, наличия переполнения.

По команде **ADD** выполняется сложение двух операндов. Результат записывается по адресу первого операнда. По команде **ADC** также выполняется сложение двух операндов, но к ним добавляется еще значение, записанное в бите переноса, установленном предыдущей командой сложения.

На рис. 5.7 показаны возможные способы размещения слагаемых, где а – операнды-слова, б – операнды-байты.

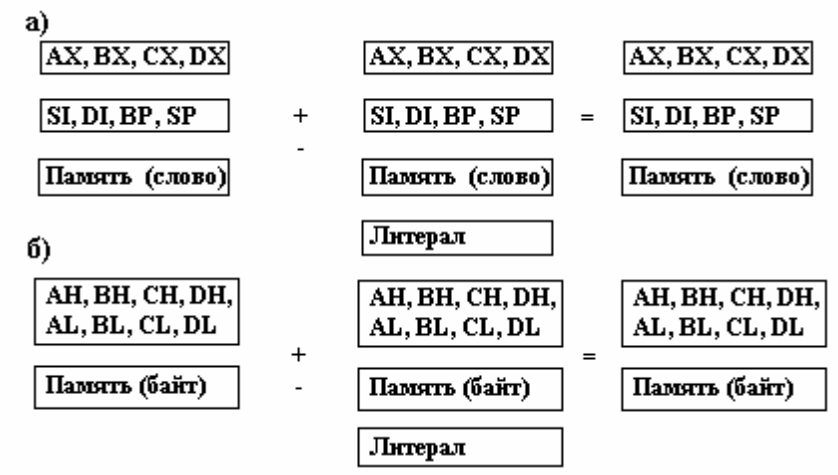


Рис. 5.7 – Способы размещения слагаемых

Приведем пример сложения двух 32-разрядных чисел:

mov ax,value1
add value2,ax
mov ax,value1+2
adc value2+2,ax

Исходные числа находится в основной памяти по адресам **value1** и **value2**, а результат записывается по адресу **value1**.

Команды вычитания

SUB <уменьшаемое-результат>, <вычитаемое>

SBB <уменьшаемое-результат>, <вычитаемое>

устанавливают флаги четности, знака результата, наличия заема, наличия переполнения.

При выполнении операции по команде **SUB** заем не учитывается, а по команде **SBB** – учитывается. Ограничения на местоположение операндов такие же, как и у команды сложения.

Команда изменения знака

NEG <операнд>

знак операнда изменяется на противоположный.

Команда добавления единицы.

INC <операнд>

значение операнда увеличивается на единицу.

Команда вычитания единицы

DEC <операнд>

значение операнда уменьшается на единицу.

Команда сравнения

CMP <операнд 1>, < операнд 2>

выполняется операция вычитания без записи результата и устанавливаются признаки во флажковом регистре.

Команды умножения

MUL <операнд>

IMUL <операнд>

устанавливают флаги наличия переноса или переполнения.

По команде **MUL** числа перемножаются без учета, и по команде – **IMUL** с учетом знака (в дополнительном коде).

На рис. 5.8 (где а – операнды-слова, б – операнды-байты) приведены возможные способы размещения сомножителей и результата (один из сомножителей всегда расположен в регистре-аккумуляторе.

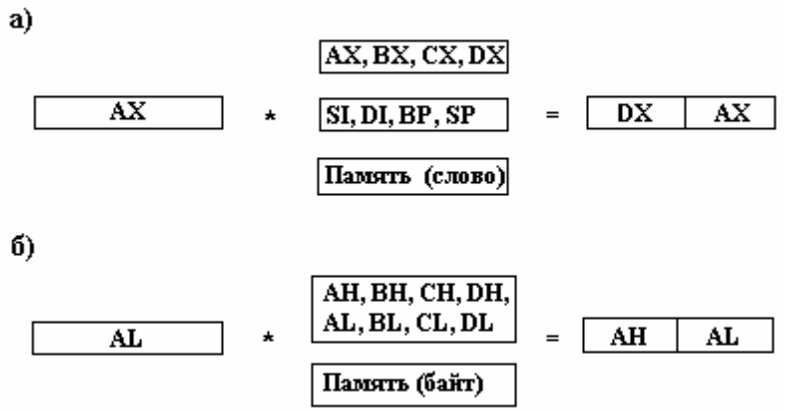


Рис. 5.8 – Способы размещения сомножителей

Пример:

imul word ptr c

Здесь содержимое основной памяти по адресу «**c**» длиной слово умножается на содержимое регистра **ax**. Младшая часть результата операции записывается в регистр **ax**, а старшая часть – в регистр **dx**.

Команда деления

DIV <операнд-делитель>

IDIV <операнд-делитель>

По команде **DIV** операция деления выполняется без учета, а по команде **IDIV** – с учетом знака (в дополнительном коде).

Команды передачи управления

Команда безусловного перехода

JMP <адрес перехода>

имеет три модификации в зависимости от длины ее адресной части:

short – при переходе по адресу, который находится на расстоянии –128...127 байт относительно адреса данной команды (длина адресной части 1 байт);

near ptr – при переходе по адресу, который находится на расстоянии 32 Кбайта (–32768...32767 байт) относительно адреса данной команды (длина адресной части 2байта);

far ptr – при переходе по адресу, который находится на расстоянии, превышающем 32 Кбайта (длина адресной части 4 байта).

При указании перехода к командам, предшествующим команде перехода, ассемблер сам определяет расстояние до метки перехода и строит адрес нужной длины. При указании перехода к последующим частям программы необходимо ставить указатели **short**, **near ptr** и **far ptr**.

В качестве адреса команды перехода используются метки трех видов:

а) < имя >: **nop**(**nop** – команда «нет операции»);

б)< имя> **label near** (для внутрисегментных переходов);

в) <имя> **label far** (для внесегментных переходов).

Примеры:

а) **jmp short b** – переход по адресу **b**;

б) **jmp [bx]** – переход по адресу в регистре **bx** (адрес определяется косвенно);

в) **a: nop –** описание метки перехода «**a»**;

г) **b label near** – описание метки перехода «**b**».

Команды условного перехода

<мнемоническая команда> <адрес перехода>

Мнемоника команд условного перехода:

JZ – переход по «ноль»;

JE – переход по «равно»;

JNZ – переход по «не нуль»;

JNE – переход по «не равно»;

JL – переход по «меньше»;

JNG, JLE – переход по «меньше или равно»;
JG – переход по «больше»;
JNL, JGE – переход по «больше или равно»;
JA – переход по «выше» (беззнаковое больше);
JNA, JBE – переход по «не выше»(беззнаковое не больше);
JB – переход по «ниже» (беззнаковое меньше);
JNB, JAE – переход по «не ниже» (беззнаковое не меньше).

Все команды имеют однобайтовое поле адреса, следовательно, смешение не должно превышать –128...127 байт. Если смещение выходит за указанные пределы, то используется специальный прием:

вместо	программируется
jz zero	**jnz continue** **jmp zero** **continue: ...**

Команды организации циклической обработки

В качестве счетчика цикла во всех командах циклической обработки используется содержимое регистра **cx**.

1) Команда организации цикла.

LOOP < адрес перехода >

при каждом выполнении уменьшает содержимое регистра **cx** на единицу и передает управление по указанному адресу, если **cx** не равно 0:

```
        mov   cx, loop_count        ; загрузка счетчика
begin_loop:
      ; ... тело цикла ...
            loop  begin_loop
```

2) Команда перехода по обнуленному счетчику.

JCXZ <адрес перехода>

передает управление по указанному адресу, если содержимое регистра **cx** равно 0. Например:

```
        mov   cx, loop_count        ; загрузка счетчика
        jcxz  end_of_loop           ; проверка счетчика
begin_loop:
   ; ... тело цикла ...
            loop  begin_loop
end_of_loop:  ...
```

3) Команды организации цикла с условием.

LOOPE <адрес перехода>

LOOPNE <адрес перехода>

уменьшают содержимое на единицу и передают управление по указанному адресу при условии, что содержимое **cx** отлично от нуля, но **LOOPE** дополнительно требует наличия признака «равно», а **LOOPNE** – «не равно», формируемых командами сравнения. Например:

```
            mov    cx,  loop_count          ;  загрузка счетчика
            jcxz    end_of_loop             ;  проверка счетчика
begin_loop:
   ;  ...  тело цикла ...
             cmp     al, 100                ; проверка содержимого al
             loopne  begin_loop   ; возврат в цикл, если cx≠0 и al≠100
end_of_loop:     ...
```

Команды вызова подпрограмм

1) Команда вызова процедуры.

CALL <адрес процедуры>

осуществляет передачу управления по указанному адресу, предварительно записав в стек адрес возврата.

При указании адреса процедуры, так же как и при указании адреса перехода, в командах безусловного перехода возникает необходимость определить удаленности процедуры от места вызова:

а) если процедура удалена не более чем на –128...127 байт, то специальных указаний не требуется;

б) если процедура удалена в пределах 32 кбайт, то перед адресом процедуры необходимо указать **near ptr**,

в) если процедура подпрограмма удалена более, чем на 32 кбайта, то перед адресом процедуры необходимо записать **far ptr**.

Пример:

call near ptr p – вызов подпрограммы "**p**".

Текст процедуры должен быть оформлен в виде:

< имя процедуры> **proc** < указатель удаленности>

... тело процедуры ...

<имя процедуры> **end**

Здесь указатель удаленности также служит дли определения длины адресов, используемых при обращении к процедуре: **near** –

при использовании двухбайтовых адресов, **far** – при использовании четырехбайтовых адресов.

2) Команда возврата управления.

RET [<число>]

извлекает из стека адрес возврата и передает управление по указанному адресу.

Если в команде указано значение счетчика, то после восстановления адреса возврата указанное число добавляется к содержимому регистра-указателя стека. Последний вариант команды позволяет удалить из стека параметры, передаваемые в процедуру через стек.

Значение «флажка нуля» равно нулю.

Команды манипулирования битами

Логические команды

NOT<операнд**>** – логическое НЕ;

AND <операнд 1>, <операнд 2> – логическое И;

OR<операнд 1>, <операнд 2> – логическое ИЛИ;

XOR <операнд 1>, <операнд 2> – исключающее ИЛИ;

TEST <операнд 1>, <операнд 2> – И без записи результата.

Операнды байты или слова.

Пример. Выделить из числа в AL первый бит:

```
and     al,  10000000B
```

Команды сдвига

<код операции> <операнд>, <счетчик>

Счетчик записывается в регистр CL. Если счетчик равен 1, то его можно записать в команду.

Коды команд сдвига:

SAL – сдвиг влево арифметический;

SHL – сдвиг влево логический;

SAR – сдвиг вправо арифметический;

SHR – сдвиг вправо логический;

ROL – сдвиг влево циклический;

ROR – сдвиг вправо циклический;

RCL – сдвиг циклический влево с флагом переноса;

RCR – сдвиг циклический вправо с флагом переноса.

Пример. Умножить число в AX на 10:

```
mov     bx, ax
```

```
shl     ax,  1
shl     ax,  1
add     ax, bx
shl     ax,  1
```

Команды ввода-вывода

Обмен данными с внешней средой осуществляется с помощью следующих команд:

IN <регистр>, <порт> (ввод из порта в регистр),

IN <регистр >, **DX** (ввод из порта, номер которого указан в регистре **DX** в регистр),

OUT <порт>, <регистр> (вывод содержимого регистра в порт),

OUT DX, <регистр> (вывод содержимого регистра в порт, номер которого указан в регистре **DX**).

В качестве регистра можно указать **AL** или **AX** (соответственно будет обрабатываться байт или два байта). Порт отождествляется с некоторым внешним устройством (0...255).

Однако при организации ввода-вывода помимо самой операции необходимо осуществить ряд дополнительных действий, например проверить готовность устройства. В связи с этим для типовых устройств разработаны стандартные программы организации ввода-вывода, которые вызываются по команде прерывания **int 21h**.

В таблице 1 приведен перечень основных функций, реализуемых подпрограммами ввода-вывода, и их кодов. Код функции должен передаваться в подпрограмму в регистре **AH**.

Таблица 1

Код функции	Функция
01	Ввод с клавиатуры одного символа в регистр AL (с проверкой на Ctrl-Break, с ожиданием, с эхо)
02	Вывод одного символа на экран дисплея из регистра DL (с проверкой на Ctrl-Break)
06	Непосредственный ввод-вывод: ввод в регистр AL (без ожидания, без эхо, без проверки на Ctrl-Break, регистр DL должен содержать 0FFH), вывод из регистра DL (без проверки на Ctrl-Break)

Окончание табл. 1

Код функции	Функция
07	Ввод в регистр AL (без проверки на Ctrl-Break, с ожиданием, без эхо)
08	Ввод в регистр AL (с проверкой на Ctrl-Break, с ожиданием, без эхо)
09	Вывод строки на экран (DS:DX – адрес строки, которая должна завершаться символом «$»)
10(0Ah)	Ввод строки в буфер (DS:DX – адрес буфера, первый байт которого должен содержать размер буфера, после ввода – второй байт содержит количество введенных символов)
11(0Bh)	Чтение состояния клавиатуры (если буфер пуст, то AL=0, иначе AL=0FFh)

Примеры:

```
а)      mov     ah,  1  ;  номер функции
        int     21h     ;   ввод символа: символ в AL

б)      mov     ah,  2  ;  номер функции
        mov     dl,  'A'
        int     21h     ;   вывод символа из DL

в)      lea     dx,  STRING ;  адрес буфера ввода
        mov     ah,  0Ah  ;  номер функции
        int     21h   ;  ввод строки:  во втором байте буфера – количество
        ...           ;  введенных символов, далее в буфере символы
STRING  db    50, 50 dup (?)

г)      lea     dx,  MSG ;  адрес выводимой строки
        mov     ah,  9  ;  номер функции
        int     21h     ;  вывод строки
        ...
MSG     db     'Пример вывода', 13, 10, '$'
```

Структура программы на ассемблере

Структура программы на языке ассемблера выглядит следующим образом (**.exe**):

```
                        TITLE  <имя программы>
<имя сегмента стека>     SEGMENT  STACK
                          DB   3000  DUB   (?)
<имя сегмента стека>     ENDS
<имя сегмента данных > SEGMENT
```

```
                <данные>
<имя сегмента данных > ENDS
<имя сегмента кодов>    SEGMENT
                ASSUME CS: <имя сегмента кодов>,  DS:<имя сегмента данных>
                EXTRN  <имя внешней процедуры >:<тип>
                PUBLIC <имя внутренней процедуры>
<имя основной процедуры> PROC FAR
                   PUSH  DS
                   MOV   AX, 0
                   PUSH  AX
                   MOV   AX, <имя сегмента данных>
                   MOV   DS, AX
                <тело процедуры>
                   RET
<имя основной процедуры> ENDP
<имя внутренней процедуры> PROC    NEAR
                <тело внутренней процедуры>
<имя внутренней процедуры> ENDP
<имя сегмента кодов>        ENDS
                   END     <имя основной процедуры >
```

Первая строка программы – заголовок, состоящий из служебного слова **TITLE** и имени программы.

Текст программы состоит из отдельных сегментов, каждый из которых начинается оператором **SEGMENT** и завершается оператором **ENDS**:

```
<имя сегмента > SEGMENT
        ... тело сегмента ...
<имя сегмента > ENDS
```

Сегмент стека содержит специальный описатель **STACK.**

Сегмент кодов, в котором располагается текст программы, начинается псевдокомандой **ASSUME**, которая сообщает ассемблеру, какой сегментный регистр должен использоваться для адресации каждого сегмента.

За псевдокомандой **ASSUME** может следовать описание используемых программой внешних подпрограмм:

EXTRN <имя внешней процедуры >:<тип **near** или **far**>
PUBLIC <имя внутренней процедуры>

Сегмент кодов всегда адресуется сегментным регистром **CS.** Значение этого регистра операционная система устанавливает автоматически. Значения сегментного регистра **DS** загружается программистом:

MOV AX, <имя сегмента данных>
MOV DS, AX

При необходимости также загружается регистр **ES**:

MOV ES, AX

Команды

PUSH DS
MOV AX, 0
PUSH AX

организуют возможность возврата управления в MS DOS командой **RET.**

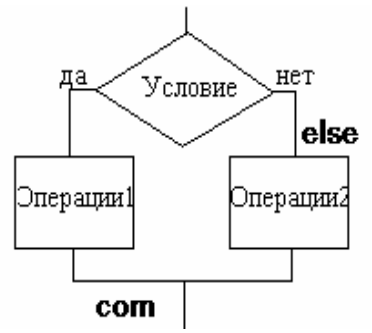
В этом случае в стек в качестве адреса возврата помещается адрес префиксной области программы, первые два байта которой содержат команду **INT 20H** возврата управления операционной системе.

Основные приемы программирования на ассемблере

Ассемблер, являясь языком низкого уровня, не содержит операторов ветвления, циклов, не поддерживает автоматического формирования адресов для структур данных, не обеспечивает автоматического выполнения преобразований при вводе-выводе данных. Все перечисленные операции программируются «вручную» с использованием имеющихся команд ассемблера.

Программирование ветвлений

Ветвления программируются с использованием команд условной и безусловной передачи управления.



cmp ...
j<условие>**ELSE**
<операции 1>
jmpCOM
ELSE:<операции 2>
COM: <продолжение>

Пример.
Написать процедуру вычисления **X=max(A,B)**:

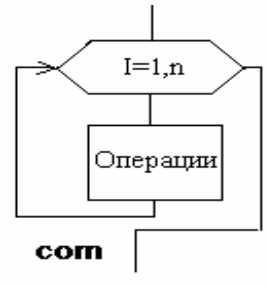
```
max      proc     near
         mov      ax, A
         cmp      ax, B      ; сравнение A и B
         jl       LESS       ; переход по меньше
         mov      X, ax
         jmp      CONTINUE   ; переход на конец ветвления
LESS:    mov      ax, B
         mov      X, ax
CONTINUE: ret
max      endp
```

Программирование циклических процессов

Программирование циклических процессов осуществляется с использованием либо команд переходов, либо – в случае счетных циклов – с использованием команд организации циклов.

а) программирование итерационных циклов (цикл-пока):


```
CYCL:    cmp  ...
         jne  COM
         <операции>
         jmp  CYCL
COM:     ....
:
```




```
         mov  cx, N
CYCL:    <операции>
         loop CYCL
com:     .....
```

Процедура суммирования чисел от 1 до 10, используя счетный цикл.

```
sum     proc    near
        mov     ax, 0    ;  обнуление суммы
        mov     bx, 1    ;  первое слагаемое
        mov     cx, 10   ;  загрузка счетчика
CYCL:   add     ax, bx   ;  суммирование
        inc     bx       ;  следующее число
        loop    CYCL     ;  возврат в цикл
continue: ret            ; выход,  сумма - в ax
sum     endp
```

6 РАБОЧИЕ ТЕТРАДИ

6.1 Глава 1. Обобщенная структурная схема ЭВМ и характеристики

Электронная (цифровая) вычислительная машина или система это комплекс оборудования, выполняющий интерпретацию программы (алгоритма) в виде физических процессов, назначением которых является реализация математических операций над информацией, представляемой в цифровой форме.

Электронно-вычислительная система – это совокупность элементов, объединенных для достижения какой-либо цели (как правило, цель есть универсальное преобразование информации).

Признаки системы:

- Система – структура.
- Функции.
- Схема (структурная / функциональная / принципиальная).
- Архитектура.

Обобщенная структура ЭВМ представлена на рис. 6.1.

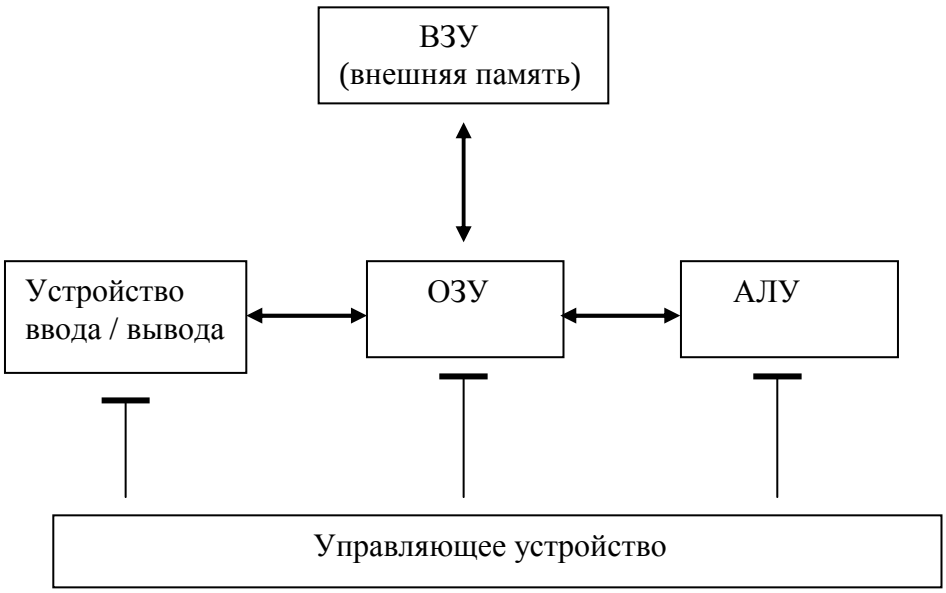


Рис. 1.1 – Структура электронной цифровой вычислительной машины

На рисунке 1.1 жирными стрелками показаны информационные связи, остальные – управляющие.

ЭВМ содержит следующие основные устройства: арифметико-логическое устройство, память, управляющее устройство, устройство ввода данных в компьютер, устройство вывода результатов преобразования информации и управления.

Наиболее важные части: память, и устройство управления.

Арифметико-логическое устройство (АЛУ) производит арифметические и логические преобразования над поступающими в него машинными словами, т.е. кодами определенной длины.

Информационные слова, хранимые в запоминающем устройстве, выбираются из него и передаются в АЛУ, где над ним выполняются операции. Характер выполняемой АЛУ операции задается командой программы.

Память хранит информацию, передаваемую из других устройств, в том числе поступающую в машину извне через устройство ввода, и выдает во все другие устройства информацию, необходимую для протекания вычислительного процесса. Память машины в большинстве случаев состоит из двух существенно отличающихся по своим характеристикам частей: быстродействующей основной или оперативной (внутренней) памяти (ОП) и сравнительно медленно действующей, но способной хранить больший объем информации внешней памяти (ВнП).

Управляющее устройство (УУ) или управляющий автомат автоматически без участия человека управляет вычислительным процессом, посылая всем другим устройствам управляющие сигналы, инициирующие выполнение соответствующей последовательности микроопераций, обеспечивающих реализацию текущей команды. Устройства ввода-вывода и пульт ручного управления образуют группу периферийных устройств ЭВМ.

Устройство ввода используется для считывания программы и исходных данных с перфокарт, перфолент или магнитной ленты и переноса их в ОП. Необходимая для решения задачи информация может вводиться непосредственно с клавиатуры.

Устройство вывода служит для выдачи из машины результатов расчета, например, путем печатания их на печатных устройствах или отображения на экране дисплея.

Принципы функционирования УПИ определяют производительность ЭВМ:

- Принцип фон Неймана, основанный на последовательной обработке информации.
- Оккама (гарвардская модель), основанный на параллельном функционировании (глава 9).
- Нейросетевой подход (глава 10).

Принципы фон Неймана:

- принцип условного перехода, Выполнение команд осуществляется при условии выполнения некоторого условия. Именно благодаря наличию команд условного перехода ЭВМ может автоматически изменять соответствующим образом ход вычислительного процесса, решать сложные логические задачи;
- принцип программного управления, являющийся основной особенностью ЭВМ, посредством которого достигается автоматическое управление процессом решения задачи;
- принцип хранимой в памяти программы, согласно которому команды программы, закодированные в цифровом виде, хранятся в памяти наравне с числами. В команде указываются не сами участвующие в операциях числа, а адреса ячеек ОП, в которых они находятся, и адрес ячейки, куда помещается результат операции;
- принцип двоичной арифметики, смысл которого состоит в простоте использования двоичных слов в физическом смысле;
- принцип иерархической памяти, которая включает пять «ступеней»: регистры, ОЗУ, кэш-память, ВЗУ, АрЗУ (архивные запоминающие устройства).

Фон-неймановская модель вычислительной машины проста и гибка при управлении вычислительным процессом. Недостаток: общая память для данных и команд и всего одна шина (магистраль) для передачи из памяти в другие устройства команд и данных ограничивают скорость работы ЭВМ.

Гарвардская модель

Принцип параллельного вычисления заключается в одновременном выполнении нескольких (многих) процессов обработки информации. Это реализуется за счет возможности построения

машины с отдельной памятью и шинами для хранения и передачи команд и данных, допускающих параллельное во времени извлечение их из памяти и передачу по шинам.

Нейросетевой подход.

Обработка информации подобно человеческому мозгу. В основе объединение простейших элементов – нейронов, связанных сложными связями. Основная процедура – «обучение» нейронов или подбор синоптических коэффициентов в нейронах.

Основные характеристики ЭВМ:

1) Эксплуатационная характеристика ЭВМ ее производительность или быстродействие (млн. опер./ сек.) **Р** и общий коэффициент эффективности машины:

$$\text{Э} = P / (\text{Сэвм} + \text{Сэкс}),$$

представляющий собой отношение производительности к сумме стоимости машины Сэвм и затрат на ее эксплуатацию Сэкс.

2) Производительность как отношение времени работы ко времени простоя:

$$P = \text{Траб}/ \text{Т прост.}$$

3) Системная производительность – количество задач, выполняемое за определенное время:

$$L = n / T.$$

К более частным характеристикам ЭВМ относятся:

- число разрядов в машинном слове (влияет на точность вычислений и диапазон представимых в машине чисел);
- скорость выполнения основных видов команд;
- емкость оперативной памяти;
- максимальная скорость передачи информации между ядром ЭВМ (процессор и оперативная память) и внешним (периферийным) оборудованием.

4) Надежность ЭВМ определяется частотой нарушения ее работоспособности из-за отказов и сбоев, затратами времени на их устранение или, другими словами, безотказностью, достоверностью функционирования и ремонтнопригодностью.

Безотказность может оцениваться средним временем наработки машины на один отказ.

Ремонтнопригодность, определяя потерю работоспособности машины вследствие необходимости производить устранение неисправностей, характеризуется средним временем устранения неисправности.

Достоверность функционирования есть свойство машины, определяемое безошибочностью производимых машиной преобразований информации и характеризуемое закономерностями появления ошибок из-за сбоев. Достоверность функционирования ЭВМ можно оценить средним временем наработки машины на один сбой.

5) Важной характеристикой ЭВМ является универсальность, которая проявляется в возможности решения на одной и той же ЭВМ задач самого различного характера.

6.2 Глава 2. Элементы булевой алгебры

Булева функция – дискретная функция, принимающая два значения, так же как и её аргумент.

Набор булевых функций: $N_н = 2^n$.

Общее количество функций: $N = 2^{2^n}$.

Теоремы булевых функций

1) $A + \overline{A} = 1$, $A + \ldots + A = A$, $A * A * \ldots * A = A$,

2) Теорема Де Моргана:

$\overline{A} + \overline{B} = \overline{A \cdot B}$;
$\overline{A} \cdot \overline{B} = \overline{A + B}$.

} доказывается по диаграммам Венна

3) Теорема склеивания:

$B + 1 = 1$;

$A \cdot (A + B) = A + A \cdot B = A$.

ДНФ – Дизъюнктивная Нормальная Форма (дизъюнкция элементарных конъюнкций).

КНФ – Конъюнктивная Нормальная Форма (конъюнкция элементарных дизъюнкций).

СДНФ – Совершенная Дизъюнктивная Нормальная Форма (ДНФ, в которой все элементарные конъюнкции – полные).

СКНФ – Совершенная Конъюнктивная Нормальная Форма (КНФ, в которой все элементарные дизъюнкции – полные).

Элементарная Конъюнкция – конъюнкция К ≥ 1 переменных или их отрицаний.

Элементарная Дизъюнкция – дизъюнкция К ≥ 1 переменных или их отрицаний.

$F(x_1, x_2,.., x_n)$, элементарную конъюнкцию (дизъюнкцию) формулы F называют **полной** если она содержит все элементы списка ровно по одному разу, в том же порядке, в котором они входят в список.

Свойства булевых функций:

1) функция $f(x_1, x_2,\ldots,x_n)$ сохраняет 0, если $f(0, 0,\ldots,0) = 0$;

2) функция $f(x_1, x_2,\ldots,x_n)$ сохраняет 1, если $f(1, 1,\ldots,1) = 1$;

3) функция $f(x_1, x_2,\ldots,x_n)$ самодвойственна, если $f(x_1, x_2,\ldots,x_n) = f^*(x_1, x_2,\ldots,x_n)$, где f^* – функция, двойственная функции $f(f^*(x_1, x_2,\ldots,x_n) = f(x_1, x_2,\ldots, x_n))$;

т. е. на противоположных значениях аргумента принимает одни и те же значения.

4) функция $f(x_1, x_2,\ldots,x_n)$ линейна, если $f = a_0 \oplus a_0 \oplus a_1 \cdot x_1 \oplus \ldots \oplus a_n \cdot x_n$;

5) функция $f(x_1, x_2,\ldots,x_n)$ монотонна, если $\forall A \subset B \Rightarrow f(A) \le f(B) \cup \forall A \subset B \Rightarrow f(A) \ge f(B)$;

т.е. или монотонно возрастает или монотонно убывает.

Примеры практического определения монотонности булевых функций:

$\overline{1\ 1\ 1\ 0}$ – эта функция немонотонная;

$\overline{1\ 1\ 1\ 1}$ – эта функция монотонная;

$\overline{0\ 0\ 1\ 1}$ – это функция также монотонна.

Примеры определения линейности булевых функций по теореме Жегалкина:

(все операции по модулю два)

$f_8 = 1\ 0\ 0\ 0 = 1 * \overline{(a * b)} + 0 * (\overline{a} * b) + \overline{1} * (a * b) + 1 * (a * b) =$

Из теоремы $a + \overline{a} = 1$ имеем, что $a = a + 1$ и $f_8 = (a + 1)(b + 1)$, тогда $F_8 = 1 + a + b + a * b$ функция нелинейная.

a	0	1	1	1
b	0	0	0	1
f	1	0	0	0

Таблицы истинности для логических элементов:

- Логические элементы без памяти (ЛЭ без памяти): И, ИЛИ, НЕ, И – НЕ, ИЛИ – НЕ.
- Логические элементы с памятью (ЛЭ с П): триггеры.
- Логические узлы без памяти (ЛУ без П): сумматор, мультиплексор, демультиплексор, шифратор, дешифратор.
- Логические узлы с памятью (ЛУ с П): регистр, счётчик.

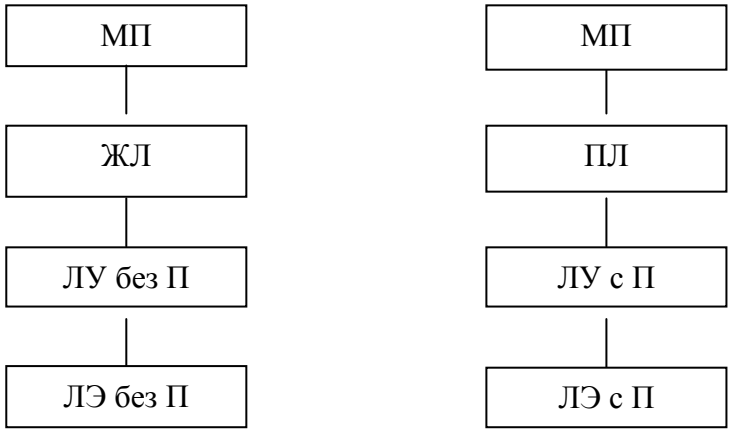


Рис. 2.1 – Иерархия логики:
МП – микропроцессор;
ЖЛ – жёсткая логика;
ПЛ – программируемая логика

Таким образом, перед тем как начать логическое проектирование, необходимо выбрать тип логики (жёсткая или программируемая).

Жесткая логика – быстродействующая, но перепрограммировать её невозможно.

Проектирование логических устройств

Минимизация:

1. Алгебраический метод.
2. Метод Квайна.
3. Метод карт Карно.
4. Метод Вейче.

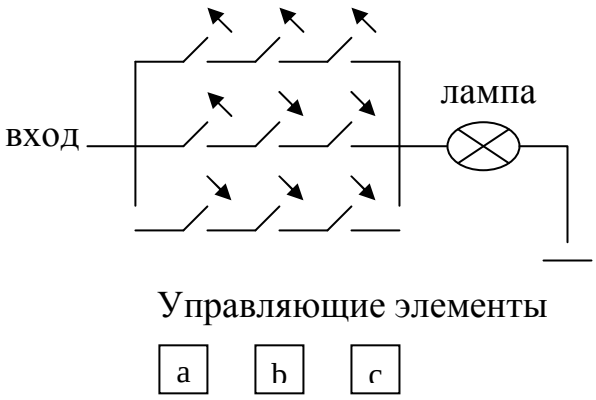
Задача синтеза:

1. Уяснить задачу.
2. Определить число аргументов и число функций.
3. Составить таблицу истинности.
4. Записать булеву функцию.
5. Произвести её минимизацию.
6. Выбрать логический базис.
7. Синтезировать логическую схему.
8. Выполнить её проверку.

Пример: спроектировать устройство (рис. 2.2), определяющее число, которое делится на три.

Таблица истинности такого устройства:

n	a	b	c	вых.
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0



Транзистор – элемент НЕ

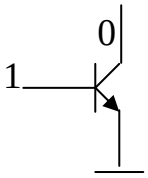
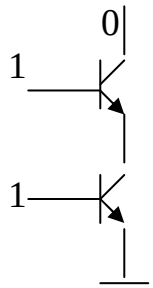
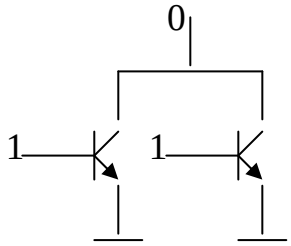


Рис. 2.2 – Электрическая схема устройства

Два транзистора, соединённых последовательно, – элемент И-НЕ.



Два транзистора, соединённых параллельно – элемент ИЛИ-НЕ.



Триггер – конечный автомат Мура, имеет следующие особенности:

1) множество входов;

2) два выхода (прямой и инверсный);

3) состояние конечного автомата (триггера) совпадает с его выходом;

$a(t + 1) = b(x, a)$;

$y = b(a)$;

x – состояние входа;

b – состояние выхода;

a – предыдущее состояние;

Автомат Миля:

$a(t + 1) = b(x, a)$;

$y = b(x, a)$.

Триггеры (RS, D, T, JK) – автоматы Мура.

Асинхронный RS-триггер.

S – SET (установка);
R – RESET (сброс);
Q(t) – состояние триггера

Таблица истинности.

R	S	Q(t)	Q(t+1)	
0	0	0	0	} т. к. на входе ничего нет.
0	0	1	1	
0	1	0	1	} установка единицы.
0	1	1	1	
1	0	0	0	} установка нуля.
1	0	1	0	
1	1	0	-	} не используются.
1	1	1	-	

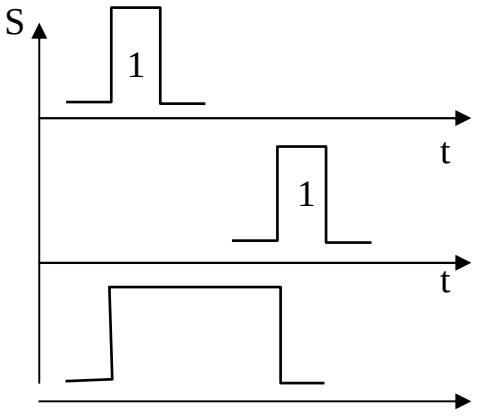
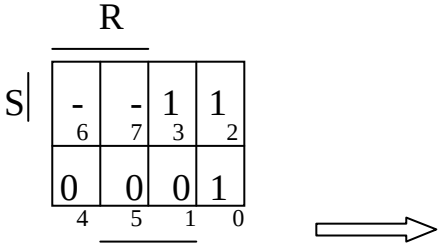


Рис. 2.3 – Эпюры t

Методом карт Карно определяем булеву функцию:



$$Q(t+1) = S + \overline{R} \cdot \overline{Q};$$

Выберем базис на элементах И-НЕ:

$$Q(t+1) = S + \overline{R} \cdot \overline{Q} = \overline{\overline{S + \overline{R} \cdot \overline{Q}}} = \overline{\overline{S} \cdot \overline{\overline{R} \cdot \overline{Q}}};$$

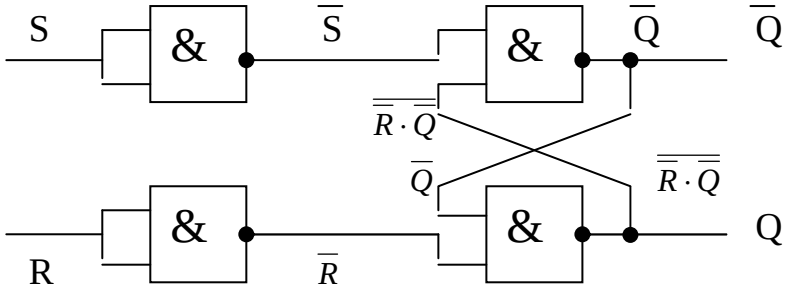


Рис. 2.4 – Схема RS-триггера

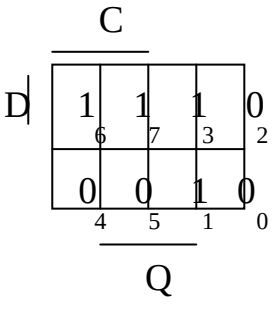
D-триггер (защелка)

Вход C: 0 – сохранить предыдущее состояние;
1 – изменить состояние.
Вход D: 0 – установка нуля;
1 – установка единицы.

Таблица истинности.

C	D	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Методом карт Карно определяем булеву функцию:



$$Q(t+1) = D \cdot C + \overline{C} \cdot Q;$$

Выберем базис на элементах И-НЕ:

$$Q(t+1) = D \cdot C + \overline{C} \cdot Q = \overline{\overline{D \cdot C + \overline{C} \cdot Q}} = \overline{\overline{D \cdot C} \cdot \overline{\overline{C} \cdot Q}};$$

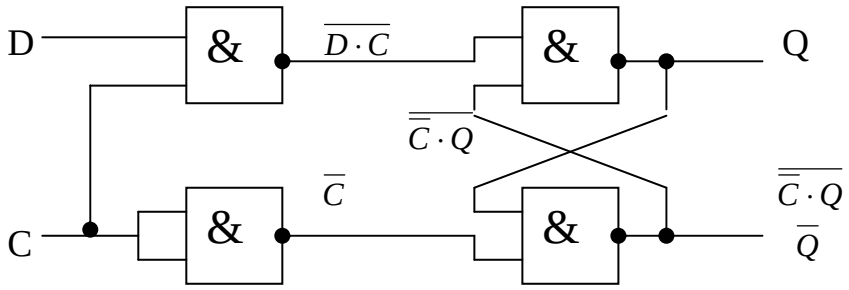


Рис. 2.5 – Схема D-триггера

Тактируемый RS-триггер

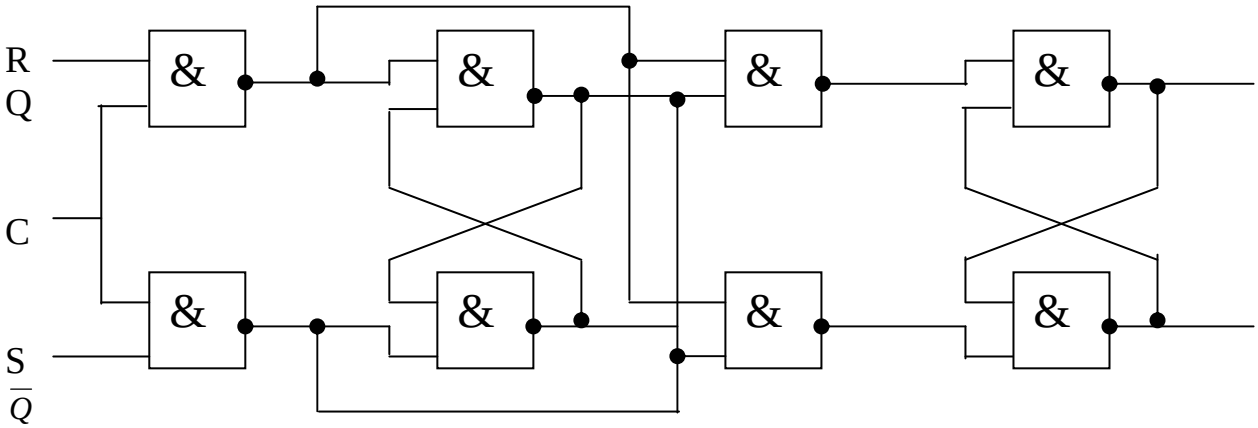


Рис. 2.6 – Двухступенчатый RS-триггер

Срабатывание триггера происходит на заднем фронте.

Т-триггер (счётный)

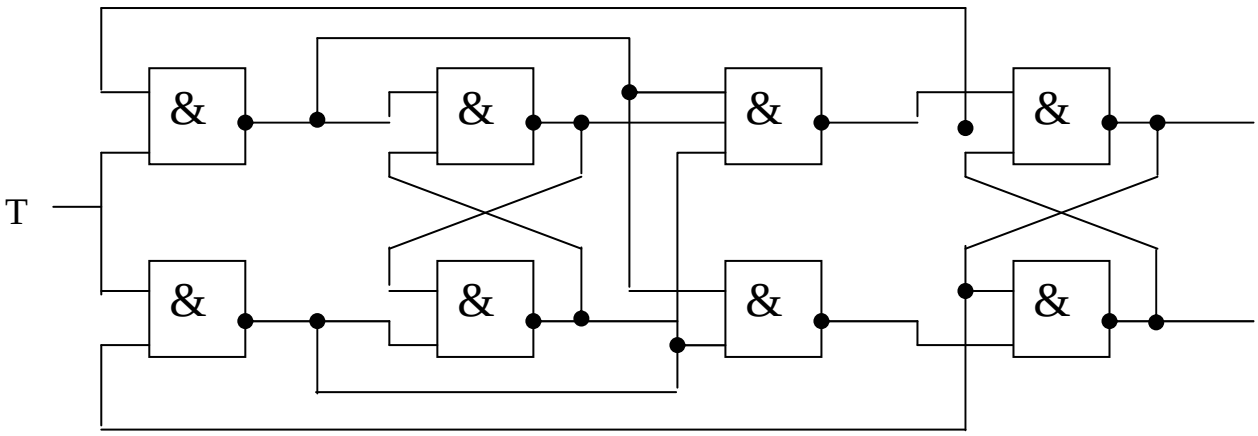


Рис. 2.7 – Т-триггер

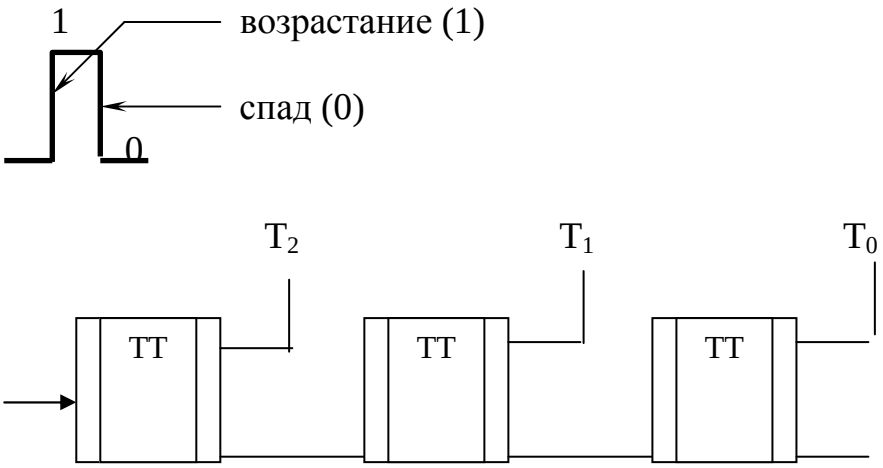


Рис. 2.8 – Схема Т-триггера

ТТ – двухтактный триггер, срабатывает по заднему фронту.

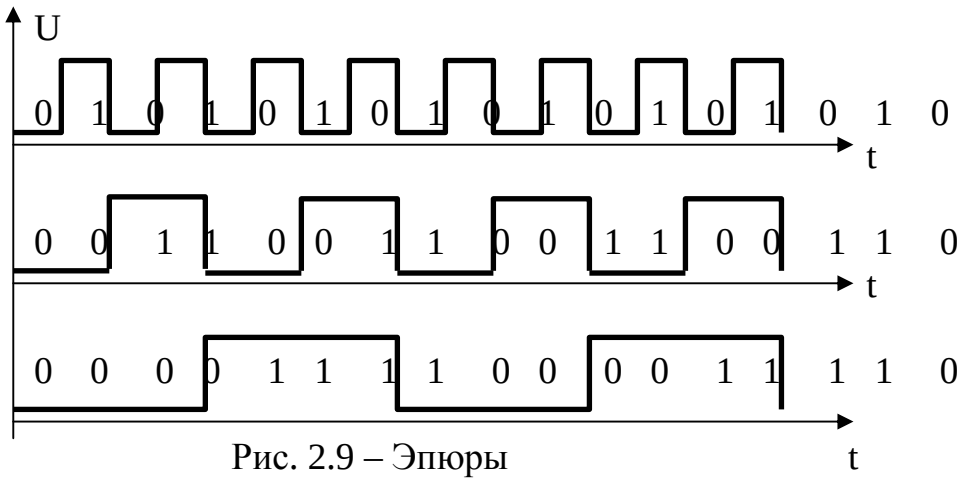


Рис. 2.9 – Эпюры

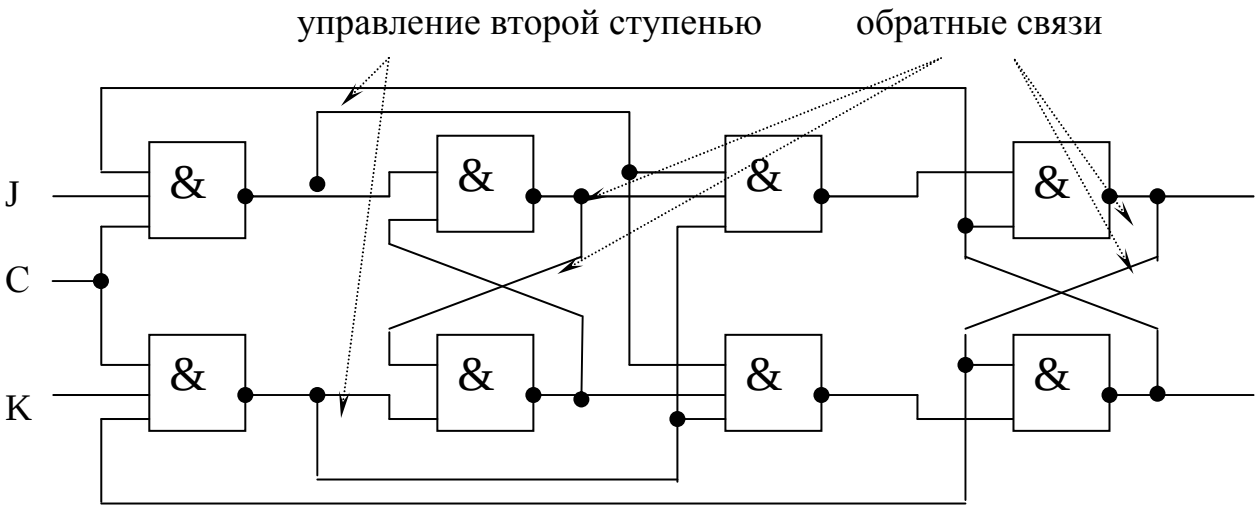


Рис. 2.10 – JK-триггер

JK-триггер = RS-триггер + T-триггер;

C	J	K	Q(t)	Q(t+1)
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Чтобы триггер работал, на вход С необходимо подавать 1, если на С – 0, триггер находится в режиме хранения.

Логические узлы

Логические узлы без памяти:
1. Сумматор.
2. Компаратор.
3. Дешифратор.
4. Шифратор.
5. Мультиплексор.
6. Демультиплексор.

Сумматор

a	b	S	P
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = \overline{\overline{a \cdot \bar{b} + \bar{a} \cdot b}} = \overline{\overline{a \cdot \bar{b}} \cdot \overline{\bar{a} \cdot b}};$$
$$P = a \cdot b;$$

Одноразрядный сумматор

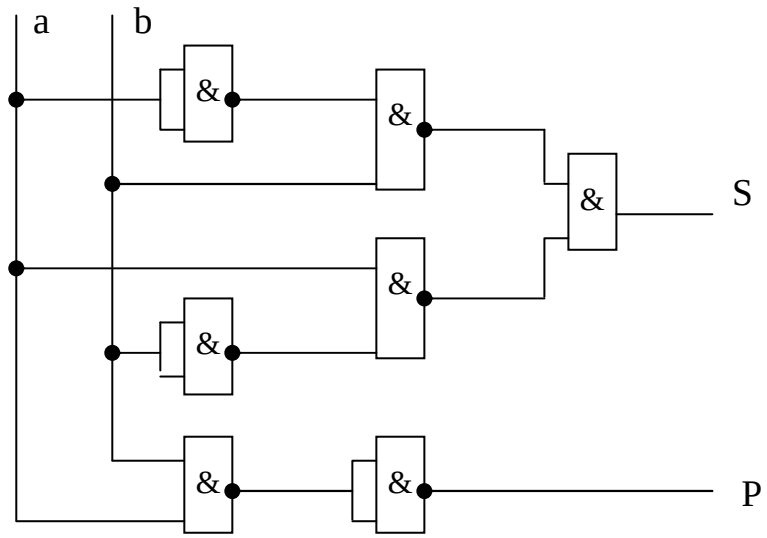


Рис. 2.11 – Сумматор

Трёхразрядный сумматор

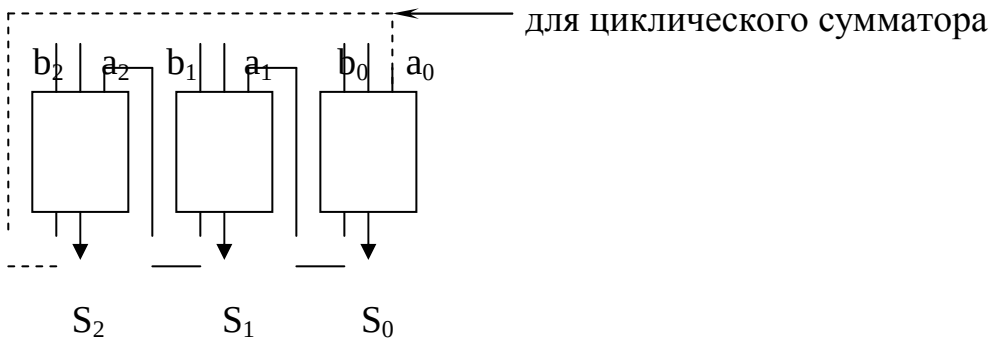


Рис. 2.12 – Трехразрядный сумматор

Компаратор

a_1	b_1	a_1	b_1	f
0	0	0	0	0
0	1	0	1	0
1	0	1	0	1
1	1	1	1	0

Дешифратор

a	b	c	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	1							
0	0	1		1						
0	1	0			1					
0	1	1				1				
1	0	0					1			
1	0	1						1		
1	1	0							1	
1	1	1								1

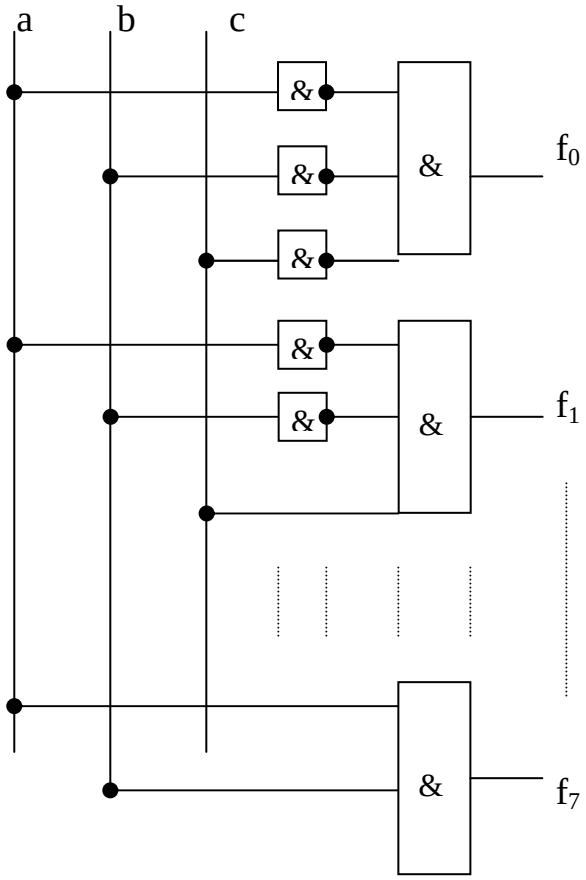
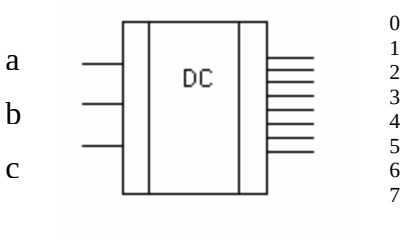


Рис. 2.13 – Дешифратор

Обозначение:



Шифратор

Таблица истинности обратно противоположна таблице истинности дешифратора (см. выше).

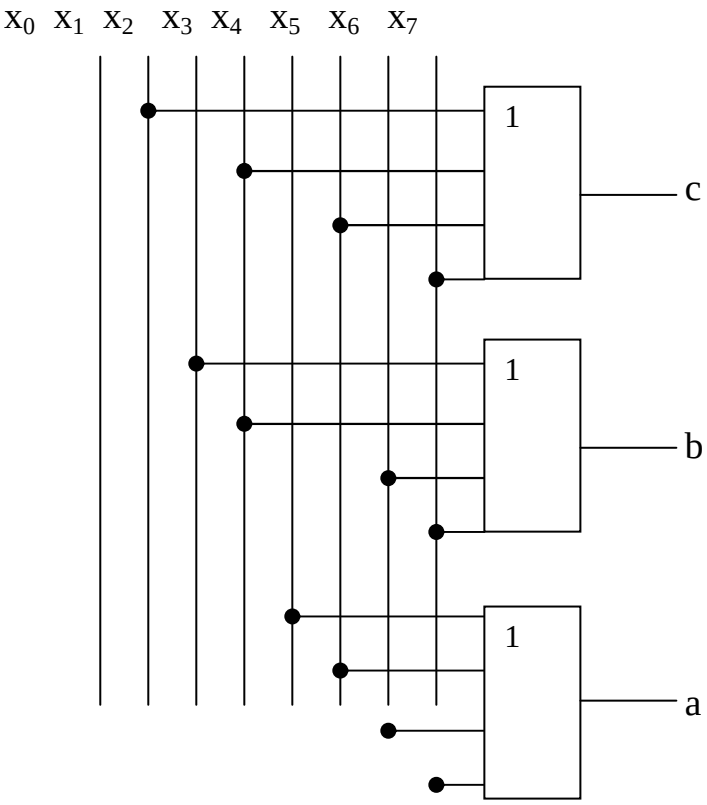
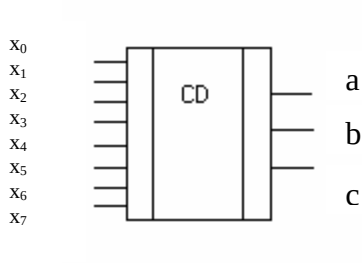
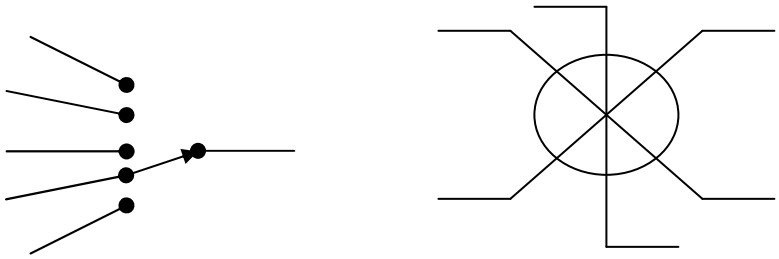


Рис. 2.14 – Шифратор

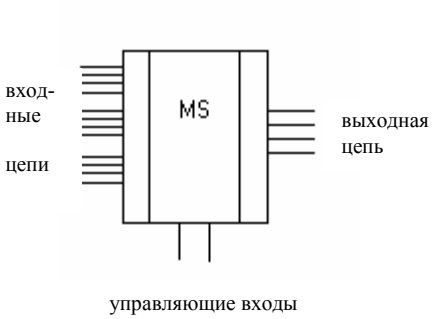
Обозначение:



Мультиплексор (коммутатор) – переключает входные цепи.

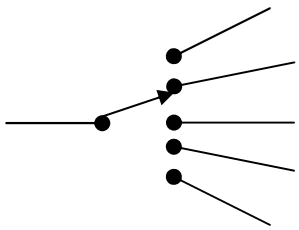


Обозначение:

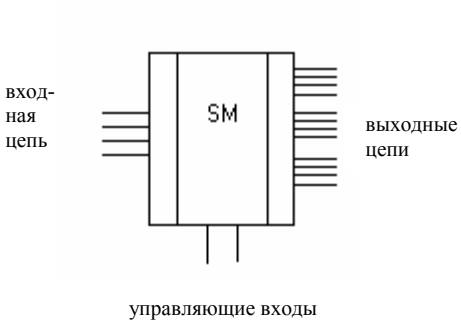


В зависимости от цифровой комбинации, поступающей на управляющие входы, мультиплексор подключает ту или иную входную цепь к выходной цепи.

Демультиплексор – прямо противоположен мультиплексору.



Обозначение:

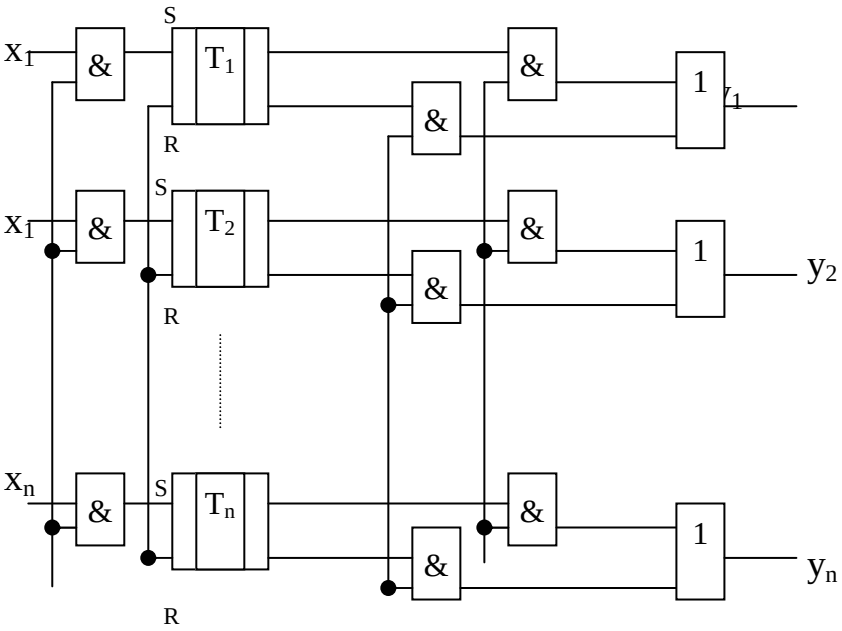


В зависимости от цифровой комбинации, поступающей на управляющие входы, мультиплексор подключает ту или иную выходную цепь к входной цепи.

Узлы с памятью

Регистр:
- параллельный;
- последовательный;
- параллельно-последовательный;
- последовательно-параллельный;
- универсальный.

Параллельный регистр

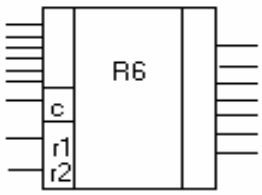


сброс считывание в прямом коде
запись считывание в обратном коде

T_n – однотактный триггер.

Рис. 2.15 – Параллельный регистр

Обозначение:



r1, r2 – управление

Последовательный регистр

сдвиг: 010 → 001

Граф переходов

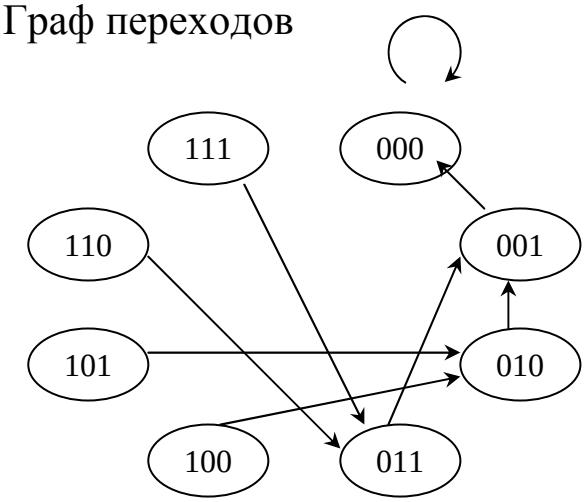
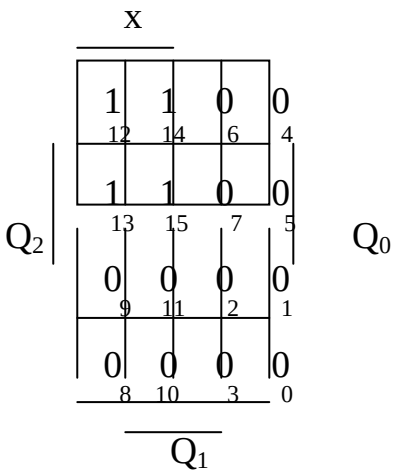


Рис. 2.16 – Граф переходов

Таблица истинности

	x	$Q_2(t)$	$Q_1(t)$	$Q_0(t)$	$Q_2(t+1)$	$Q_1(t+1)$	$Q_0(t+1)$	D_2	D_1	D_0
8	1	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	0	0
10	1	0	1	0	0	0	1	0	0	1
11	1	0	1	1	0	0	1	0	0	1
12	1	1	0	0	0	1	0	0	1	0
13	1	1	0	1	0	1	0	0	1	0
14	1	1	1	0	0	1	1	0	1	1
15	1	1	1	1	0	1	1	0	1	1



$D_2 = 0$

$D_1 = x \cdot Q_2$

$D_0 = x \cdot Q_1$

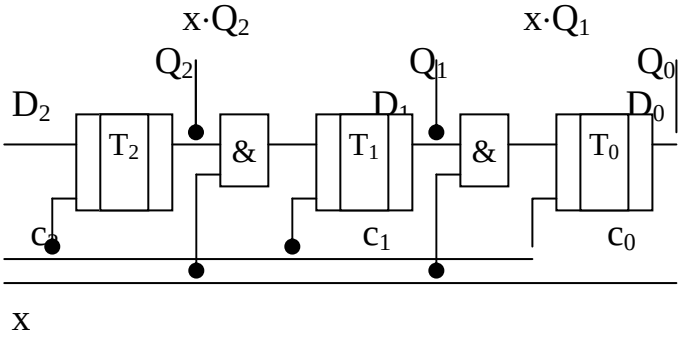
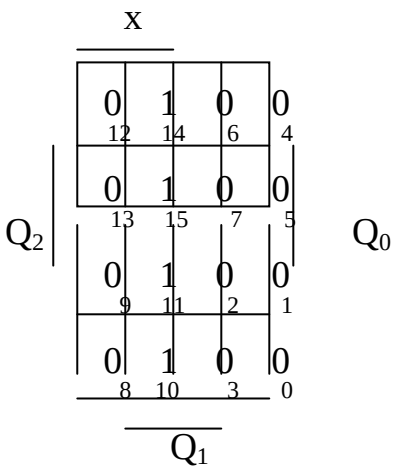


Рис. 2.17 – Регистр сдвига

Проверка



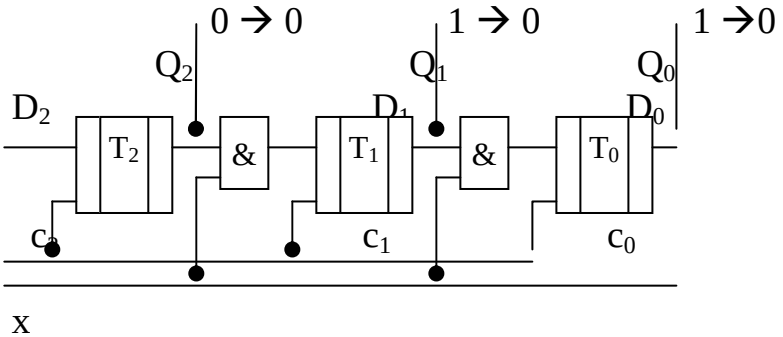


Рис. 2.18 – Регистр сдвига (проверка)

Счётчик предназначен для подсчёта импульсов, поступающих на его входы, результатом является кодовая комбинация, соответствующая числу импульсов.

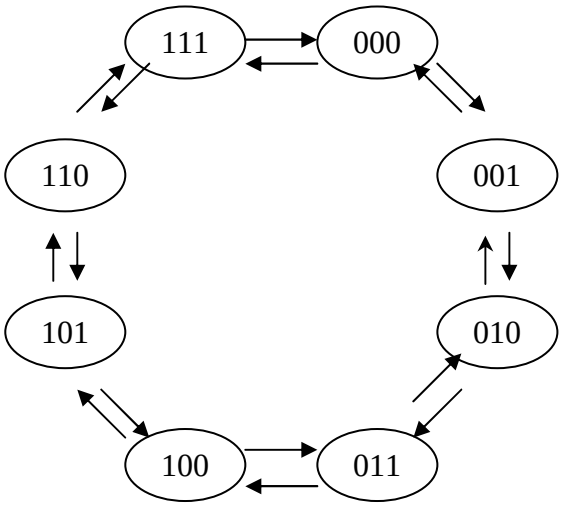
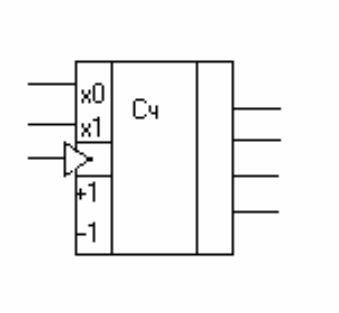


Рис. 2.19 – Граф переходов

При x = 1счётчик считает в прямом направлении, при x = 0 – в обратном.

Обозначение:



Счётчик на Т-триггерах

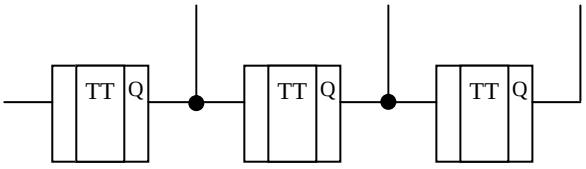


Рис. 2.20 – Счетчик на Т-триггерах

6.3 Глава 3. Двоичная арифметика

Правила выполнения арифметических действий над двоичными числами определяются арифметическими действиями над одноразрядными двоичными числами.

Сложение	Вычитание	Умножение
$0+0=0$	$0-0=0$	$0*0=0$
$0+1=1$	$1-0=0$	$1*0=0$
$1+0=1$	$1-1=0$	$0*1=0$
$1+1=10$	$10-1=1$	$1*1=1$

↑↵
перенос
в старший
разряд

Правила выполнения арифметических действий во всех позиционных системах счисления аналогичны.

Сложение

Как и в десятичной системе счисления, сложение двоичных чисел начинается с правых (младших) разрядов. Если результат сложения цифр МЗР обоих слагаемых не помещается в этом же разряде результата, то происходит перенос. Цифра, переносимая в соседний разряд слева, добавляется к его содержимому. Такая операция выполняется над всеми разрядами слагаемых от МЗР до СЗР.

Вычитание

Операция вычитания двоичных чисел аналогична операции в десятичной системе счисления. Операция вычитания начинается, как и сложение, с МЗР. Если содержимое разряда уменьшаемого меньше содержимого одноименного разряда вычитаемого, то происходит заем 1 из соседнего старшего разряда. Операция повторяется над всеми разрядами операндов от МЗР до СЗР.

Второй вариант операции вычитания – когда уменьшаемое меньше вычитаемого – приведен в разделе представления двоичных чисел в дополнительном коде.

Умножение

Как и в десятичной системе счисления, операция перемножения двоичных многоразрядных чисел производится путем образования частичных произведений и последующего их суммирования. Частичные произведения формируются в результате умножения множимого на каждый разряд множителя, начиная с МЗР. Каждое частичное произведение смещено относительно предыдущего на один разряд. Поскольку умножение идет в двоичной системе счисления, каждое частичное произведение равно либо 0 (если в соответствующем разряде множителя стоит 0), либо является копией множимого, смещенного на соответствующее число разрядов влево (если в разряде множителя стоит 1). Поэтому умножение двоичных чисел идет путем сдвига и сложения. Таким образом, количество частичных произведений определяется количеством единиц в множителе, а их сдвиг – положением единиц (МЗР частичного произведения совпадает с положением соответствующей единице в множителе). Положение точки в дробном числе определяется так же, как и при умножении десятичных чисел.

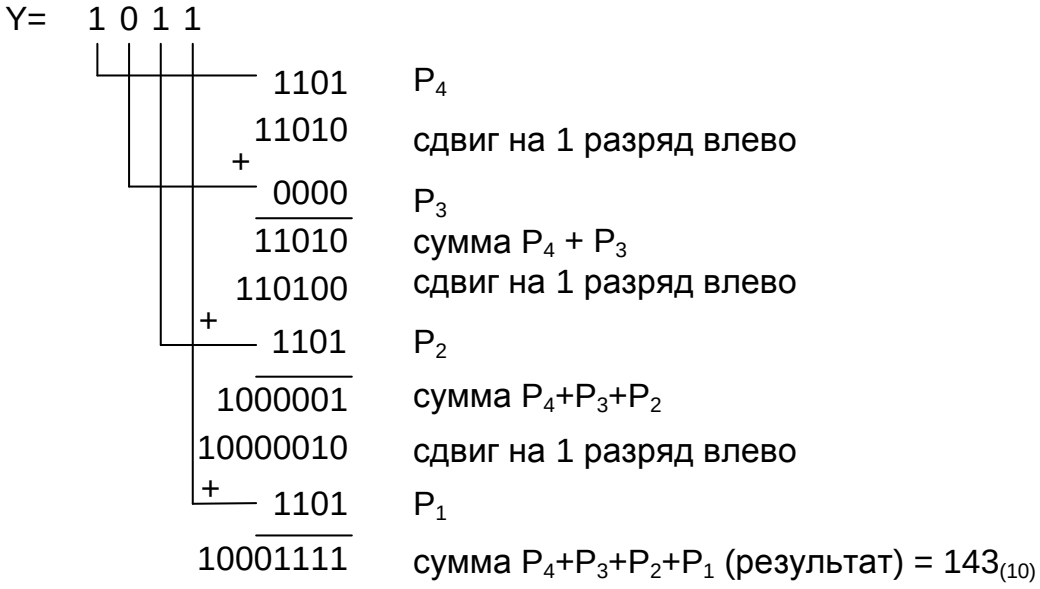
Операция умножения состоит в формировании суммы частичных произведений, которые суммируются с соответствующими сдвигами друг относительно друга. Этот процесс суммирования можно начинать либо с младшего, либо со старшего частичного произведения. В ЭВМ процессу суммирования придают последовательный характер, т.е. формируют одно частичное произведение, к нему с соответствующим сдвигом прибавляют следующее и т.д. (т.е. не формируют все частичные произведения, а потом их складывают). В зависимости от того, с какого частичного произведения начинается суммирование (старшего или младшего), сдвиг текущей суммы осуществляется влево или вправо. При умножении целых чисел для фиксации результата в разрядной сетке число разрядов должно равняться сумме числа разрядов в X и Y.

Пример:

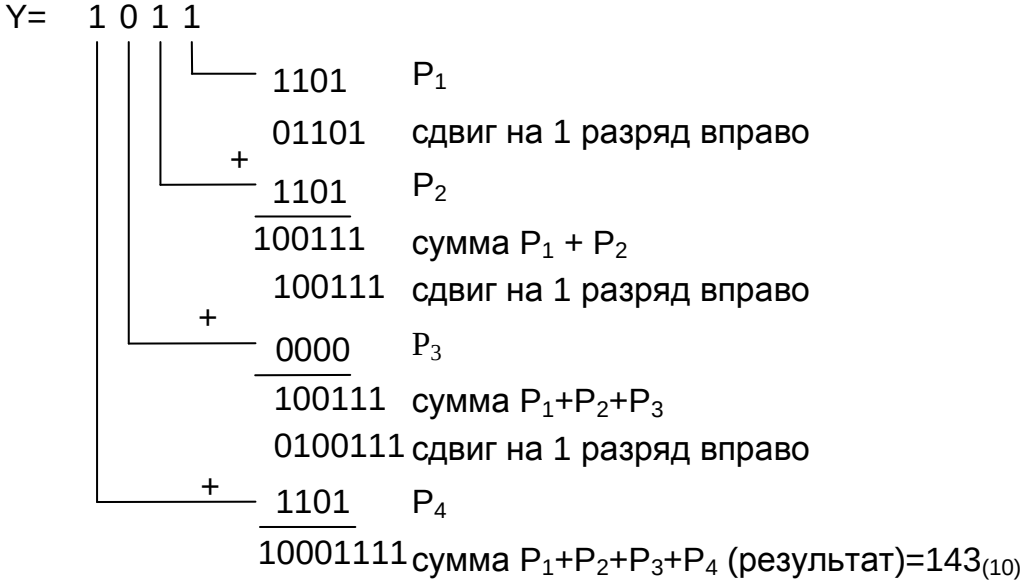
Найдем произведение двух чисел
$X*Y=1101_{(2)}*1011_{(2)}=13_{(10)}*11_{(10)}= 143_{(10)}$.

Обозначим P_i – i-ое частичное произведение.

1. Умножение старшими разрядами вперед.



2. Умножение младшими разрядами вперед.



Прямой, обратный и дополнительный коды

В ЭВМ в целях упрощения выполнения арифметических операций применяют специальные коды для представления чисел. При помощи этих кодов упрощается определение знака ре-

зультата операции. Операция вычитания (или алгебраического сложения) чисел сводится к арифметическому сложению кодов, облегчается выработка признаков переполнения разрядной сетки. В результате упрощаются устройства ЭВМ, выполняющие арифметические операции.

Для представления чисел со знаком в ЭВМ применяют прямой, обратный и дополнительный коды.

Общая идея построения кодов такова. Код трактуется как число без знака, а диапазон представляемых кодами чисел без знака разбивается на два поддиапазона. Один из них представляет положительные числа, другой – отрицательные. Разбиение выполняется таким образом, чтобы принадлежность к поддиапазону определялась максимально просто.

Наиболее распространенным и удобным является формирование кодов таким образом, чтобы значение старшего разряда указывало на знак представляемых чисел, т.е. использование такого кодирования позволяет говорить о старшем разряде как о знаковом (бит знака) и об остальных как о цифровых разрядах кода.

Прямой код

Это обычный двоичный код, рассмотренный в разделе двоичной системы счисления. Если двоичное число является положительным, то бит знака равен 0, если двоичное число – отрицательное, то бит знака равен 1. Цифровые разряды прямого кода содержат модуль представляемого числа, что обеспечивает наглядность представления чисел в прямом коде (ПК).

Рассмотрим однобайтовое представление двоичного числа. Пусть это будет $28_{(10)}$. В двоичном формате – $0011100_{(2)}$ (при однобайтовом формате под величину числа отведено 7 разрядов). Двоичное число со знаком будет выглядеть так, как показано на рис. 3.1.

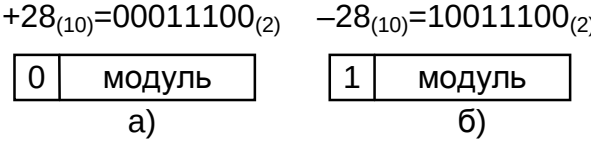


Рис. 3.1 – Формат двоичного числа со знаком в прямом коде:
а – положительное число; б – отрицательное

Сложение в прямом коде чисел, имеющих одинаковые знаки, достаточно просто: числа складываются и сумме присваивается знак слагаемых. Значительно более сложным является алгебраическое сложение в прямом коде чисел с разными знаками. В этом случае приходится определять большее по модулю число, производить вычитание модулей и присваивать разности знак большего по модулю числа. Такую операцию значительно проще выполнять, используя обратный и дополнительный коды.

Обратный код

В обратном коде (ОК), так же как и в прямом коде, для обозначения знака положительного числа используется бит, равный нулю, и знака отрицательного – единице. Обратный код отрицательного двоичного числа формируется дополнением модуля исходного числа нулями до самого старшего разряда модуля, а затем поразрядной заменой всех нулей числа на единицу и всех единиц на нули. В знаковом разряде обратного кода у положительных чисел будет 0, а у отрицательных – 1.

На рис. 3.2 приведен формат однобайтового двоичного числа в обратном коде.

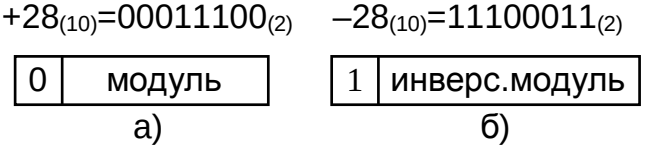


Рис. 3.2 – Формат двоичного числа со знаком в обратном коде:
а – положительное число; б – отрицательное

В общем случае ОК является дополнением модуля исходного числа до наибольшего числа без знака, помещенного в разрядную сетку.

Алгоритм формирования ОК очень прост, при этом ОК позволяет унифицировать операции сложения и вычитания в АЛУ, которые в прямом коде выполняются по-разному. Однако работа с ОК вызывает ряд трудностей. В частности, возникают два нуля: +0 и –0, т.е. в прямом коде (в котором представлены положитель-

ные числа) имеет место (+0) = 000...0, а в обратном коде (в котором представлены отрицательные числа) – (–0) = 111...1.

Кроме того, в операциях сложения и вычитания требуется дополнительная операция по прибавлению бита переноса в младший разряд суммы. Рассмотрим правила алгебраического сложения в ОК (поскольку A–B=A+(–B)). Алгоритм сложения в ОК включает в себя:

- сложение кодов, включая знаковый разряд;
- прибавление переноса к МЗР (младшему значащему разряду) суммы.

Дополнительный код

Дополнительный код (ДК) строится следующим образом. Сначала формируется обратный код (ОК), а затем к младшему разряду (МЗР) добавляют 1. При выполнении арифметических операций положительные числа представляются в прямом коде (ПК), а отрицательные числа – в ДК, причем обратный перевод ДК в ПК осуществляется аналогичными операциями в той же последовательности. На рис. 3.3 рассмотрена цепь преобразований числа из ПК в ДК и обратно в двух вариантах.

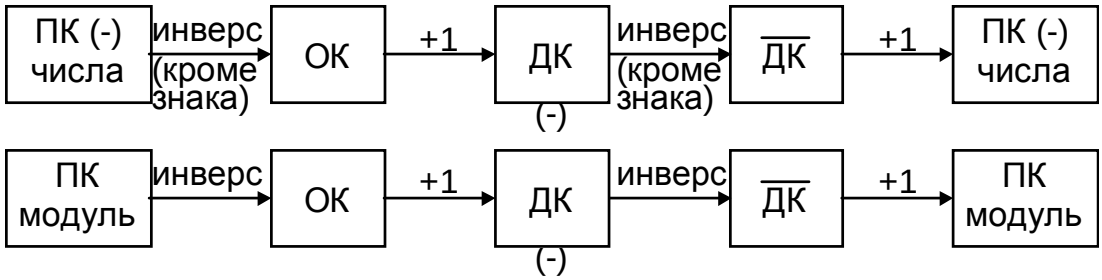


Рис. 3.3 – Два варианта преобразования чисел из ПК в ДК и обратно

Использование ДК для представления отрицательных чисел устраняет двусмысленное представление нулевого результата (возникновение двух нулей: +0 и –0), так как –0 исчезает.

В общем случае использованием ДК для записи отрицательных чисел можно перекрыть диапазон десятичных чисел от -2^{k-1} до $+2^{k-1}-1$, где k – число используемых двоичных разрядов, включая знаковый. Так, с помощью одного байта можно представить десятичные числа от –128 до +127 либо только положительные числа от 0 до 255 (здесь под положительными числами по-

нимаются числа без знака). В табл.1 приведены 4-разрядные двоичные числа от 0000 до 1111 и десятичные числа в случае представления их со знаком и без знака. Из этой таблицы следует, что в формате 4-разрядного двоичного числа могут быть представлены десятичные числа со знаком в диапазоне от –8 до +7 или десятичные числа без знака в диапазоне от 0 до +15.

Оба этих способа представления чисел (со знаком и без знака) широко используются в ЭВМ.

В ЭВМ используется быстрый способ формирования ДК. Его суть заключается в следующем. Двоичное число в ПК просматривается от МЗР к СЗР. Пока встречаются нули, их копируют в одноименные разряды результата. Первая встретившаяся единица также копируется в соответствующий разряд, а каждый последующий бит исходного числа заменяется на противоположный (0 – на 1, 1 – на 0).

Сложение и вычитание в дополнительном коде

При выполнении арифметических операций в современных ЭВМ используется представление положительных чисел в прямом коде (ПК), а отрицательных – в обратном (ОК) или в дополнительном (ДК) кодах.

Правило. При алгебраическом сложении двух двоичных чисел, представленных обратным (или дополнительным) кодом, производится арифметическое суммирование этих кодов, включая разряды знаков. При возникновении переноса из разряда знака единица переноса прибавляется к МЗР суммы кодов при использовании ОК и отбрасывается при использовании ДК. В результате получается алгебраическая сумма в обратном (или дополнительном) коде.

При алгебраическом сложении чисел со знаком результатом также является число со знаком. Суммирование происходит по всем разрядам, включая знаковые, которые при этом рассматриваются как старшие. При возникновении переноса из старшего разряда единица переноса отбрасывается и возможны два варианта результата:

- знаковый разряд равен нулю: результат – положительное число в ПК;

• знаковый разряд равен единице: результат – отрицательное число в ДК.

6.4 Глава 4. Операционные устройства

Обработка информации в ЭВМ осуществляется в АЛУ (основная обработка), а также в контроллерах ПУ (вспомогательная, предварительная обработка). Все эти устройства – АЛУ, ПУ – по принципам организации, построения относятся к классу ОУ и предназначены для выполнения операций из списка функциональных операций по инициативе ЦП.

Идея декомпозиции ОУ на ОА и УА принадлежит академику В.М. Глушкову (рис. 4.1). Конструктивность идеи в следующем.

Разделение ОУ (АЛУ прежде всего) на две части – пассивную исполнительную (ОА) и активную управляющую (УА).

АЛУ, часть узлов (сумматор и др. КС) являются исполнителями, а часть узлов (распределитель сигналов, например) являются управляющими элементами. У этих узлов разные принципы построения, организации, причем принципы построения ОА сложнее, чем УА. Кроме того, поскольку принципы организации разные, то и их проектирование тоже раздельное.

Операционное устройство представляет собой конечный автомат, состоящий из двух частей:

1) операционный автомат;

2) управляющий автомат;

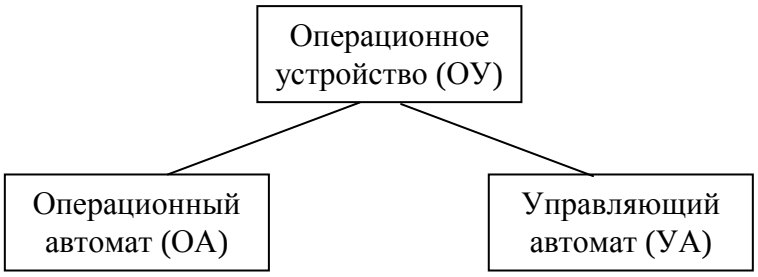


Рис. 4.1 – Декомпозиция ОУ

Операционный автомат (ОА) – предназначен для арифметических и логических операций под управлением управляющего автомата.

Управляющий автомат (УА) – предназначен для формирования управляющих воздействий.

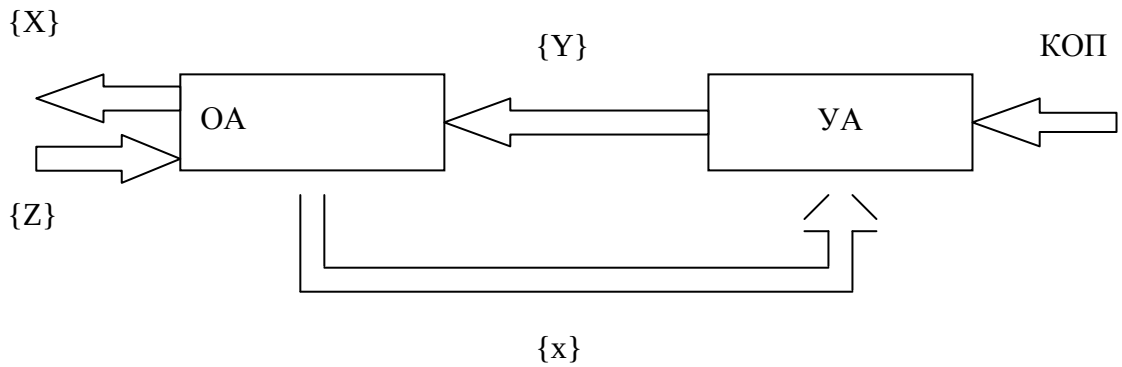


Рис. 4.2 – Функциональная схема ОУ:
КОП – код операции;
{X} – входные сигналы;
{Y} – управляющие команды;
{Z} – выходные сигналы;
{x} – множество осведомительных сигналов (количество, переполнение и т.п.).

Основа функционирования – принцип микропрограммного управления.

Организация ОУ реализует принцип микропрограммного управления:

1. Любая операция – это сложное действие, состоящее из совокупности элементарных действий, называемых микрооперациями (МО). Выполнение каждой МО осуществляется специальной комбинационной схемой (КС) за один такт машинного времени.

2. Порядок выполнения МО определяется алгоритмом операции и зависит от значений логических условий x. ЛУ принимают значения истина или ложь в зависимости от значений операндов.

3. Алгоритм, представленный, записанный в терминах МО и ЛУ, называется микропрограммой (МП). МП задает порядок выполнения МО и проверки ЛУ во времени.

4. Совокупность микропрограмм $МП_1$, …, $МП_g$ задает функцию ОУ.

Все команды, формируемые УА, называются микрокомандами (запись в триггер, считывание кода, сдвиг).

Rg A, Rg B – входные регистры;

S – сумматор;

Rg C – выходной регистр;

Перечень микроопераций:

y_1 – запись через управляющую шину во входные регистры A и B;

y_2 – проверка знака числа (сдвиг для определения знакового разряда);

y_3 – считывание кодов из регистров;

y_4 – сложение;

y_5 – проверка знака суммы;

y_6 – выдача результата на выходную шину.

Совокупность микрокоманд образует микропрограмму действия ОА под воздействием микрокоманды, называемой микрооперацией.

Операционный автомат предназначен для выполнения двух простейших операций (сложение, сдвиг), следовательно, в нём присутствуют два основных устройства – сумматор и регистры.

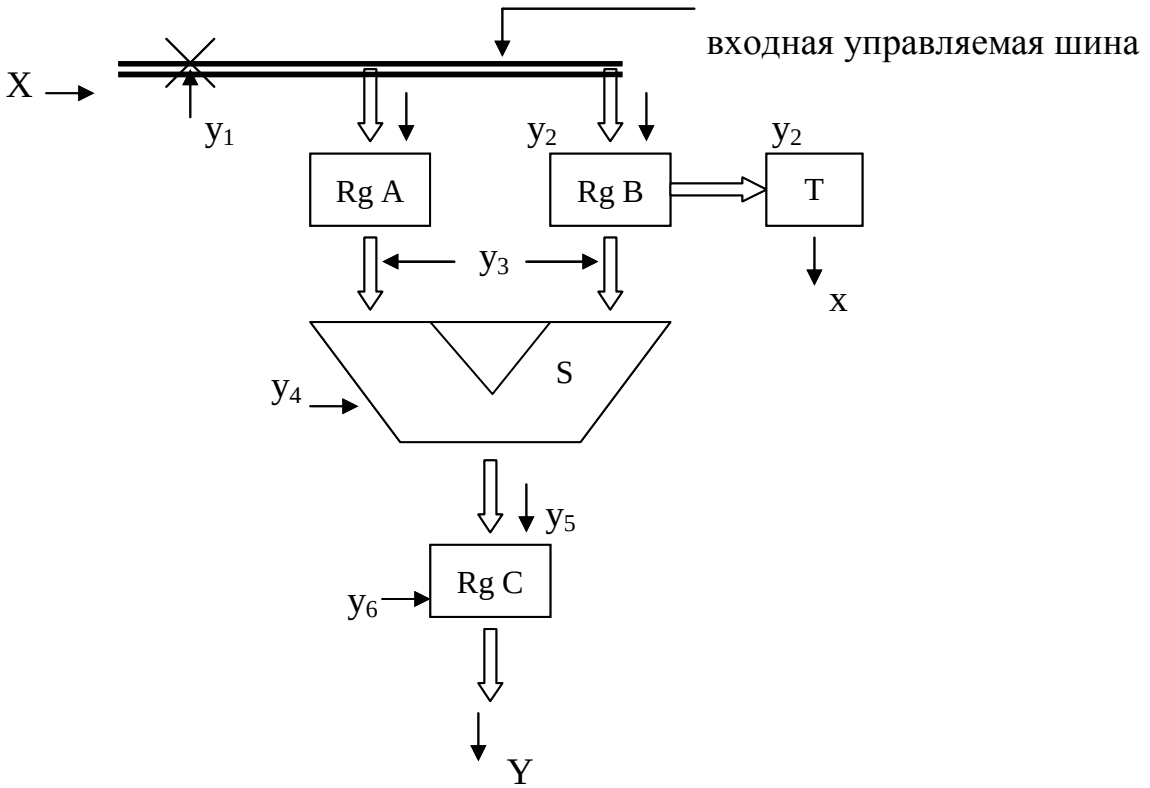
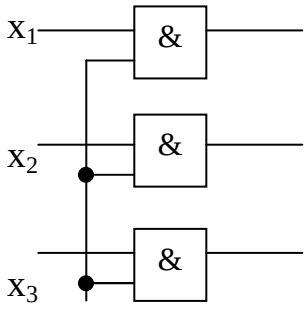


Рис. 4.3 – Структурная схема операционного автомата

Элементы базиса построения операционного автомата:
Управляемая шина



1 – управляющая команда

Микрооперации y_1, y_2, y_3, y_4, y_5, y_6 – формирует Управляющий автомат.

Типы УА:
1) УА с жёсткой логикой;
2) УА с программируемой логикой;

УА с жёсткой логикой вырабатывает управляющие воздействия путем срабатывания триггеров, число которых определяется числом устойчивых состояний в граф-схеме алгоритма и входными условиями.

В УА с программируемой логикой коды микроопераций выбираются из ячеек ПЗУ исходя из входных условий.

Пример схема алгоритма (ГСА) – представлен на рис. 4.4. Здесь Сч – счетчик циклов.

Операция умножения разделяется на 7 МО, основные из которых – сложение (реализуется за один такт комбинационным двоичным сумматором) и сдвиг (реализуется регистром сдвига). Порядок выполнения МО зависит от значений двух ЛУ (осведомительных сигналов): В(0), СЦ=0. ГСА умножения задает порядок выполнения МО и проверки ЛУ во времени. Например, в зависимости от В(0) в следующий момент времени в следующем такте будет выполняться либо МО сложения С:=С+А (если В(0)=1), либо МО сдвига, если В(0)=0.

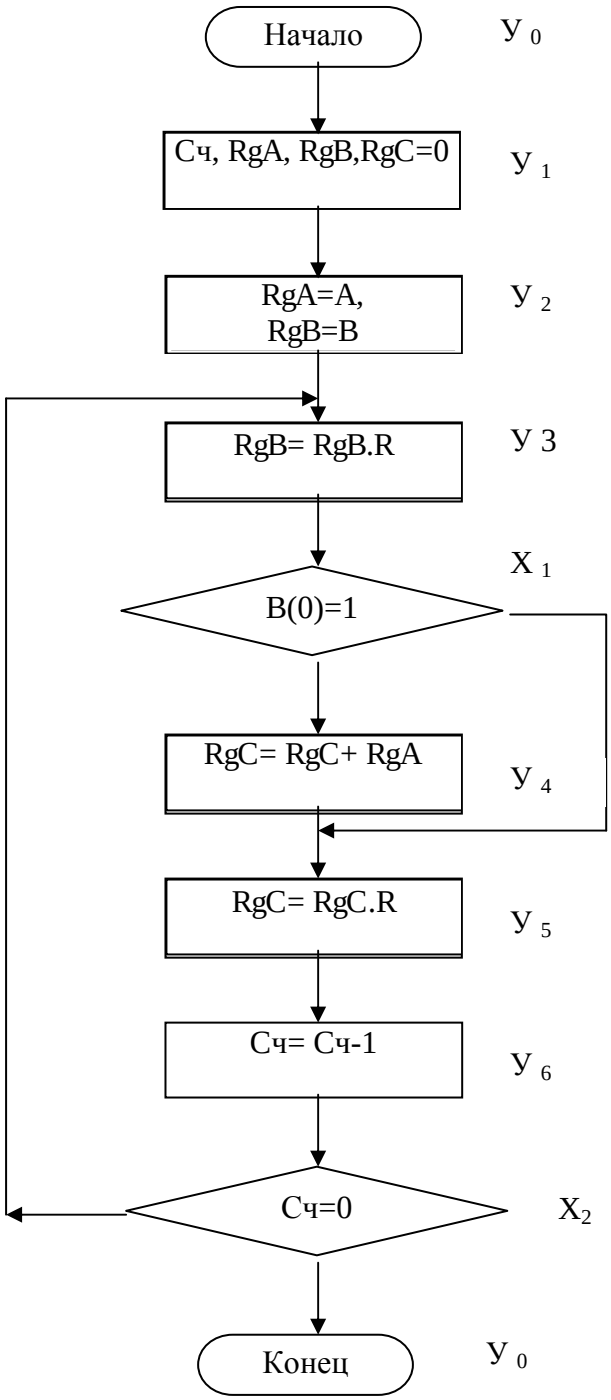


Рис. 4.4 – Пример граф-схемы алгоритма умножения

Обработка информации с помощью ОУ осуществляется путем выполнения операций из списка F в последовательности, которая задается алгоритмом (программой) решения задачи: ОА, выполняя программу, распределяет выполнение операций, предписанных командами программы, между различными ОУ – АЛУ, контроллерами ПУ.

Запуск (инициализация) операции $f_g \in F$ осуществляется путем подачи кода операции в ОУ из УУ. Реализация операции f_g осуществляется путем выполнения МО в порядке, заданном микропрограммой, хранимой внутри ОУ (т.е. без участия УУ). Работа (функционирование) ОУ во времени осуществляется тактами. Реализация МПg в общем случае занимает различное количество тактов n, т.е. время выполнения операции $t_g = nT$, где T – продолжительность такта.

Принцип микропрограммного управления является основой организации (построения) ОУ. Эти процедуры достаточно схожи с фон Неймановскими принципами функционирования. В основе управления лежит алгоритм. Только у Неймана он представляется в виде макропрограммы и поступает в процессор извне (из ОЗУ извлекается процессором). Здесь алгоритм в виде МП уже находится внутри ОУ. При выполнении программы ЦП генерирует определенную последовательность операций, реализуемых ОУ. При выполнении операции ОУ генерирует последовательность МО, реализуемых комбинационными схемами КС.

Отличия между этими принципами: 1) операция – сложное действие, для реализации которого необходимо ОУ. МО – элементарное действие, для реализации которого достаточно КС. 2) Операция выполняется за n тактов: $t_{опер} = nT$. МО выполняется за один такт (алгоритм – в структуре КС). КС управляется данными на ее входах.

Функция ОУ определяется совокупностью микропрограмм $МП_1$, …, $МП_G$, описывающих алгоритмы операций f_1, …, f_g. Для описания МП в языке используются различные средства, обеспечивающие описание слов, МО, ЛУ, а также средства, описывающие порядок их выполнения во времени.

Описание слов и массивов. Слово описывается своим именем и длиной: $C(n_1{:}n_2)$. Здесь C – имя слова, n_1, n_2 – номера старшего и младшего разрядов слова соответственно. Часть слова называется **полем** и описывается аналогично словам: A(0:7), A(0:15), B(15), A(0) и т.п.

Массивы слов (например, запоминающее устройство) описываются в виде: $M[m_1{:}m_2](n_1{:}n_2)$. Здесь M – имя массива, m_1,m_2 – номера первого и последнего элемента (ячейки) массива соответ-

ственно, n_1, n_2 – определены выше. Пример описания локальной памяти как массива регистров: ЛП[0:15](0:31) – 16 тридцатидвухразрядных регистров.

Описание МО. Для описания МО используется оператор присваивания «:=» (или «←»). Слева от оператора указывается слово, поле, составное слово или элемент массива. Справа – двоичное выражение, которое описывает правило получения результата МО. Запись двоичного выражения осуществляется при помощи символов, обозначающих различные операции над операндами (словами), представленными в двоичной форме. Например, + – сложение кодов, ∨ – логическая операция или и т.п. Примеры – в МП умножения.

Описание ЛУ. Для описания ЛУ используются различного рода отношения: «<» – меньше, «>» – больше, «=0» – равно нулю, «≠0» – не равно нулю и т.п. Примеры: A<0, B≥0, C=0 и т.п. (смотри МП умножения).

Порядок выполнения МО. Порядок выполнения МО и проверки ЛУ задается в графической форме – в виде так называемой **граф-схемы алгоритма** (ГСА). ГСА строится с использованием вершин четырех типов: начальной, конечной, операторной и условной и дуг, связывающих эти вершины (рис. 4.5).

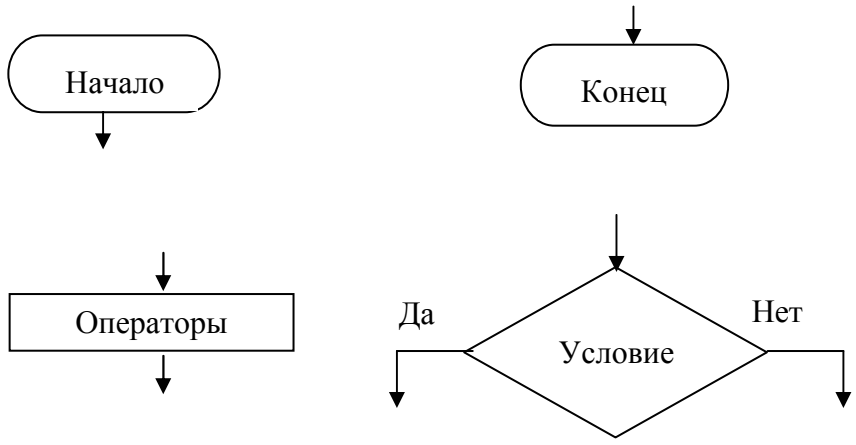


Рис. 4.5 – Элементы ГСА

Начальная вершина имеет одну выходящую дугу. Конечная вершина имеет одну входящую дугу. Операторная вершина имеет одну входящую и одну выходящую дугу. В ней записывается

один или несколько операторов присваивания, описывающих МО. Условная вершина имеет одну входящую и две исходящих, отмеченных символами «да» (1) и «нет» (0). Выход по дуге «да» осуществляется в случае, если ЛУ принимает истинное значение (1), и по дуге «нет» – если ложное значение (0).

Назначение ОА – выполнение микроопераций из списка $Y=\{y_1,\ldots,y_M\}$ под воздействием управляющих сигналов $y_m \in Y$ и формирование значений логических условий (осведомительных сигналов) $X=\{x_1, \ldots, x_L\}$.

С каждым сигналом $y_m \in Y$ в ОА отождествляется определенная МО. Поступление сигнала y_3 в ОА приводит к выполнению этой МО и записи её результата в С.

С каждым логическим условием $x_l \in X$ в ОА отождествляется значение осведомительного сигнала, который принимает значение истина (единица) или ложь (ноль).

Управляющий автомат (УА) предназначен для управления работой ОА. Он задает порядок выполнения МО в ОА путем выработки управляющих сигналов $y_m \in Y$ в той последовательности, которая задается микропрограммой операции $f_g \in F$ и значениями осведомительных сигналов $X=\{x_1,\ldots x_L\}$, поступающих из ОА.

Построение управляющих автоматов

Цифровым автоматом называется последовательностная схема, которая имеет набор состояний, обозначаемых обычно $A_1, A_2, \ldots, A_N$.

В моменты прихода тактовых импульсов автомат переходит из одного состояния в другое, определяемое как текущим состоянием, так и набором входных (осведомительных) сигналов $X_1, X_2, \ldots, X_N$, при этом формируется последовательность наборов выходных (управляющих) сигналов $Y_1, Y_2, \ldots, Y_N$.

Для каждого автомата задается закон функционирования или алгоритм переходов из одного состояния в другое под действием разных комбинаций входных сигналов с описанием комбинаций формируемых при этом выходных сигналов. Такой алго-

ритм может быть задан либо в виде графа, либо в виде таблицы переходов.

Когда автомат не работает, он находится в начальном состоянии A_1. При запуске автомат сохраняет состояние A_1 в течение одного такта, за время которого формируются соответствующие значения входных сигналов X_1. По окончании первого такта автомат переключается в очередное состояние A_2, предписанное законом функционирования, и в ОА начинает выполняться следующий набор микроопераций. Момент окончания микропрограммы отмечается возвратом автомата в начальное состояние – a_1.

Примером цифрового автомата служит счетчик – автомат, у которого нет вообще входных осведомительных сигналов, а есть только тактовый, на него подаются счетные импульсы. Число его состояний равно коэффициенту пересчета, граф этого автомата представляет собой кольцо с последовательным переходом от одного состояния к следующему. Более сложные цифровые автоматы могут иметь несколько возможных переходов из каждого состояния под действием разных наборов входных сигналов.

Автомат Мили и автомат Мура

По способу формирования выходных сигналов автоматы подразделяют на автоматы Мили и автоматы Мура.

Автомат Мили определяет закон функционирования

$$A(t + 1) = [A(t), X(t)]$$
$$Y(t) = [A(t), X(t)]$$

Выходные сигналы Y(t) зависят как от состояния автомата A(t) в текущий момент времени t, так и от входных сигналов X(t).

Автомат Мура

$$A(t+1) = [A(t), X(t)]$$
$$Y(t) = [A(t)]$$

Выходные сигналы Y(t) зависят только от состояния автомата A(t) в текущий момент времени t.

Реализация автоматов Мили проще, но в них возникают проблемы с синхронностью формирования выходных сигналов.

При практической реализации цифровых автоматов они могут быть реализованы разными способами:

- на элементах с жесткой логикой;
- на постоянном запоминающем устройстве (микропрограммные автоматы);

Управляющие автоматы на элементах с жесткой логикой

В управляющих автоматах на элементах с жесткой логикой для хранения информации о состоянии используется набор триггеров, на их тактовый вход подается тактирующий сигнал, входные сигналы X подаются на комбинационные устройства, вырабатывающие сигналы управления для (функции возбуждения) триггеров. Выходные сигналы Y формируются при помощи других комбинационных схем из выходных сигналов триггеров A (для автомата Мили) или из выходных сигналов триггеров Аи входных сигналов Y (для автомата Мура, именно его структурная схема представлена на рис. 4.6).

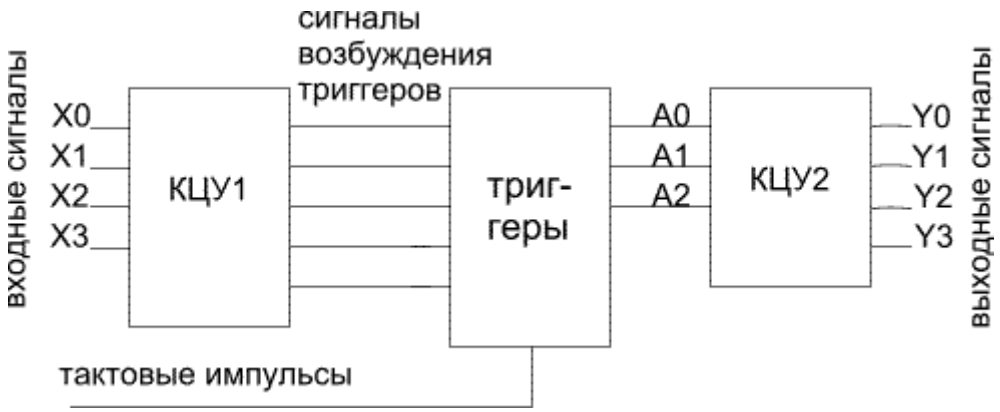


Рис. 4.6 – Схема сигналов

Подобный автомат реализуется схемой, процесс синтеза которой называется структурным синтезом. Процесс структурного синтеза автомата разделяется на следующие этапы:

- выбор типа запоминающих и логических элементов;
- кодирование состояний автомата;
- синтез комбинационной схемы, формирующей сигналы возбуждения и выходные сигналы.

В.М. Глушковым разработан общий конструктивный прием, называемый каноническим методом структурного синтеза автомата. Этот метод позволяет свести задачу синтеза автомата к за-

даче синтеза комбинационной схемы путем построения системы булевых функций, выражающих зависимость выходных сигналов и сигналов возбуждения от входных сигналов и состояний автомата. Полученные булевы функции минимизируются (например, методом карт Карно) и используются в качестве формы для построения схемы автомата.

Достоинства и недостатки автоматов с жесткой логикой

Схемы с жесткой логикой, как правило, позволяют обеспечить наибольшее быстродействие из всех возможных методов построения цифрового автомата, однако при возрастании сложности реализуемых алгоритмов схемы автоматов с жесткой логикой очень быстро становятся более сложными, чем схемы автоматов с микропрограммным или программным управлением. Поэтому схемы автоматов с жесткой логикой в настоящее время используют только в том случае, когда требуется максимальное быстродействие.

Построение схем автоматов с программируемой логикой

Автоматы с программируемой логикой являются микропроцессорными системами, в которых алгоритм функционирования реализован программным путем. Алгоритм программы для такого автомата строится следующим образом, каждая вершина графа автомата заменяется группой блоков, в которых формируются выходные сигналы и анализируются входящие, стрелки, соединяющие вершины графа автомата, заменяются командами условного или безусловного перехода.

Устойчивое состояние представляется в ячейках ПЗУ, вместо триггеров. В этом случае схема УА для всех операций одинакова (одна и та же).

Недостатки УА с ПЛ: очень длинный регистр; указываются два поля адреса (во избежание этого производится удаление одного регистра и устанавливается счетчик).

Команды: 0 – условная адресация, 1 – безусловная адресация.

Добавляется инвертор к дешифратору от выходов Yi.

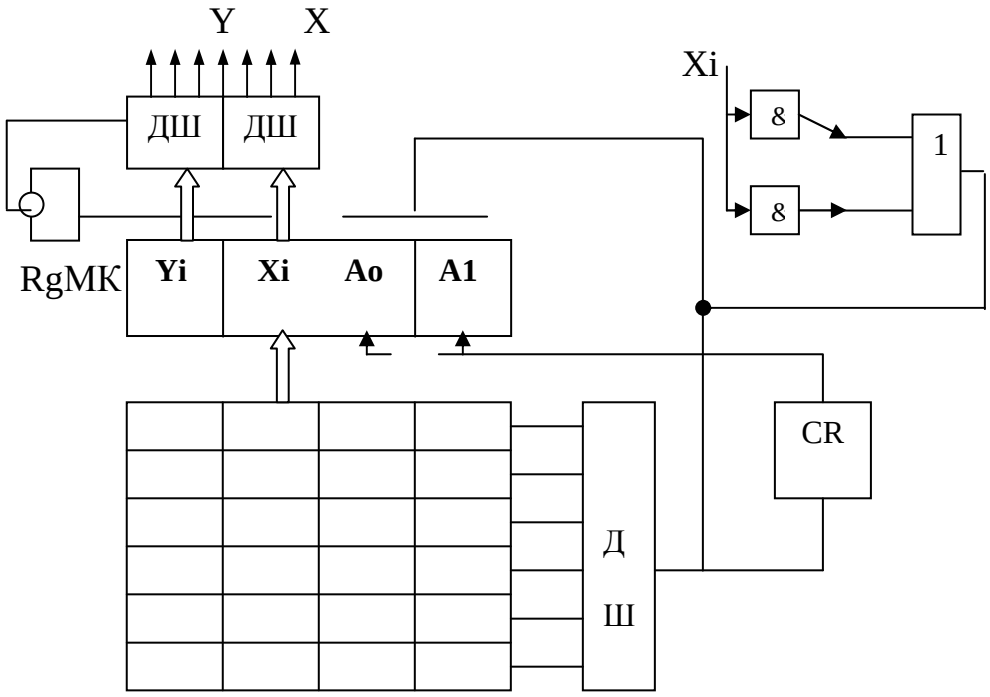
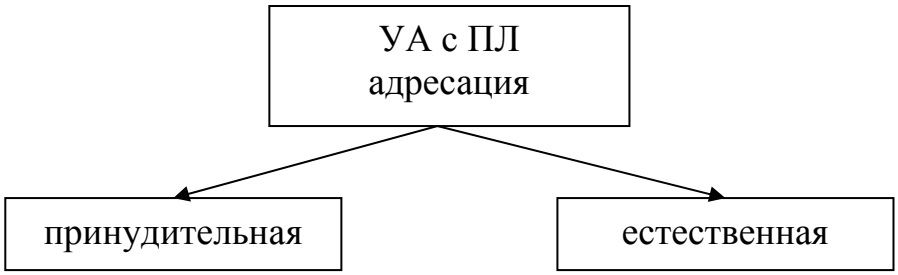


Рис. 4.7 – Функциональная схема УА с ПЛ

УА с ПЛ делятся на:

- УА с принудительной адресацией;
- УА с естественной адресацией.



По сравнению с автоматами на элементах с жесткой логикой или микропрограммных программные автоматы обладают наиболее низким быстродействием.

В то же время в случае достаточно сложных алгоритмов работы программные автоматы оказываются наиболее простыми с точки зрения схемотехники. Особенно это ощутимо в случае использования микроконтроллеров.

6.5 Глава 5. Процессоры

Простейший микропроцессор 8080

Краткая характеристика: f=2МГц; многокристальный процессор; приблизительное число транзисторов 30 тысяч; размер 5х5 мм^2; использована е – проводимость МОП. Данный процессор реализован по CISC-технологии.

Можно выделить свойства: раздельная шина адреса, шина данных и шина управления.

Состав комплекта. В состав базового комплекта входят: программируемое устройство прямого доступа в память; программируемое устройство параллельного ввода; программируемый таймер.

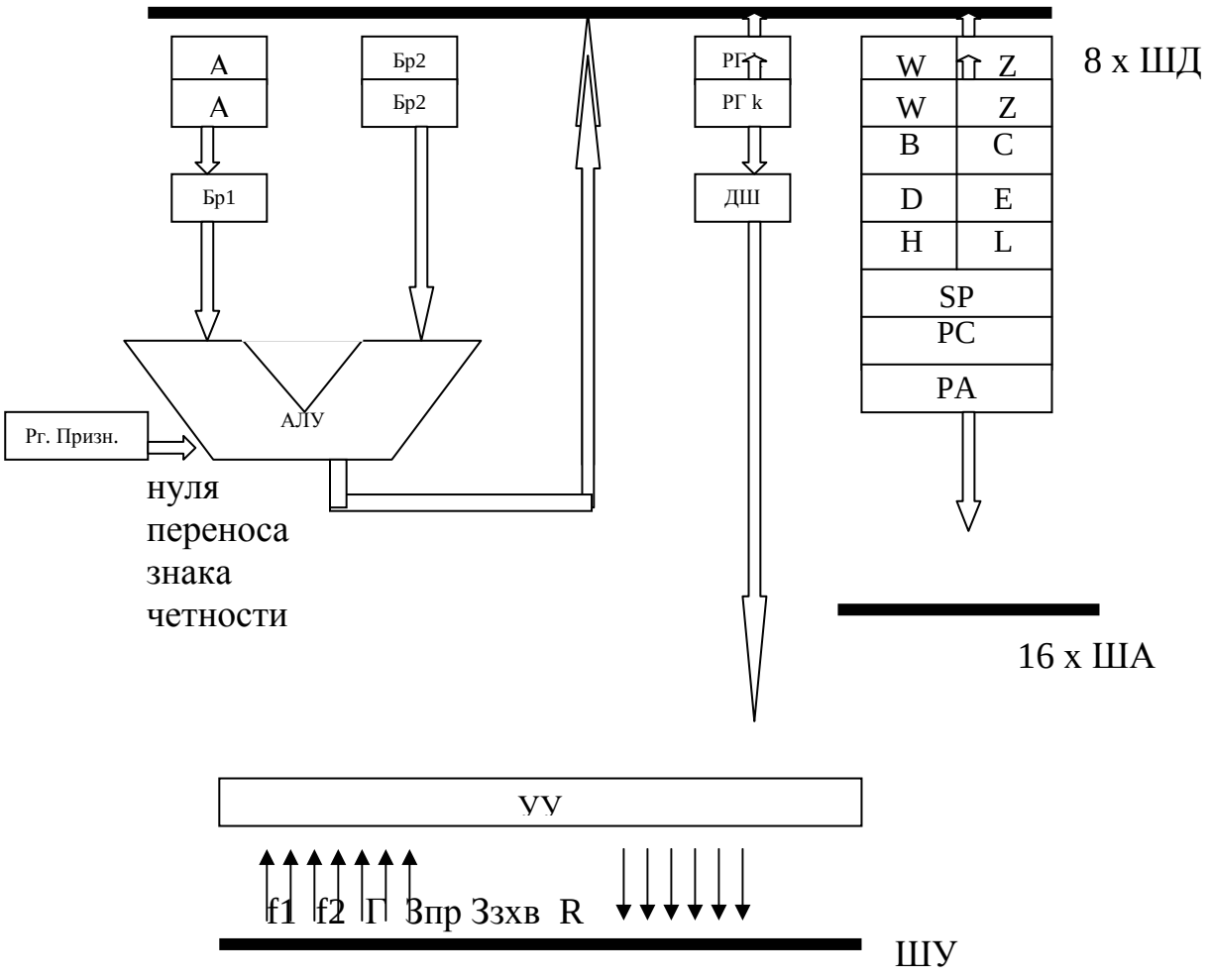


Рис. 5.1 – Структура МП 8080

Существует 10 машинных циклов. Один цикл – 3–5 тактов.

МЦ – машинный цикл, Т – машинный такт.

Если частота 2–4 МГц ⎍⎍⎍ T ~ 1/F – 0.5*10^–6 с.

Машинные циклы:

1. M1 – извлечение кода команд.
2. – чтение из памяти.
3. – запись в память.
4. – чтение из стека.
5. – запись в стек.
6. – чтение внешнего устройства.
7. – запись данных во внешнее устройство.
8. – обслуживание прерываний.
9. – останов (STOP).
10. – обслуживание прерываний в режиме останов.

Словосостояние – определяет коды и дальнейшее выполнение машинных циклов. Для хранения словосостояния (СС) существует регистр Rg СС.

СС – приходит из МП из регистра управления:

D0 – обслуживание прерывания;

D1 – запись;

D2 – стек;

D3 – обл. останов;

D4 – вывод;

D5 – M1;

D6 – ввод;

D7 – память.

Цикл Разряд.	M_1							M_8		
D0								1		
D1								1		
D2								0		
D3								0		
D4								1		
D5								0		
D6								0		
D7								0		

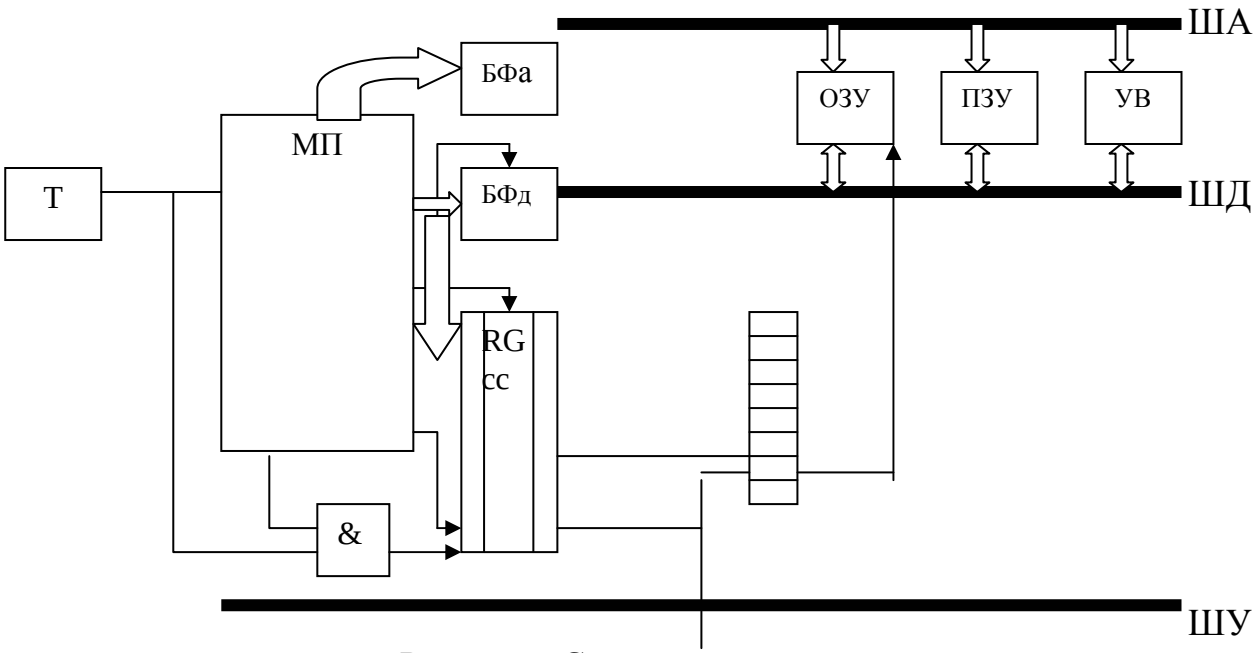
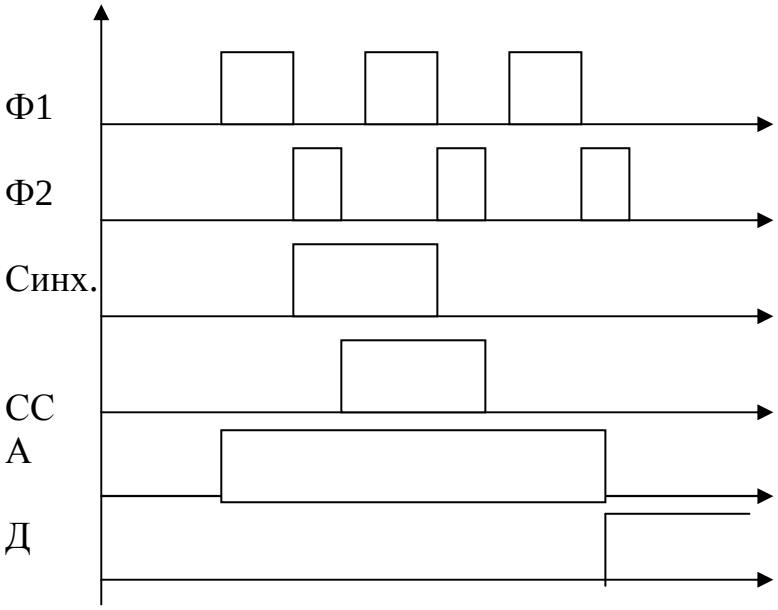


Рис. 5.2 – Структурная схема системы

Эпюры напряжений



Аппаратная реализация

1. Микропроцессор КР580ИК80.

2. Шинные формирователи (ШФ) – предназначены для соединения МКП с элементами процессорного пространства.

3. ГФ24 – генератор-формирователь – предназначен для формирования импульсов Ф1, Ф2, сигналов сброса, сигналов времени (отчетов времени).

4. УСАПП – последовательный порт 580ИК51 (ПП).

5. Программируемый параллельный интерфейс ИК55 (параллельный порт).

Организация магистрали

Согласование нагрузки низкоомное. ШФ – согласование нагрузки с МП.

Запрос захвата – формируется контроллером ПДП и поступает в МП, который проверяет выполнимость всех команд и дает разрешение на доступ к памяти, второй шаг – чтение из памяти или устройства.

ВК – выбор кристалла, Пр – приемник.

Этот интерфейс предназначен для двунаправленного обмена данными в последовательном коде между ЭВМ и внешни устройством.

УСАПП – UART интерфейс RS 232.

Существует 5 режимов работы:

- асинхронный;
- синхронный;
- передача-прием;
- внутренняя синхронизация;
- внешняя синхронизация.

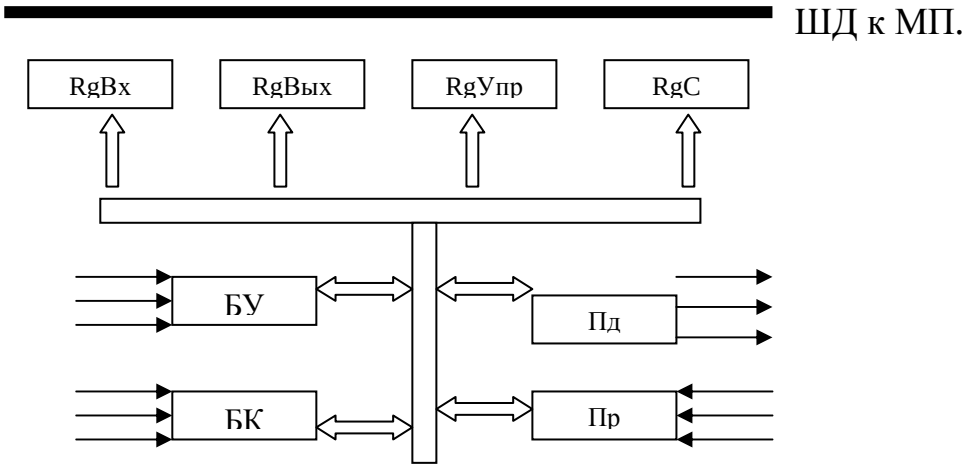


Рис. 5.3 – Схема УСАПП

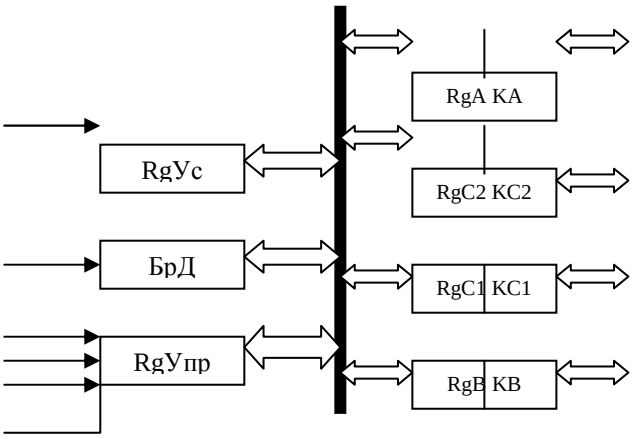
Достоинства УСАПП: можно программировать. Задаем слово команд и слово инструкций. Программируемые данные записываются в аккумулятор, далее из него по шине данных перемещаются в регистр управления.

Параллельный порт

1. Обмен данными в параллельном коде.

2. Формирование сигналов на наличие, готовность, запрос, приемо-передачу данных.

Имеется три канала: A, B, C – восьмиразрядные. Канал C – может делиться на два 4-разрядных.



A0–A1 – предназначены для адресации каналов.

Рис. 5.4 – Структурная схема параллельного порта

Можно выделить три режима функционирования:

1. KB, KA, KC1, KC2 – работают на ввод/вывод информации.

2. Информация передается через KB, KA, а KC работает для приема и вывода сигналов управления.

3. Информация идет через KB, KA, а KC – через регистр управления.

Прямой доступ к памяти (ПДП)

ПДП – это передача данных в память и из памяти без взаимодействия с микропроцессором.

В контроллере ПДП должен быть элемент, формирующий адрес для ОЗУ (счетчик) +(регистр).

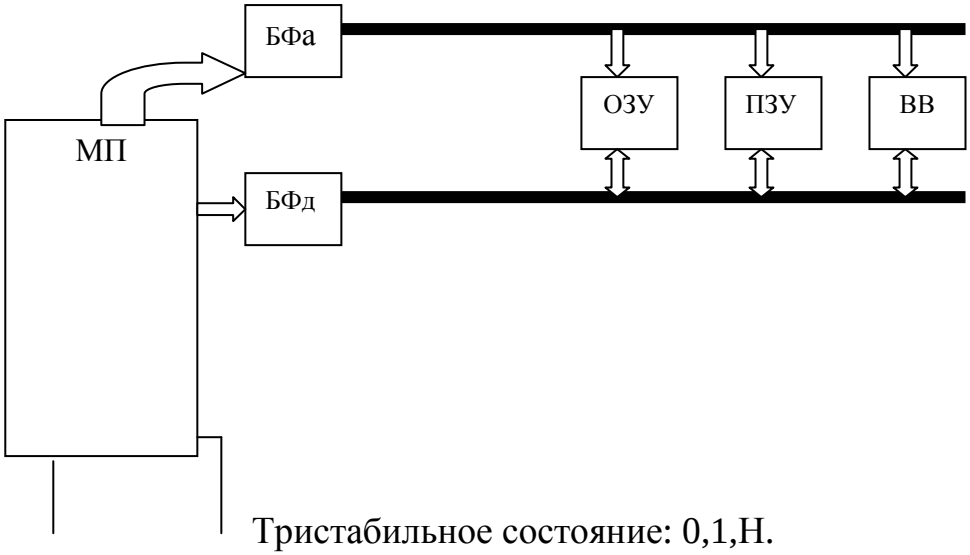
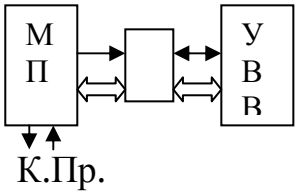


Рис. 5.5 – Структурная схема ПДП

Устройство прерывания предназначено для обеспечения возможностей связи устройств ввода-вывода с микропроцессором путем согласования скоростей.



Организация памяти

Физическая память, к которой микропроцессор имеет доступ по шине адреса, называется оперативной памятью. На самом нижнем уровне помять компьютера можно рассматривать как массив битов. Один бит может хранить значение 0 либо 1. Для физической адресации битов и работы с ними идеально подходят логические схемы. Но микропроцессору неудобно работать с памятью на уровне битов, поэтому реально ОЗУ организовано как последовательность ячеек – байтов. 1 байт = 8 битов. Каждому байту соответствует свой уникальный адрес (его номер), называемый физическим. Диапазон значений физических адресов зависит от разрядности шины адреса микропроцессора.

Механизм управления памятью полностью аппаратный. Это означает, что программа не может сама сформировать физический адрес памяти на адресной шине. Ей приходится «играть» по правилам процессора. Механизм позволяет обеспечить:

- компактность хранения адреса в машинной команде;
- гибкость механизма адресации;
- защиту адресных пространств задач в многозадачной системе;
- поддержку виртуальной памяти.

Микропроцессор аппаратно поддерживает несколько моделей использования оперативной памяти:

- сегментированная модель. В этой модели память для программы делится на непрерывные области памяти (сегменты), а сама программа может обращаться только к данным, которые находятся в этих сегментах;
- страничная модель. Ее можно рассматривать как надстройку над сегментированной памятью. В случае использования этой модели оперативная память рассматривается как совокупность блоков фиксированного размера 4Кб. Основное применение этой модели связано с организацией виртуальной памяти, что позволяет операционной системе использовать для работы программ пространство большее, чем объем физической памяти.

Виды адресации операндов в памяти

1. Прямая адресация.
2. Непосредственная.
3. Косвенно-базовая (регистровая) адресация.
4. Косвенно-базовая (регистровая) адресация со смещением.
5. Косвенная индексная адресация со смещением.
6. Косвенно-базовая индексная адресация.
7. Косвенно-базовая индексная адресация со смещением.

Микропроцессор 80x86

Краткая характеристика: f=33 МГц; приблизительное число транзисторов 29 тысяч; размер 10x10 мм2; использована e – проводимость МОП. Данный процессор реализован по CISC-технологии. Можно выделить свойства: раздельная шина адреса, шина данных и шина управления; регистры общего назначения, расши-

ренный стек, конвейер, реализована сегментация памяти, существует возможность мультипроцессирования.

Программистская модель процессора включает систему команд, перечень (спецификацию) всех регистров, возможность описания памяти, внешних устройств. Отобразим программистскую модель микропроцессора 80x86 на следующем рисунке.

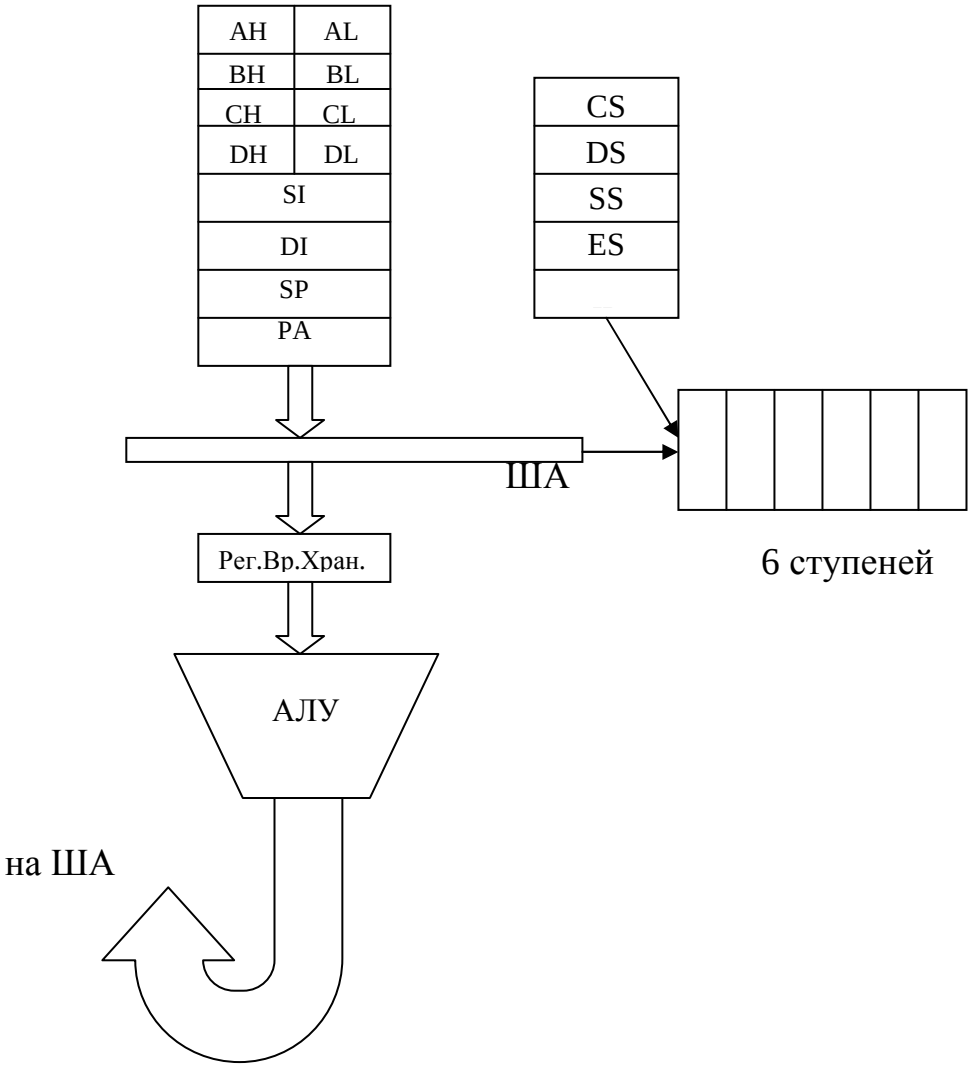


Рис. 5.6 – Структура микропроцессора 80×86

Микропроцессор 80286

Краткая характеристика: f=33 МГц; корпус с 68 выводами. Шина адреса 24 бита, шина данных 16 бит.

GDTR, LDTR – регистры локальные дескриптивной таблицы. Независимое выполнение четырех задач.

Защищенный режим. TR – регистр задач. Присутствует наличие дескриптивных регистров, предназначенных для описания сегментных регистров.

Виртуальный режим. Увеличены вычислительные возможности примерно в два раза, усовершенствованы средства доступа, новый таймер, имеется совместимость с нижестоящими сериями. Команды – 246+16 команд управления памятью, шины адреса и шины данных разделены.

Микропроцессор Intel 80386 (i386)

Обновления:

- технология CHMOS (H – высокой плотности – 0,8 микронов);
- применение 32 разрядов в архитектуре;
- размер кристалла 10x10 мм;
- 275 тыс. транзисторов;
- частоты 16, 20, 25, 33 Мгц;
- 3,8 операций в секунду.

Функциональные новшества:

- многозадачность;
- встроенное управление памятью;
- виртуальная память;
- разделение на страницы;
- защищенный режим;
- большое адресное пространство от 1 Мбайта до 4 Мбайт;
- расслоение памятью;
- кэш-память.

МП i386 поддерживает две операционные системы (ОС) – UNIX и MS_DOS.

Несмотря на введение в него последних достижений микропроцессорной техники, 80386 сохраняет совместимость по объектному коду с программным обеспечением, в большом количестве написанным для его предшественников, 8086 и 80286. Особый интерес представляет такое свойство 80386, как виртуальная машина, которое позволяет 80386 переключаться в выполнении программ, управляемых различными операционными системами, например UNIX и MS-DOS. Это свойство позволяет производи-

телям оригинальных систем непосредственно вводить прикладное программное обеспечение для 16-битных машин в системе на базе 32-битных микропроцессоров.

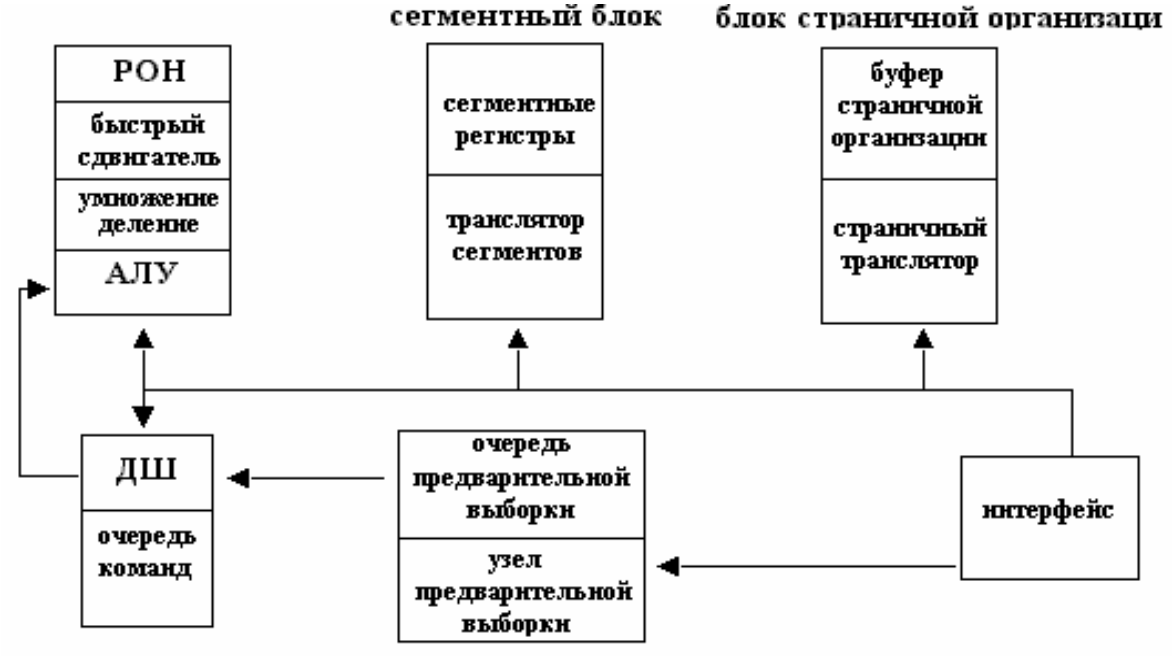


Рис. 5.7 – Функциональная структура микропроцессора i386

32-битная архитектура 80386 обеспечивает программные ресурсы, необходимые для поддержки «больших» систем, характеризуемых операциями с большими числами, большими структурами данных, большими программами (или большим числом программ) и т.п. Физическое адресное пространство 80386 состоит из 2 байт или 4 гбайт; его логическое адресное пространство состоит из 2 байт или 64 терабайт (тбайт). Восемь 32-битных общих регистров 80386 могут быть взаимозаменяемо использованы как операнды команд и как переменные различных способов адресации. Типы данных включают в себя 8-, 16- или 32-битные целые и порядковые, упакованные и неупакованные десятичные, указатели, строки бит, байтов, слов и двойных слов. Микропроцессор 80386 имеет полную систему команд для операций над этими типами данных, а также для управления выполнением программ. Способы адресации 80386 обеспечивают эффективный доступ к элементам стандартных структур данных: массивов, записей, массивов записей и записей, содержащих массивы.

В 80386 также используются регистр GDT 32-разрядная адресация + смещение, регистр LDTR (16-разрядный селектор), 64-разрядный регистр для ускоренной операции сдвига, 4 разряда управления CR0 – CR3.

CR1 – резервный;

CR2 – линейный адрес отката страниц;

CR3 – регистр, отвечающий за разбиение на страницы.

Разряд CR0 состоит из:

P	Резерв	1	2	3	4	5

Если P=1, то включается механизм разбиения на страницы.

Остальные пять полей характеризуют:

1 – включение защиты;

2 – присутствие сопроцессора;

3 – эмуляцию сопроцессора;

4 – переключение задач;

5 – тип расширения процессора.

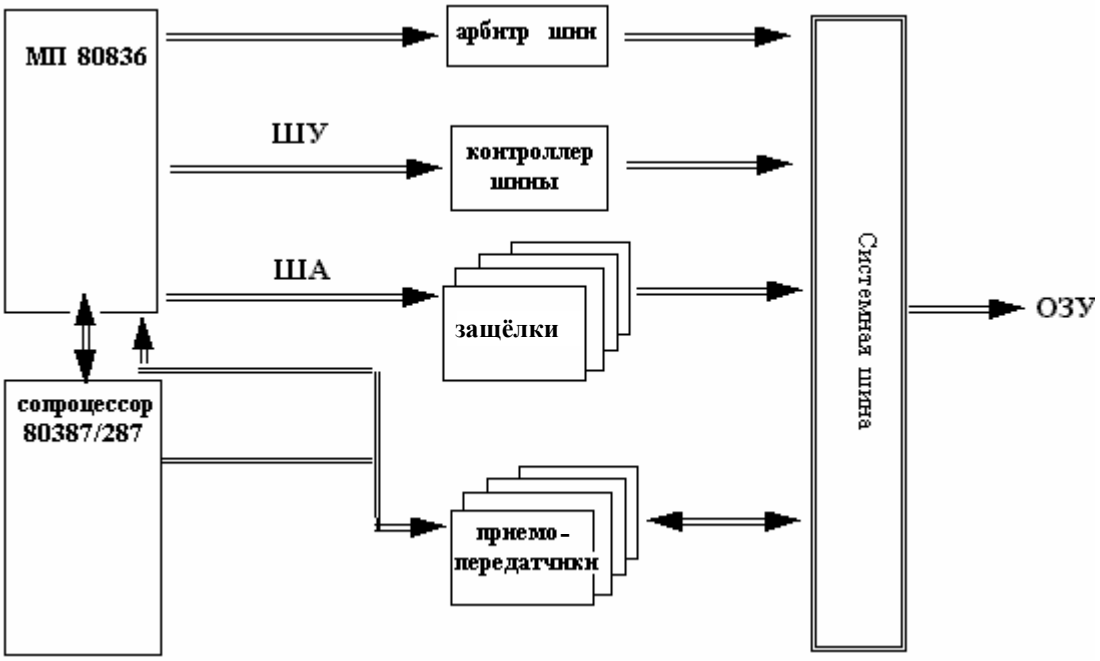


Рис. 2 – Физическая структура микропроцессора i386

Микропроцессор Intel 80486 (i486)

Обновления:

- увеличение плотности монтажа;
- увеличение количества регистров до 31:

- РОН (16 регистров),
- регистры для защищенного режима,
- DR0 – DR7 – регистры отладки,
- TR3 – TR5 – регистры проверки кэш-памяти,
- TR6 – TR7 – регистры контроля буфера трансляции страниц;

- формат команды – 1–15 байт:
  - предел операнда;
  - предел адреса;
  - размер адреса;
  - байт повтора;
- шина на 64 бита;
- одночастотная синхронизация;
- передача данных через шину пакетными циклами по 32 бита за 1 такт;
- встроенный самоконтроль.

Качественные характеристики:

- выполнение полного набора арифметических и логических операций по 8/16/32 бита;
- прямая адресация 4-Гбитной физической памяти при раздельных ША и ШД;
- поддержка внутреннего устройства с плавающей точкой;
- целостность памяти поддерживается сегментной и страничной адресацией;
- наличие внутренней кэш-памяти объемом 8 Кбайт (предусмотрена внешняя кэш-память);
- совмещение процедур выборки декодирования, преобразование адреса, выполнение команды за 1 такт (аппаратная реализация);
- передача данных через системную шину двойным словом за 1 такт.

В данном МП i486 произошло повышение быстродействия генератора в 5 раз. Он на 100% программно совместим с микропроцессорами 386(TM) DX & SX. Один миллион транзисторов объединенной кэш-памяти (сверхбыстрой оперативной памяти), вместе с аппаратурой для выполнения операций с плавающей запятой и управлением памяти на одной микросхеме, тем не менее,

поддерживают программную совместимость с предыдущими членами семейства процессоров архитектуры 86. Часто используемые операции выполняются за один цикл, что сравнимо со скоростью выполнения RISC-команд. Новые возможности расширяют многозадачность систем. Новые операции увеличивают скорость работы с семафорами в памяти. Оборудование на микросхеме гарантирует непротиворечивость кэш-памяти и поддерживает средства для реализации многоуровневого кэширования. Встроенная система тестирования проверяет микросхемную логику, кэш-память и микросхемное постраничное преобразование адресов памяти. Возможности отладки включают в себя установку ловушек контрольных точек в выполненяемом коде и при доступе к данным. Внутренний кэш прозрачен для работающих программ. Процессор i486 может использовать внешний кэш второго уровня вне микросхемы процессора.

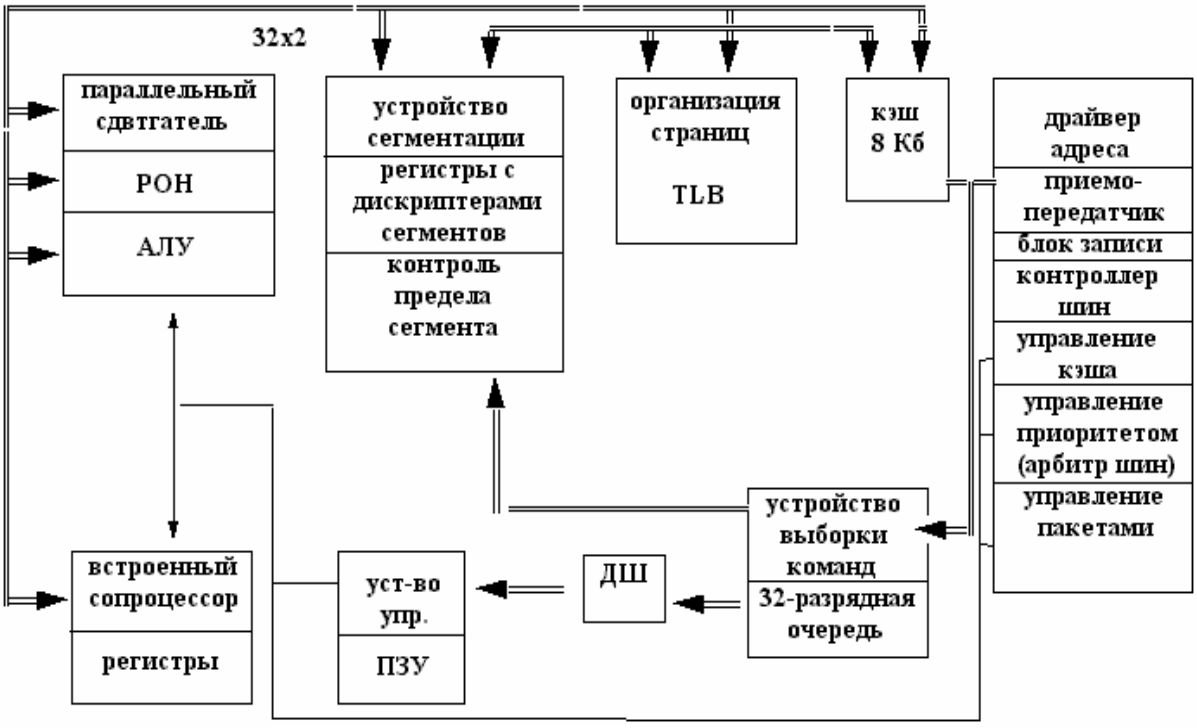


Рис. 5.9 – Структурная схема микропроцессора i486

Микропроцессор Intel Pentium

Обновления:

- 3,6 млн. транзисторов;

- суперскалярная архитектура (многопроцессорность, одновременное выполнение многих операций);
- раздельное копирование кэш-команд и данных;
- предсказание правильного адреса перехода;
- высокопроизводительное вычисление с плавающей запятой;
- расширенная до 64 бит шина данных (ШД);
- поддержка многопроцессорного режима (аппаратная);
- средства задания размеров страниц;
- средства обнаружения ошибок и функциональной избыточности;
- самотестирование;
- управление производительностью (ведется статистика, обращение к шине, правильность переходов);
- наращиваемость мощности с помощью овердрайв-процессоров;
- двухконвертная организация архитектуры;
- выполнение операций за 1 такт;
- кэш с обратной записью (128–256 Кбайт);
- блок предсказания переходов производит дополнение каждого програмного цикла программы и предсказывает наиболее вероятный переход (до 80%);
- 8-тактовый конверт;
- передача 528 Мб/сек по шине;
- два интеджер-процессора;
- аппаратная реализация операций умножения, деления, сдвига.

Первые Pentium работали на частотах 60 и 66 MHz и были рассчитаны на напряжение питания 5V. В процессоре Pentium впервые появляется предсказание переходов. Переход – это изменение последовательности выполнения команд в соответствии с алгоритмом программного обеспечения. Согласно статистике переходы встречаются в среднем через каждые 6 команд. Различают безусловные переходы (типа GOTO), когда управление передается по новому указанному адресу, и условные (типа IF), когда изменяется ход выполнения программы в зависимости от результатов сравнения. Условные переходы снижают общую производительность процессора, так как в ожидании этого перехода

конвейер работает вхолостую. Поэтому имеется специальный буфер адресов перехода, который хранит данные о последних переходах. Путем применения специальных алгоритмов предсказания переходов удается с той или иной точностью предсказать следующий переход, согласно которому и ведутся дальнейшие вычисления, чтобы процессор не простаивал, а потом, после осуществления перехода, выдаются уже готовые (или не совсем готовые, но уже прошедшие некоторый путь по конвейеру) результаты. Максимальная эффективность предсказаний для CPU Pentium составляет примерно 80%.

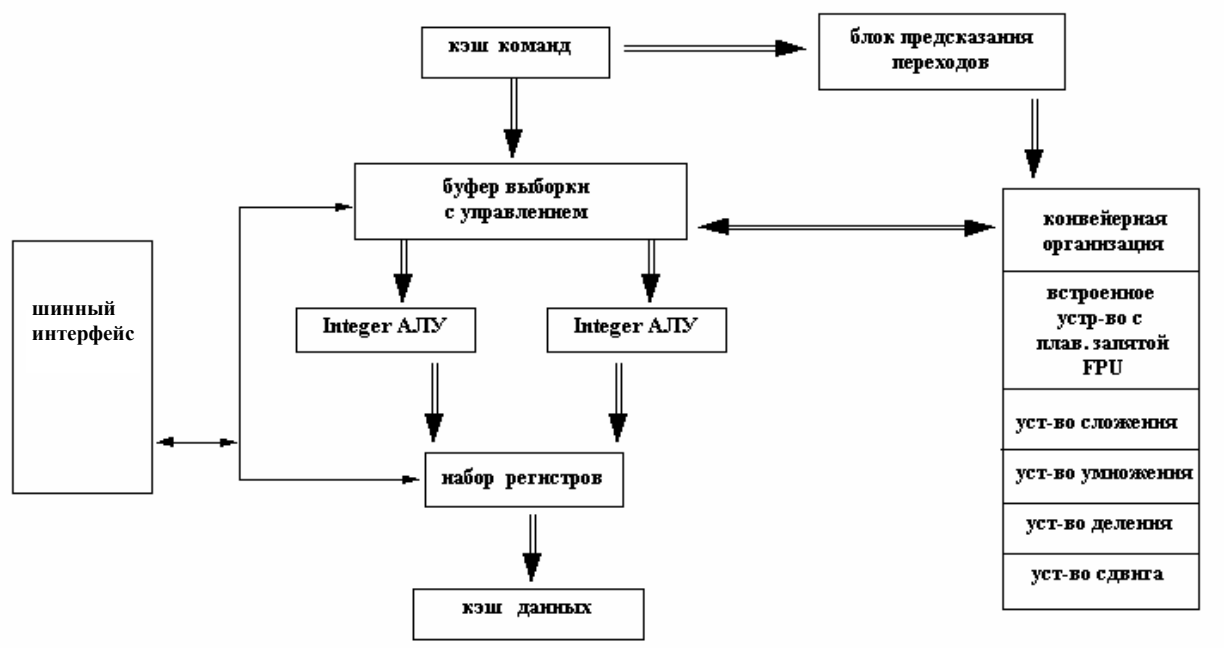


Рис. 5.10 – Структурная схема микропроцессора Intel Pentium

Микропроцессор Intel Pentium P6 (Pentium Pro)

CPU Pentium P6 имеет четырнадцать ступеней при конвейерной обработке, а количество самих конвейеров увеличилось до трех. Количество транзисторов теперь составляет 4,5–5 миллионов. Выполнение команд стало потоновым. Применяются статистический и динамический методы предсказания переходов, что повышает их эффективность до 90%. Впервые (уже который раз мне приходится писать это слово) L2 cache стал встроенным в микросхему самого процессора, что не замедлило сказаться на эффективности использования процессорного времени, так как

кэш-память теперь смогла работать на более высоких частотах по сравнению с системной платой. Впоследствии внешний кэш стал встраиваться во все Intel'овские (и не только Intel'овские) микропроцессоры. CPU Pentium P6 функционируют на частотах 133, 150, 166 и 200 MHz. Pentium P6 выполняет расчеты на 20–40% быстрее, чем обычный Pentium. Pentium P6 также поддерживает многопроцессорные (до 4-х штук в системе) конфигурации.

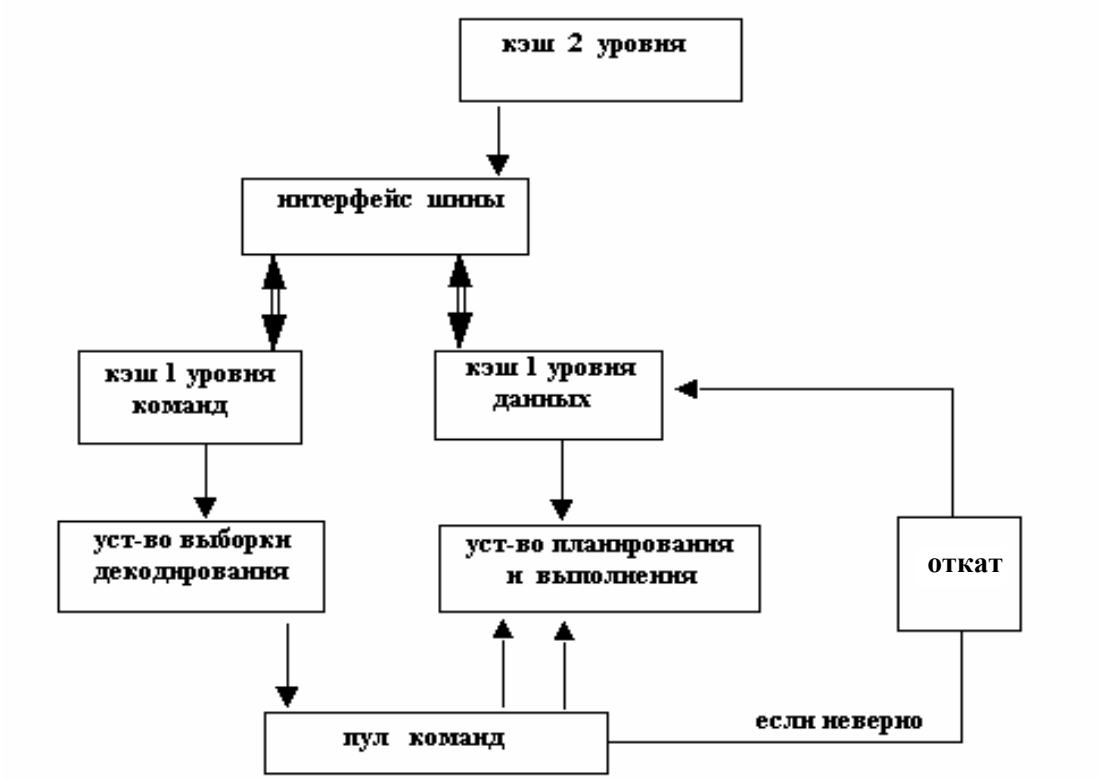


Рис. 5.11 – Структурная схема микропроцессора Intel Pentium P6

6.6 Глава 6. Организация ввода-вывода

Простейшая схема подключения устройств ввода-вывода к компьютеру с общей шиной (рис. 6.1). Шина имеет три набора линий:

- адреса
- данные
- управляющие сигналы.

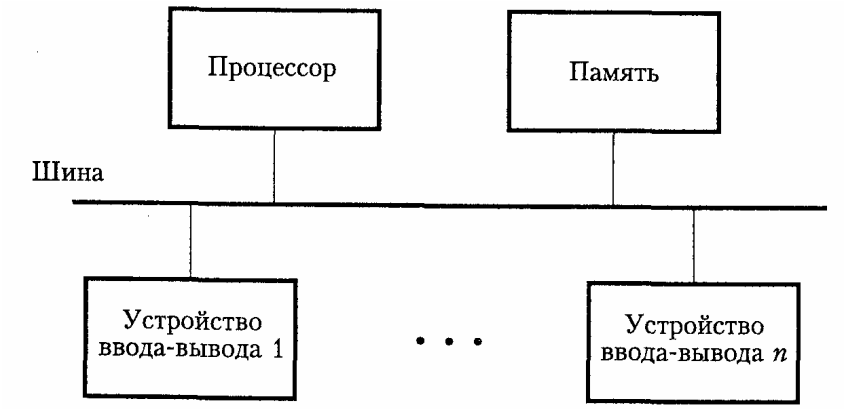


Рис. 6.1 – Архитектура ввода-вывода с общей шиной

Каждому устройству ввода-вывода присваивается уникальный набор адресов. Функционирование:

1. Устройство генерирует запрос прерывания.
2. Процессор прерывает текущую выполняемую программу.
3. Последующие прерывания запрещаются, для чего изменяются управляющие биты в регистре состояния (за исключением схем, в которых линия запроса прерывания управляется фронтом сигнала).
4. Устройство информируется о том, что его запрос распознан, и в ответ сбрасывает сигнал запроса на прерывание.
5. Запрошенное прерыванием действие выполняется программой обработки прерывания.
6. Прерывания разрешаются, выполнение программы возобновляется.

Организация системы ввода-вывода, при которой устройства ввода-вывода и память разделяют одно адресное пространство, называется вводом-выводом с отображением в память. Обмен данными с устройствами ввода-вывода выполняется с помощью тех же машинных команд, что и при обращении к памяти. Это упрощает ввод-вывод.

Три варианта – ввод-вывод с отображением в память:

- раздельные адресные пространства;
- номер порта ввода-вывода – назначается каждому управляющему регистру 8- или 16-разрядное целое число. Адресные пространства ОЗУ и устройства ввода-вывода в этой схеме не пересекаются.

Недостатки первого варианта:

- для чтения и записи нужны специальные команды (IN и OUT);
- необходим специальный механизм защиты от процессов;
- необходимо сначала считать регистр устройства в регистр процессора;
- одно адресное пространство, отображаемый на адресное пространство памяти ввод-вывод;
- регистры отображаются на адресное пространство памяти.

Недостатки второго варианта:

- при кэшировании памяти могут кэшироваться и регистры устройств;
- устройства должны проверять все обращения к памяти, чтобы определить, на какие им реагировать;
- смешанное адресное пространство;
- смешанное адресное пространство используется в x86 и Pentium.

Третий вариант объединяет достоинства других вариантов:

- от 0 до 64К отводится портам,
- от 640 до 1М зарезервировано под буферы данных.

Аппаратные элементы, необходимые для присоединения устройств ввода-вывода к шине, представлены на рис. 6.2.

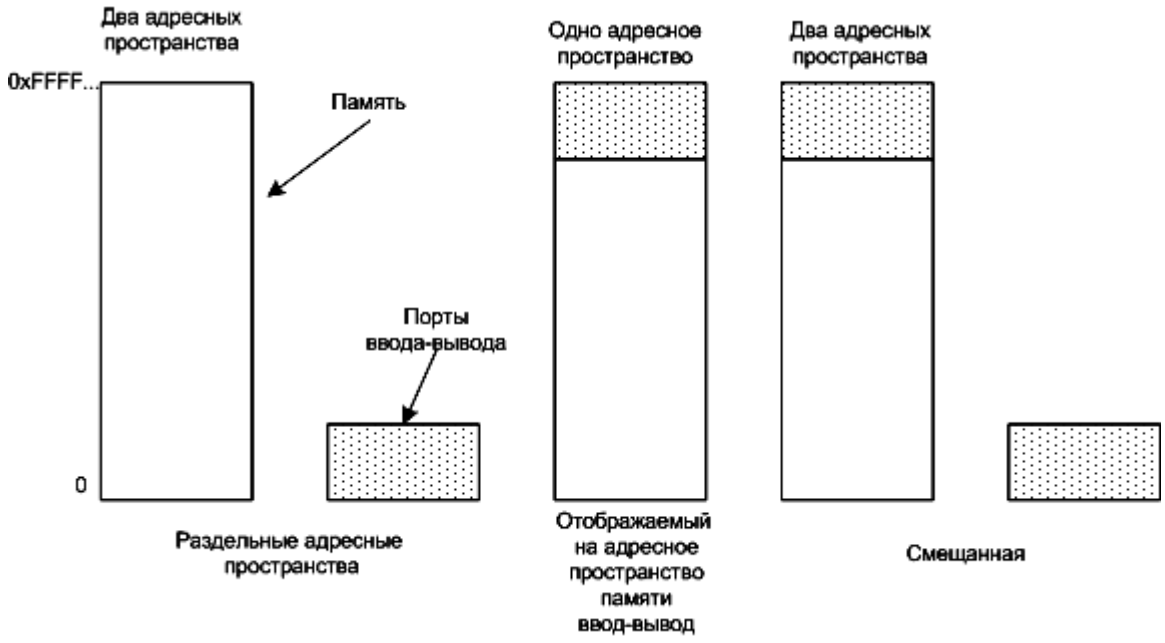


Рис. 6.2 – Варианты реализации ввода-вывода
с отображением в память

При появлении адреса, устройство его дешифрирует.

Регистры данных (РД) содержит данные.

Регистр состояния (РС) содержит информацию о состоянии устройств. Регистры данных и состояния соединяются шиной данных, и им присваиваются уникальные адреса.

Дешифратор адреса, регистры данных и состояния, управляющие схемы необходимы для координирования операций ввода-вывода, это – интерфейс устройства.

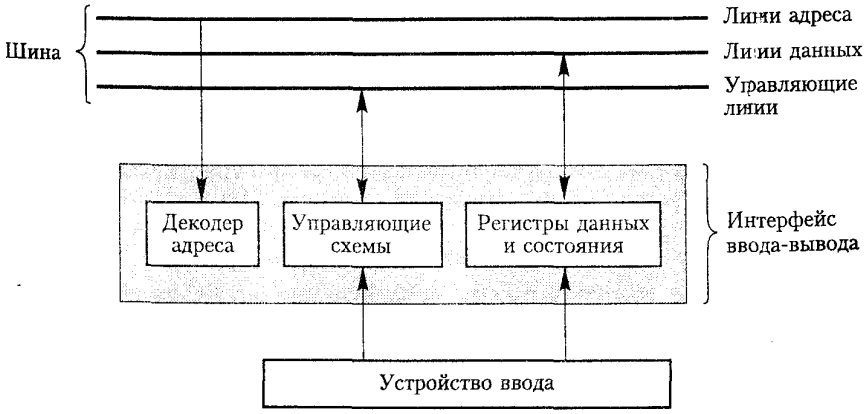


Рис. 6.3 – Интерфейс ввода-вывода для устройства

Ввод-вывод с прерываниями

Прерывание – это процесс, временно переключающий микропроцессор на выполнение другой программы с последующим возобновлением выполнения прерванной программы.

Достоинство: не нужен опрос всех устройств ввода-вывода, устройство само оповещает процессор о своей готовности к пересылке данных.

Недостатки:

- нужна отдельная линия для сигналов запроса и подтверждения;
- на каждый ввод-вывод нужно отдельное прерывание;
- время на завершение текущей команды;
- время на сохранение информации о состоянии прерванной программы (обычно в стеке);
- время на переход к подпрограмме (обычно по вектору прерывания);
- время на восстановление состояния и возврат из подпрограммы.

Функционирование

После того как устройство ввода-вывода начало работу, процессор переключается на другие задачи.

Чтобы сигнализировать процессору об окончании работы, устройство инициализирует прерывание, выставляя сигнал на выделенную устройству линию шины.

Контроллер прерываний – обслуживает поступающие прерывания от устройств.

1. Если необработанных прерываний нет, прерывание выполняется немедленно.

2. Если необработанные прерывания есть, контроллер игнорирует прерывание. Но устройство продолжает удерживать сигнал прерывания на шине до тех пор, пока оно не будет обработано.

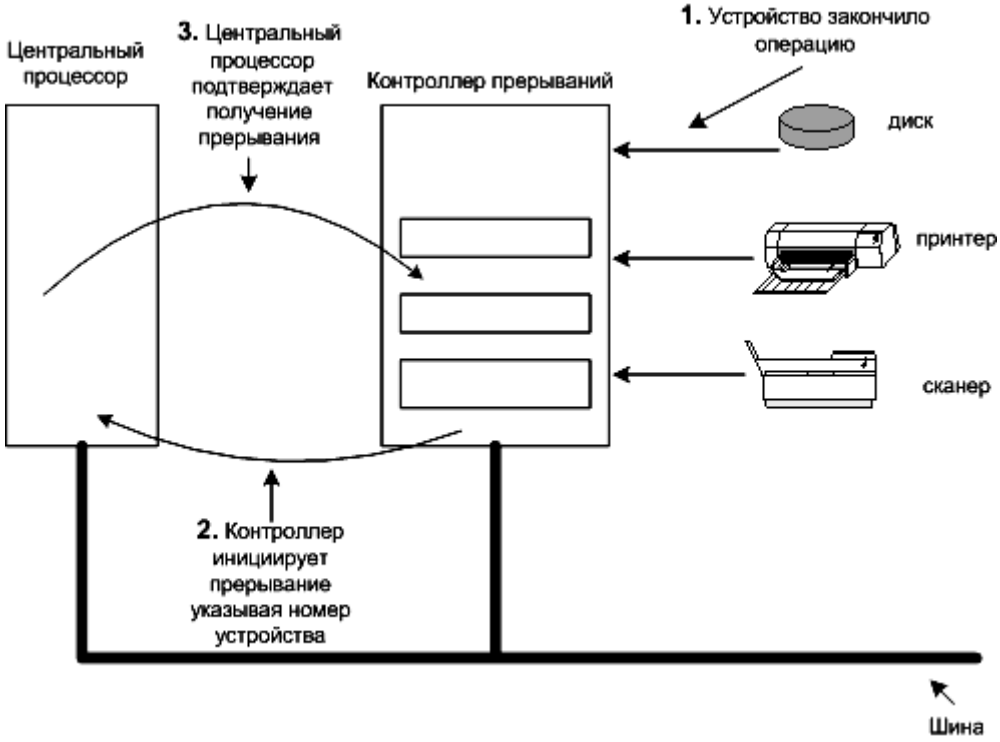


Рис. 6.4 – Функционирование в режиме прерываний

Алгоритм работы:

- Устройство выставляет сигнал прерывания.
- Контроллер прерываний инициирует прерывание, указывая номер устройства.
- Процессор начинает выполнять обработку прерывания, вызывая процедуру.

Эта процедура подтверждает получение прерывания контроллеру прерываний.

Возникновение проблем при поступлении запросов на прерывание одновременно от нескольких устройств. Для обслуживания прерываний может использоваться одна линия, и каждое из устройств подсоединяется к этой линии с помощью ключа, соединенного с «землей». Для того чтобы запросить прерывание, устройство замыкает этот ключ. Поскольку замыкание одного или нескольких ключей приводит к падению напряжения на линии до 0, значение INTR является логической суммой (ИЛИ) запросов отдельных устройств.

При получении процессором запроса прерывания по общей линии необходимо выбрать активное устройство, активизирующее линию. (Разряд IPQ в регистре состояний). Когда первое устройство будет обслужено, наступит очередь второго. Нужно время на проверку IRQ.

Вводится система приоритетов. Запросы прерываний принимаются от устройств с более высоким приоритетом. Многоуровневая схема приоритетов реализуется отдельными линиями запроса и подтверждением прерываний для каждого устройства. Каждой линии запроса прерывания присваивается свой уровень приоритета, и запросы прерываний направляются на арбитражную схему процессора. Запрос принимается только в том случае, если у него более высокий уровень приоритета, чем у процессора в данный момент.

Ввод-вывод с прямым доступом к памяти (DMA – Direct Memory Access).

Доступ непосредственного обращения к памяти, минуя процессор. Процессор отвечает только за программирование DMA: настройку на определенный тип передачи, задание начального адреса и размера массива обмениваемых данных. Операции ПДП выполняются управляющей схемой (контроллер ПДП), входящей в состав интерфейса устройства ввода-вывода. Контроллер ПДП выполняет ту же задачу, что и процессор, обращающийся к основной памяти. Для каждого пересылаемого слова он генерирует адрес памяти и сигналы шины, управляющие пересылкой данных. Контроллер ПДП увеличивает адрес, по которому будет

записываться каждое следующее слово, и отслеживает количество таких операций.

Прямой доступ к памяти реализуется с помощью DMA – контроллера (рис. 6.5).

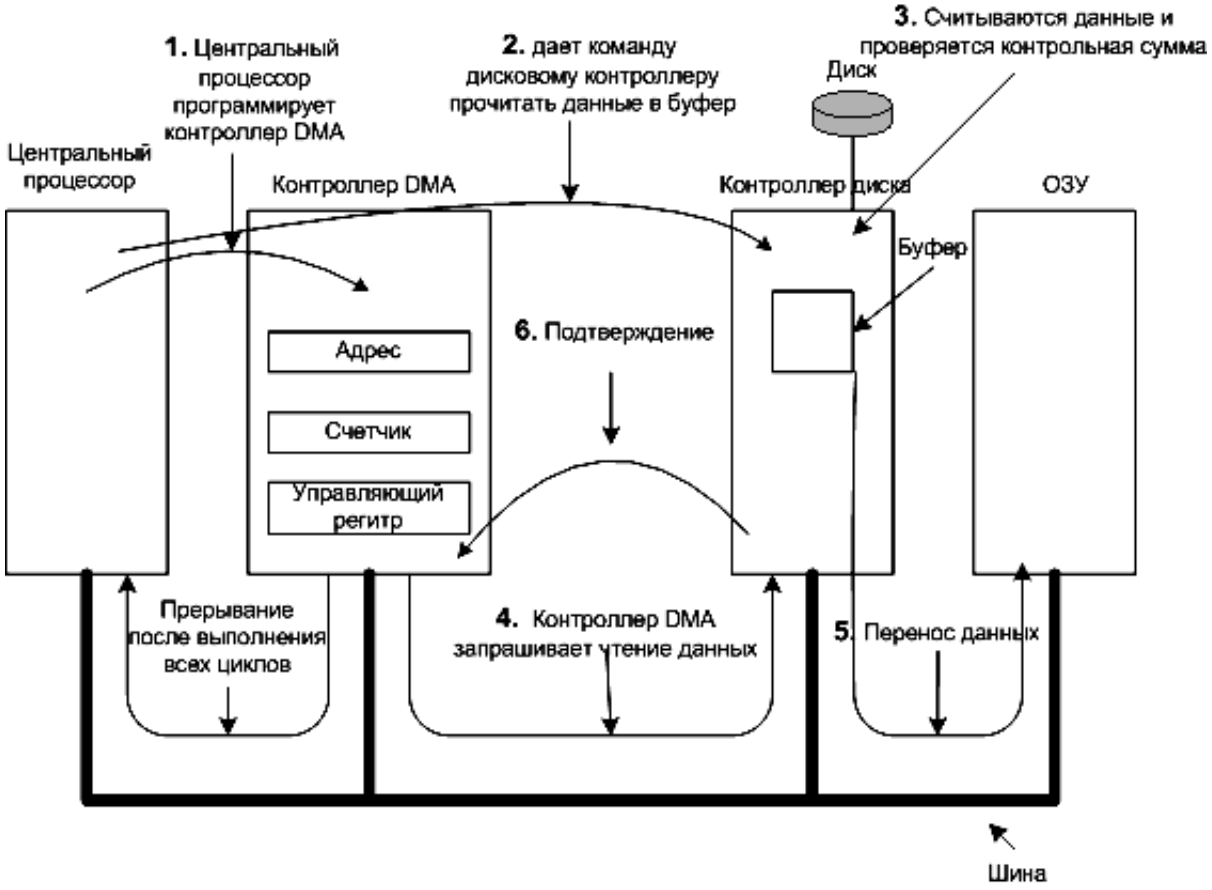


Рис. 6.5 – Работа DMA-контроллера

Контроллер содержит несколько регистров:

- регистр адреса памяти;
- счетчик байтов;
- управляющие регистры могут содержать:

  - порт ввода-вывода;
  - чтение или запись;
  - единицы переноса (побайтно или пословно).

При отсутствии контроллера происходит следующее:

1. Процессор дает команду дисковому контроллеру прочитать данные в буфер.

2. Считываются данные в буфер, контроллер проверяет контрольную сумму считанных данных (проверка на ошибки). Процессор, до прерывания, переключается на другие задания.

3. Контроллер диска инициирует прерывание.

4. Операционная система начинает работать и может считывать из буфера данные в память.

Функционирование контроллера:

1. Процессор программирует контроллер (какие данные и куда переместить).

2. Процессор дает команду дисковому контроллеру прочитать данные в буфер.

3. Считываются данные в буфер, контроллер диска проверяет контрольную сумму считанных данных (процессор, до прерывания, переключается на другие задания).

4. Контроллер DMA посылает запрос на чтение дисковому контроллеру.

5. Контроллер диска поставляет данные на шину, адрес памяти уже находится на шине, происходит запись данных в память.

6. Когда запись закончена, контроллер диска посылает подтверждение DMA-контроллеру.

7. DMA-контроллер увеличивает используемый адрес и уменьшает значение счетчика байтов.

8. Все повторяется с пункта 4, пока значение счетчика не станет равным нулю.

9. Контроллер DMA инициирует прерывание.

При одновременном использовании процессором и контроллерами ПДП шины для доступа к основной памяти возникает конфликт. Для разрешения конфликтных ситуаций применяется процедура выбора, реализуемая схемами управления шиной и устройствами, запрашивающими операции с памятью.

Два подхода к разрешению конфликтов на шине:

- централизованный арбитраж,
- распределенный арбитраж.

При централизованном подходе разрешение конфликтов выполняется арбитром шины, а при распределенном подходе выбор следующего владельца шины производится с участием всех

устройств. Арбитром шины может служить как процессор, так и какое-либо отдельное устройство, подключенное к шине.

Простая реализация арбитража – это использование гирляндной цепи. Когда линия запроса шины активизируется, процессор передает сигнал предоставления шины, указывающий контроллерам ПДП, что, как только шина освободится, они смогут ее использовать. Линия, по которой передается этот сигнал, соединяется со всеми контроллерами ПДП по принципу гирляндной цепи. Таким образом, если первый контроллер ПДП запросит управление шиной, он заблокирует передачу сигнала ее предоставления остальным устройствам. Если же шина ему не нужна, он передаст этот сигнал дальше, то есть на его выход будет подан сигнал.

При распределенном арбитраже все устройства, ожидающие своей очереди на использование шины, на равных правах участвуют в арбитражном процессе. Каждому соединенному с шиной устройству назначается 4-разрядный идентификационный номер. Когда одно или несколько устройств запрашивают шину, они активизируют сигнал начала арбитража и помещают свои 4-разрядные идентификационные номера на линии. Победитель определяется в результате обработки сигналов, переданных по линиям всеми претендентами. Выходом этой сети является код на четырех линиях, представляющий запрос с наивысшим идентификационным номером.

Программное обеспечения ввода-вывода

Основные функции:

- Независимость от устройств.
- Единообразное именование – имя файла или устройства не должны отличаться.
- Обработка ошибок – ошибки обрабатываются на уровне контроллера, драйвера и т.д.
- Перенос данных – синхронный и асинхронный (в последнем случае процессор запускает перенос данных и переключается на другие задачи до прерывания).
- Буферизация.

- Выделенные (принтер) и невыделенные (диск) устройства – принтер должен предоставляться только одному пользователю, а диск – многим. ОС должна решать все возникающие проблемы.

Программный ввод-вывод

В этом случае всю работу выполняет центральный процессор.

Процесс ввода-вывода на примере печати строки ABCDEFGH.

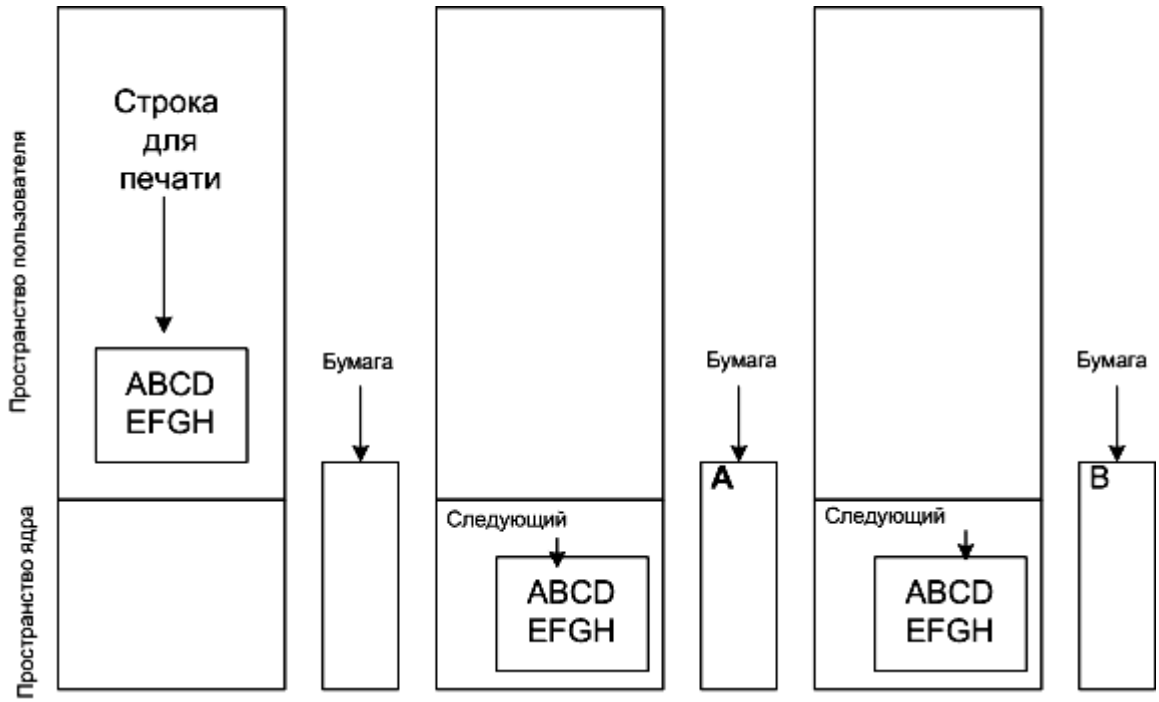


Рис. 6.6 – Вывод строки на печать

Алгоритм печати:

1. Строка для печати собирается в пространстве пользователя.
2. Обращаясь к системному вызову, процесс получает принтер.
3. Обращаясь к системному вызову, процесс просит распечатать строку на принтере.
4. Операционная система копирует строку в массив, расположенный в режиме ядра.
5. ОС копирует первый символ в регистр данных принтера, который отображен на памяти.
6. Символ печатается на бумаге.

7. Указатель устанавливается на следующий символ.

8. Процессор ждет, когда бит готовности принтера выставится в готовность.

9. На первый пункт.

При использовании буфера принтера сначала вся строка копируется в буфер, после этого начинается печать.

6.7 Глава 7. Шины и интерфейсы

В основу построения современных ЭВМ положен магистрально-модульный принцип.

Основные положения:

- устройства ЭВМ (процессоры, ЗУ, контроллеры ПУ) оформляются в виде модулей, совместимых на конструктивном, электрическом и функциональном уровнях;

- объединение модулей в систему осуществляется на основе магистралей (концепция открытых систем);

- объединение модулей в систему осуществляется по определенным правилам сопряжения, называемым интерфейсами.

Интерфейс – это совокупность аппаратных и программных средств, реализующих стандарт по организации связей в магистрально-модульной системе.

Классификация интерфейсов.

1. По способу соединения модулей в структуру интерфейсы различают:

- магистральные – одна общая шина используется для объединения модулей и обмена информацией;
- радиальные;
- цепочечные;
- смешанные (комбинированные).

2. По способу передачи информации: параллельные, последовательные и параллельно-последовательные.

3. По принципу обмена: синхронные и асинхронные.

4. По режиму передачи информации:

- симплексный режим – передача только в одном направлении;

- дуплексный режим – двусторонняя одновременная передача;
- полудуплексный режим – двусторонняя передача, но в разные моменты времени – с разделением (мультиплексированием) по времени.

Основные элементы интерфейса:

- совокупность правил обмена (протокол обмена);
- аппаратная часть интерфейса;
- программное обеспечение интерфейса (алгоритм управления обменом, реализующий протокол обмена).

Современные интерфейсы обычно делятся на три класса: параллельные, последовательные и связные.

Параллельный интерфейс обеспечивает быструю передачу n-разрядного двоичного кода.

Электрические цепи интерфейса в зависимости от их назначения принято разделять на две основные группы: информационные и управления. Информационные шины ШИ используются для передачи в разные моменты времени данных, команд и адресов.

Цепи, образующие шину управления ШУ, используются для передачи различных сигналов управления.

Для передачи информации по шинам используется два способа: синхронный или асинхронный.

При **синхронном** способе управление передачей осуществляется сигналом синхронизации (стробом), который вырабатывается передающим устройством и подается в приемное устройство, где используется для приема информации (рисунок 7.1).

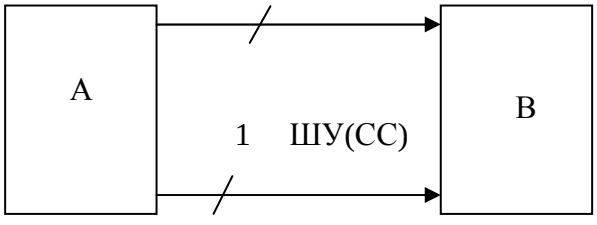


Рис. 7.1 – Схема соединения

Достоинство – простота.

Недостатки:

- потери времени для тех пар устройств, для которых фактическое время передачи меньше T_1;
- низкая надежность передачи, так как нет сигнала, что приемное устройство приняло информацию по СС.

В связи с этими недостатками синхронный способ, как правило, применяется только в параллельных интерфейсах при передаче на небольшие расстояния при условии, что разброс времени приема незначителен, а надежность приемных устройств высокая, т.е. обычно в пределах одной интегральной схемы (кристалла процессора, например).

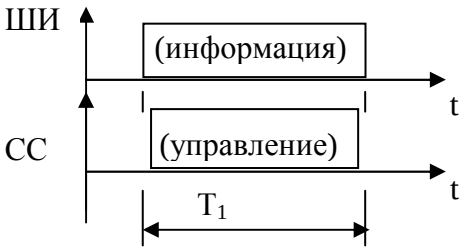


Рис. 7.2 – Диаграммы сигналов

Асинхронный способ (типа «запрос-ответ»). При этом способе приемное устройство, принявшее информацию по сигналу синхронизации СС, вырабатывает ответный сигнал, подтверждающий факт приема информации с ШИ.

Схема соединения устройств и временная диаграмма в этом случае имеют вид, представленный на рисунке 7.3.

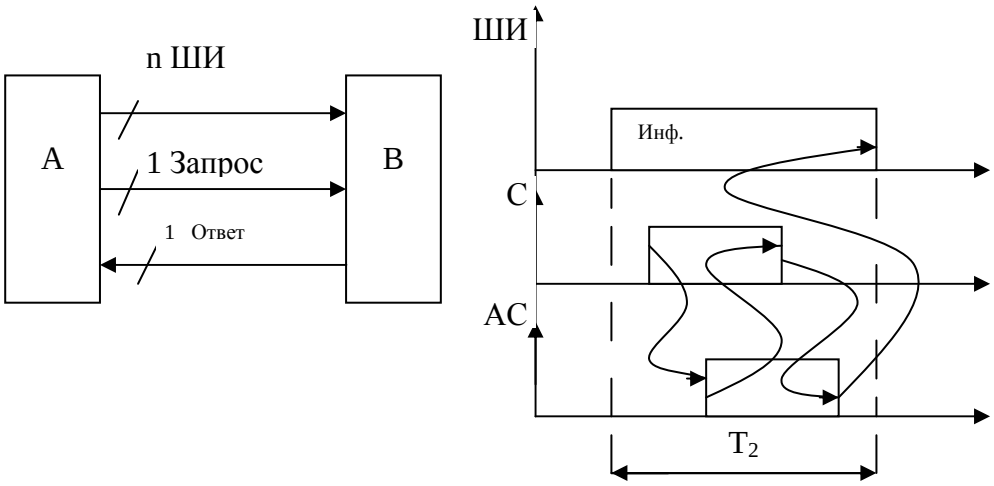


Рис. 7.3 – Схема с ответом и диаграммы

Время передачи T_2 является переменным, так как зависит от конкретных (фактических, а не максимальных) параметров: длины цепей интерфейса между устройствами А и В, времени реакции устройств, участвующих в обмене.

Область применения асинхронных интерфейсов – подключение устройств различного быстродействия, например периферийных устройств (рис. 7.4).

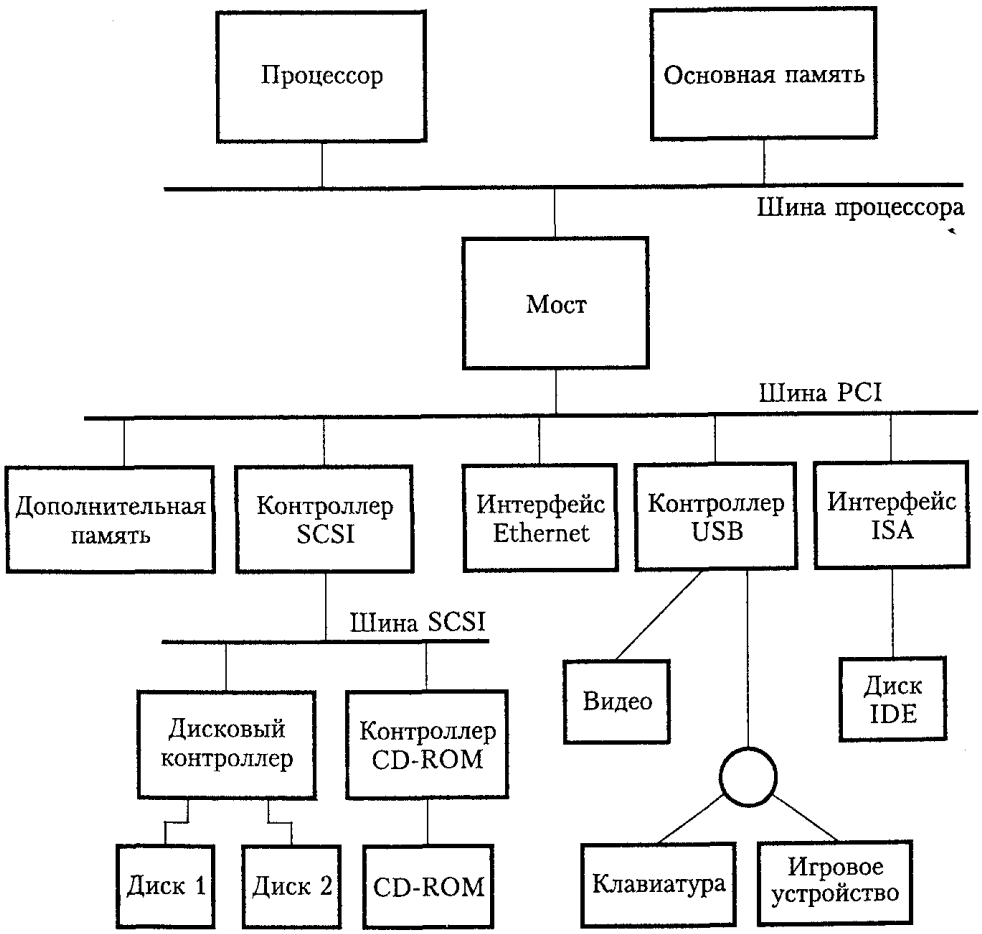


Рис. 7.4 – Пример компьютерной системы, в которой используется несколько стандартов интерфейса

Последовательный интерфейс состоит обычно из одной цепи (точнее – пары цепей – витой пары), данные по которой передаются бит за битом, то есть в последовательном коде.

Связные интерфейсы обеспечивают передачу информации по каналам связи на большие расстояния: от 1 км и выше – до сотен, тысяч км. Однако скорость передачи в них, как и в последовательных интерфейсах, невелика.

В вычислительной технике последовательные и связные интерфейсы используются для подключения удаленных ПУ, а также для связи между машинами, входящими в состав сети ЭВМ (локальной или глобальной).

На рис. 7.5 представлена универсальная схема параллельного интерфейса, которую можно конфигурировать множеством способов. Линии от Р7 до РО могут использоваться как для ввода, так и для вывода данных. С целью обеспечения большей гибко-

сти схема позволяет использовать часть линий только для ввода, а остальные линии – только для вывода. Выбор линий осуществляется программным путем. Регистр DATAOUT соединяется с указанными линиями через повторители с тремя состояниями, которые управляются регистром, определяющим направление передачи данных DDR (Data Direction Register).

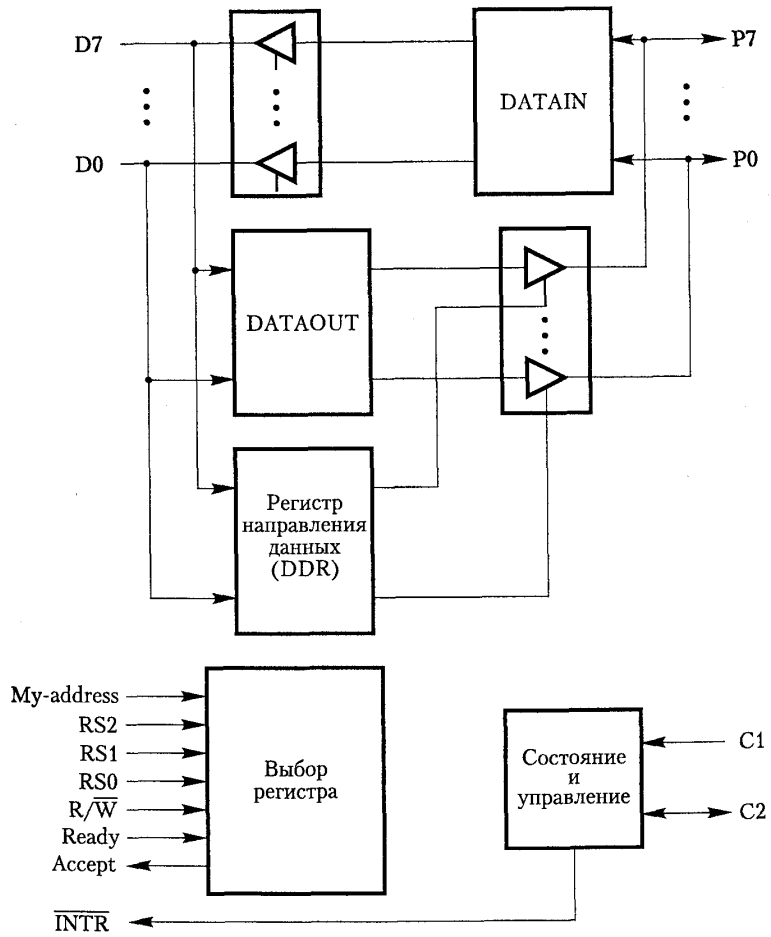


Рис. 7.5 – Универсальный 8-разрядный параллельный интерфейс

Процессор может записать в регистр DDR 8-разрядную маску. Если некоторый разряд этого регистра содержит значение 1, линия данных функционирует как выходная; в противном случае – как входная.

Две линии, C1 и C2, управляют взаимодействием интерфейсной схемы и подключенного к ней устройства ввода-вывода. Эти линии также являются программируемыми. Двунаправлен-

ная линия C2 поддерживает несколько сигнальных режимов, включая и режим квитирования.

Линии Ready и Accept используются для квитирования со стороны процессора, и их можно соединить с линиями Master-ready и Slave-ready. Входной сигнал My-address должен быть соединен с выходом декодера, который распознает адрес, назначенный данному интерфейсу. Существуют три линии выбора регистра, позволяющие адресовать до восьми регистров интерфейса: регистры входных и выходных данных, регистр направления данных, управляющий регистр и регистр состояния для различных режимов функционирования схемы. Кроме того, имеется выход запроса прерывания INTR, который должен быть соединен с линией запроса прерывания шины компьютера.

Схемы параллельного интерфейса, подобные показанной выше, встречаются довольно часто. Вместо одного порта для подключения устройства ввода-вывода такая схема может включать два и более портов.

Последовательный порт используется для соединения процессора с устройствами ввода-вывода, которые передают данные по одному биту за раз. Важной особенностью интерфейсной схемы последовательного порта является то, что она способна передавать данные в последовательном режиме со стороны устройства в параллельном режиме со стороны шины. Взаимопреобразование последовательных и параллельных форматов данных выполняется при помощи сдвиговых регистров, обладающих функцией параллельного доступа. На рис. 7.6 приведена структурная схема типичного последовательного интерфейса, включающая регистры DATAIN и DATAOUT. Входной сдвиговый регистр принимает от устройства ввода-вывода последовательные биты. После получения всех 8 бит данных содержимое этого регистра в параллельном режиме загружается в регистр DATAIN. Аналогичным образом выходные данные из регистра DATAOUT загружаются в выходной сдвиговый регистр, откуда биты по очереди отправляются к устройству ввода-вывода.

Флаг SIN устанавливается в 1, когда новые данные загружаются в регистр DATAIN, а когда процессор считывает содержимое этого регистра, значение флага SIN сбрасывается в 0. Как только данные пересылаются из входного сдвигового регистра в

DATAIN, сдвиговый регистр может начать прием от устройства ввода-вывода следующего 8-битового символа. Флаг указывает, доступен ли выходной буфер для записи. Когда процессор записывает в DATAOUT новые данные, этот флаг очищается, а когда данные из DA1 Аи перемещаются в сдвиговый регистр, SOUT устанавливается в 1.

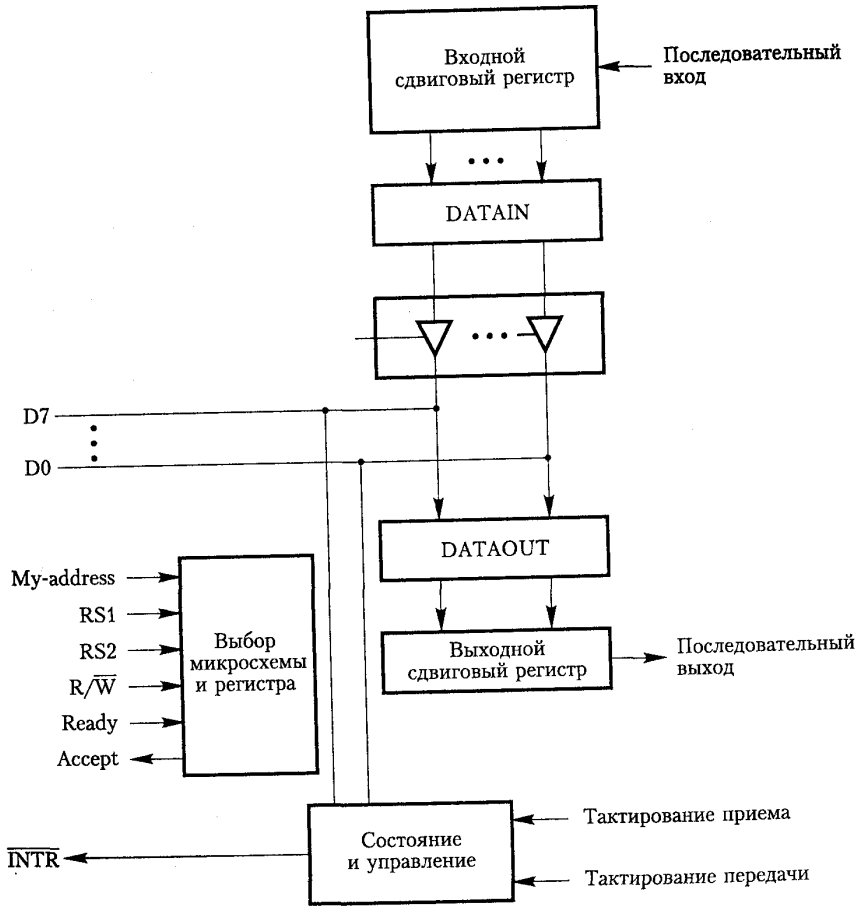


Рис. 7.6 – Последовательный интерфейс

Для последовательной передачи данных требуется меньшее количество проводов, она удобна для применения в устройствах, которые физически находятся на значительном расстоянии от компьютера. Скорость пересылки данных, часто измеряемая в битах, зависит от конкретного устройства. Для того чтобы последовательный интерфейс мог работать с различными устройствами, он должен поддерживать разную тактовую частоту.

Последовательный интерфейс используется для подключения к компьютерам большого количества устройств ввода-вывода, для него разработано несколько популярных стандартов. UART (Universal Asynchronous Receiver Transmitter) – универсальный асинхронный приемопередатчик, предназначенный для использования асинхронного стартстопного режима.

Для коммуникационных соединений разработан другой популярный стандарт, называемый RS-232-C.

Процессор, основная память и устройства ввода-вывода могут соединяться между собой посредством шины, основным назначением которой является предоставление канала связи для пересылки данных.

Пересылка данных по шине подчиняется определенному набору правил, называемых шинным протоколом, управляющих поведением соединенных с шиной устройств, а также последовательностью выдачи информации на шину, управляющих сигналов и т.п.

Линии шины, используемые для пересылки данных, бывают трех типов: линии данных, линии адреса и управляющие линии. Управляющие сигналы определяют, какую операцию, чтения или записи, следует выполнить. Обычно для этой цели используется линия $R/\overline{W}$. Значение 1 на этой линии соответствует операции чтения, а значение 0 – операции записи. Когда команда допускает использование операндов разных размеров, например, байтов, слов и длинных слов, размер данных также указывается на управляющих линиях.

Сигналы управления шиной также используются для тактирования операций. Они определяют, в какой момент процессор и устройства ввода-вывода могут поместить данные на шину или прочитать их с таковой. Для тактирования пересылки данных по шине разработано множество схем, которые можно разделить на два основных типа: синхронные и асинхронные.

В любой операции пересылки данных по шине одно из устройств играет роль хозяина шины. Это устройство инициирует пересылку данных с помощью команд чтения или записи. Поэтому его можно назвать инициатором. Обычно хозяином шины является процессор, но эту роль могут выполнять и другие устрой-

ства, поддерживающие функцию прямого доступа к памяти. Устройство, к которому обращается хозяин шины, называется подчиненным или целевым.

В случае синхронной шины все устройства получают синхронизирующую информацию по общей тактовой линии. На эту линию подаются тактовые импульсы со строго фиксированной частотой. Промежуток времени между последовательными тактовыми импульсами в простейшей синхронной шине составляет цикл шины, в течение которого выполняется одна операция пересылки данных.

Последовательность событий, происходящих при выполнении операции ввода (чтения) данных. В момент времени t_0 хозяин шины помещает на адресные линии адрес устройства и отсылает по управляющим линиям необходимую команду. В этой команде определяется операция ввода и, если нужно, задается длина считываемого операнда. Информация передается по шине со скоростью, определяемой ее физическими и электрическими характеристиками. Длительность тактового импульса $t_1 - t_0$ должна быть больше максимального времени задержки на распространение сигнала между двумя соединенными с шиной устройствами. Причем она должна быть достаточно большой, так как все устройства должны успеть декодировать адрес и управляющие сигналы с тем, чтобы адресуемое (подчиненное) устройство могло ответить на команду в момент времени t_1. В течение промежутка времени от t_0 до t_1 информация на шине ненадежна, поскольку состояние сигналов изменяется. В момент времени t_1 адресуемое подчиненное устройство помещает запрошенные входные данные на линии данных.

В конце тактового цикла, то есть в момент времени t_2, хозяин шины стробирует (сохраняет в буфере в указанный момент времени) данные на линиях данных в свой входной буфер. Для правильной загрузки данных в любое устройство хранения, в том числе в регистр на основе триггеров, они должны находиться на его входе в течение времени, достаточного для их сохранения. Поэтому период времени $t_2 - t_1$ должен быть больше максимального времени распространения сигнала по шине в сумме со временем установки входного буферного регистра хозяина шины.

Похожая процедура выполняется и при выводе данных. Хозяин шины помещает на линии данных выходные сведения, а на линии адреса и управляющие линии – адрес и команду. В момент времени t_2 адресуемое устройство стробирует линии данных и загружает информацию в свой буфер данных.

Поскольку на передачу сигнала от одного устройства к другому уходит некоторое время, разные устройства видят изменения этого сигнала в разные моменты. Одно представление соответствует тому, как данный сигнал видит хозяин шины, а другое – тому, как его видит подчиненное устройство. При этом предполагается, что изменения тактового сигнала все подключенные к шине устройства замечают одновременно.

В начале такта 1 (t_0), на переднем фронте тактового сигнала, хозяин шины передает по ней сигналы адреса и команды. Однако до момента времени t_{AM} эти сигналы не появляются на шине, прежде всего из-за задержки в схеме управления шиной. Какое-то время спустя, в момент t_{AS}, сигналы достигают подчиненного устройства, которое декодирует адрес и в момент времени t_1 отсылает запрошенные данные. Эти сигналы задерживаются до момента t_{DS}. К хозяину шины они прибывают в момент времени t_{DM}. Далее, в момент времени t_2, хозяин загружает данные в свой входной буфер. Промежуток $t_2 – t_{DM}$ является временем установки его входного буфера. После момента времени t_2 данные должны оставаться неизменными в течение всего периода их хранения в буфере.

Описанный способ пересылки данных имеет ряд ограничений:

- при пересылке данных за один такт период $t_2 –t_0$ больше задержки самого медленного интерфейса устройств. В результате все устройства будут работать со скоростью самого медленного из них;

- процессор не определяет устройство, ответившее на запрос, а только предполагает, что в момент времени t_2 выходные данные получены устройством ввода-вывода или что входные данные имеются на линиях данных. Если же по какой-либо причине устройство не ответит, процессор даже не обнаружит ошибку.

Для преодоления этих ограничений в большинство шин включают поддержку управляющих сигналов, передаваемых в качестве ответа устройства. Эти сигналы информируют хозяина шины о том, что подчиненное устройство распознало адрес и готово участвовать в операции пересылки данных. Кроме того, они позволяют откорректировать длительность периода пересылки данных в соответствии с требованиями участвующих в операции устройств. Для упрощения этой задачи используется тактовый сигнал высокой частоты, при котором цикл пересылки занимает несколько тактов. Таким образом, количество тактов, затрачиваемых на операцию пересылки данных, зависит от конкретной пары устройств.

Пример реализации такого подхода может быть следующим: В течение такта 1 хозяин шины пересылает адрес и информацию о команде, запрашивая операцию чтения. Подчиненное устройство получает и декодирует эту информацию. По следующему переднему фронту тактового сигнала, то есть в начале такта 2, устройство принимает решение ответить на запрос и начинает процедуру доступа к запрошенным данным. На извлечение данных требуется время, поэтому подчиненное устройство не может ответить немедленно. На третьем такте данные готовы и помещаются на шину. В то же время подчиненное устройство выдает управляющий сигнал, называемый Slave-ready (подчиненный готов). Хозяин шины, ожидавший этого сигнала, стробирует данные в свой входной буфер в конце такта 3. На этом операция пересылки данных по шине заканчивается, и хозяин шины может отправить по ней новый адрес, чтобы на такте 4 начать другую операцию пересылки.

Сигнал Slave-ready – это направляемое подчиненным устройством хозяину шины сообщение, говорящее о том, что им отправлены правильные данные. Подчиненное устройство отвечает во время такта 3. Другое устройство может ответить раньше или позже. Таким образом, сигнал Slave-ready позволяет изменять длительность операции пересылки данных в соответствии с возможностями конкретного устройства. Если адресуемое устройство не отвечает, хозяин шины ждет в течение заранее определенного количества тактов, а затем отменяет операцию. Отсутствие

ответа может быть результатом отправки неверного адреса или некорректного функционирования устройства.

Для тактирования шины не всегда используется тот же сигнал, что и для тактирования процессора. Последний обычно имеет значительно более высокую частоту, поскольку управляет внутренним функционированием микросхемы процессора. Задержки на распространение сигналов внутри процессора меньше, чем задержки на шине, связывающей, в частности, микросхемы на печатных платах. То, какая тактовая частота может использоваться в конкретном компьютере, зависит от технологии его производства. Для современных процессорных микросхем типична тактовая частота свыше 500 МГц, для памяти и шин ввода-вывода может использоваться частота от 50 до 150 МГц.

Альтернативная схема управления пересылкой данных по шине основывается на механизме квитирования, то есть подтверждения связи, между хозяином шины и подчиненным устройством. Концепция квитирования является обобщением идеи использования сигнала Slave-ready. В схеме с квитированием тактовая линия заменяется двумя управляющими линиями синхронизации: Master-ready и Slave-ready. Первая принадлежит хозяину шины, который передает по ней сигнал готовности к транзакции, а по второй отвечает подчиненное устройство.

Хозяин шины помещает на нее адрес и информацию о команде. Затем по линии Master-ready он сообщает об этом всем устройствам. В ответ подключенные к шине устройства декодируют адрес. То устройство, для которого предназначена команда, выполняет таковую и информирует об этом хозяина шины по линии Slave-ready. Хозяин дожидается этого сигнала и только после этого удаляет с шины свои сигналы. В случае операции чтения он стробирует данные в свой входной буфер.

Временная последовательность событий (рис. 7.7):

t_0 – хозяин шины помещает на нее адрес и команду, и все устройства на шине начинают декодировать эту информацию.

t_1 – хозяин шины активизирует линию Master-ready, чтобы проинформировать устройства ввода-вывода о том, что адрес и команда поданы на шину. Задержка, равная $t_1 - t_0$, формируется на тот случай, если на шине произойдет сдвиг сигналов. Сдвигом называется неодновременное прибытие в точку назначения двух

сигналов, одновременно выданных источником. Причиной такого сдвига может стать разное время распространения сигнала по различным линиям шины. Для того чтобы иметь гарантию, что сигнал Master-ready не достигнет какого-либо устройства раньше адреса и команды, нужно задержать его на время $t_1 - t_0$, которое больше времени максимально возможного сдвига. (В случае синхронной шины сдвиг на шине тоже учитывается – он является составной частью максимальной задержки на распространение сигнала.) Когда адресная информация достигает очередного устройства, она декодируется его интерфейсной схемой. На декодирование тоже уходит какое-то время, которое следует включить в период $t_1 - t_0$.

t_2 – после декодирования адреса и команды подчиненное устройство выполняет операцию ввода, то есть перемещает информацию из своего регистра данных на линии данных. Одновременно оно активизирует сигнал Slave-ready. Если интерфейсная схема помещает данные на шину с некоторой задержкой, подчиненное устройство должно соответственно задержать и выдачу сигнала Slave-ready. Длительность промежутка времени $t_2 - t_1$ зависит от расстояния между хозяином шины и подчиненным устройством, а также от задержек в схеме подчиненного устройства. Указанный промежуток времени имеет разную длительность, схема получается асинхронной.

t_3 – сигнал Slave-ready достигает хозяина шины и сообщает ему, что на шине имеются данные. Предполагается, что интерфейс устройства помещает на шину сигнал Slave-ready одновременно с данными, хозяин шины должен выдержать небольшую паузу на случай сдвига сигналов. Кроме того, он должен учесть время установки собственного входного буфера. После задержки, равной сумме времени максимального сдвига сигналов на шине и минимального времени установки буфера, хозяин шины стробирует данные в свой входной буфер. Одновременно он удаляет с шины сигнал Master-ready, сообщая тем самым о получении данных.

t_4 – хозяин шины удаляет с шины адрес и команду. Задержка между моментами времени t_3 и t_4 производится на случай сдвига на шине. Если адрес, который видит какое-либо из устройств,

начнет изменяться, пока сигнал Master-ready равен 1, адресация будет выполнена неточно.

t_5 – когда интерфейс устройства фиксирует переход сигнала Master-ready из 1 в 0, он удаляет с шины данные и сигнал Slave-ready. На этом пересылка входных данных завершается.

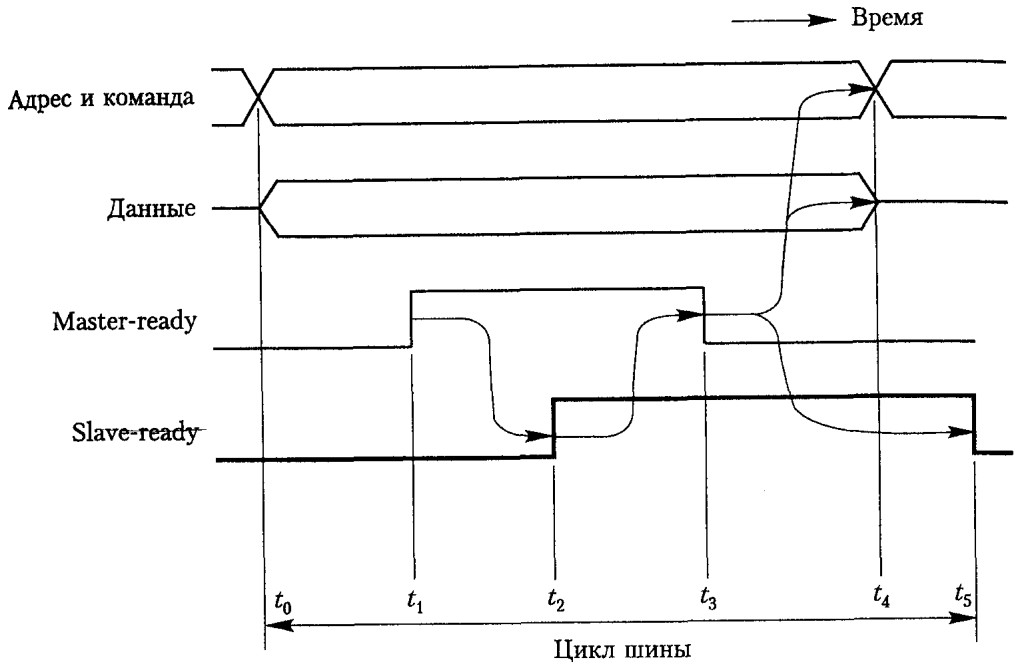


Рис. 7.7 – Пересылка выходных данных по шине с использованием квитирования

Процесс выполнения операции вывода аналогичен процессу выполнения операции ввода. Хозяин шины одновременно с адресом и командой помещает на линии данных выходную информацию. Получив сигнал Master-ready, подчиненное устройство стробирует данные в свой входной буфер и сообщает об этом установкой сигнала Slave-ready в 1. Дальше все происходит точно так, как при вводе данных.

Хозяин шины компенсирует сдвиг на шине и задержку на декодирование адреса. Для этой цели предназначены задержки $t_1 - t_0$ и $t_4 - t_3$. Если длительность задержки достаточна для декодирования адреса интерфейсом устройства ввода-вывода, интерфейсная схема может использовать сигнал Master-ready для пропуска других сигналов на шину и с шины.

Сигналы квитирования в приведенных схемах взаимосвязаны таким образом, что за изменением одного сигнала следует изменение другого. Поэтому такая схема, называемая полным квитированием, обладает исключительной гибкостью и надежностью.

Важнейшим преимуществом асинхронной шины является то, что процесс квитирования избавляет конструктора от необходимости синхронизации тактовых сигналов отправителя и получателя, что значительно упрощает разработку. Никакие задержки, связанные с распространением сигнала по шине или с интерфейсными схемами, не отражаются на работе системы. Когда величина задержек изменяется, например, из-за изменения нагрузки при добавлении или удалении интерфейсной схемы, автоматически изменяется и время передачи данных. Для синхронных шин схемы тактирования должны разрабатываться очень тщательно, что обеспечит правильную синхронизацию, а задержки не должны превышать строго рассчитанный предел.

Скорость передачи данных по асинхронной шине, управляемой посредством механизма полного квитирования, несколько снижается из-за того, что каждая такая операция выполняется с четырьмя задержками, по две в каждую сторону. В случае синхронной шины тактовый период должен включать задержку на распространение сигнала только в одном направлении. Благодаря этому пересылка выполняется быстрее. Для более медленных устройств используются дополнительные такты. Поэтому высокоскоростные шины большинства современных компьютеров имеют синхронную архитектуру.

Шина PCI – это системная шина, появившая в процессе стандартизации для высокоскоростных дисковых и графических устройств. Она поддерживает функции, типичные для шины процессора, но в стандартизированном формате, независимо от типа процессора. Подключенные к этой шине устройства представляются процессору непосредственно соединенными с его собственной шиной. Им назначаются адреса из адресного пространства памяти процессора.

В современных компьютерах при выполнении операции пересылки данных перемещаются блоки информации. Поэтому необходима кэш-память. Данные пересылаются между кэш-

памятью и основной памятью в виде пакетов, по несколько слов в каждом. Участвующие в такой пересылке слова сохраняются по последовательным адресам памяти. Когда процессор, а точнее кэш-контроллер, задает адрес и запрашивает операцию чтения из основной памяти, память отвечает ему отправкой последовательности слов данных, начинающихся с этого адреса. Аналогичным образом в ходе операции записи процессор задает адрес и последовательность слов данных, которые должны быть поочередно записаны в память, начиная с этого адреса. Шина PCI предназначена для поддержки режима работы чтения или записи пакетов длиной в одно слово.

Шина поддерживает три независимых адресных пространства: памяти, ввода-вывода и конфигурации. Адресное пространство ввода-вывода используется такими процессорами, как Pentium, имеющими отдельное адресное пространство ввода-вывода. Стандартом PCI для совместимости с более широким спектром устройств рекомендуется применять именно этот подход. Конфигурационное адресное пространство предназначено для поддержки технологии plug-and-play. Сопровождающая адрес 4-разрядная команда указывает, какое из трех адресных пространств должно использоваться в этой операции пересылки данных.

На рис. 7.8 показана схема, на которой основная память компьютера непосредственно соединена с шиной процессора. Мост PCI создает для основной памяти отдельное физическое подключение. По причинам электротехнического характера шина может быть разделена на сегменты, соединенные между собой посредством мостов. Однако независимо от того, к какому сегменту шины подключено конкретное устройство, оно может отображаться в адресное пространство процессора.

Сохранение адресной информации на шине необходимо пока не выбрано подчиненное устройство, сохраняющее адрес в своем внутреннем буфере, тогда адрес должен находиться на шине в течение одного такта, после чего адресные линии могут быть освобождены для пересылки данных в последующих тактах. Благодаря этому снижается стоимость операции пересылки, напрямую зависящая от количества проводов шины.

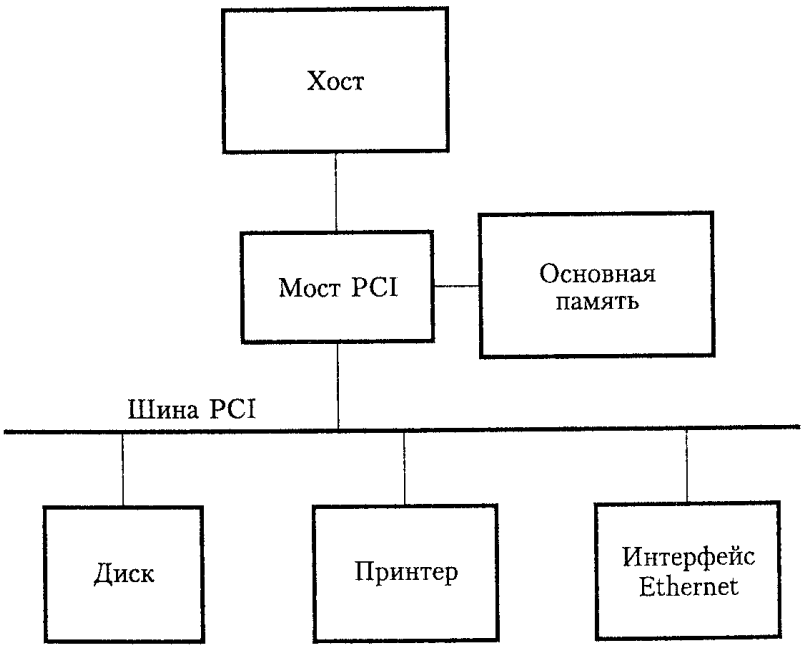


Рис. 7.8 – Пример использования шины PCI в компьютерной системе

Хозяином шины в каждый конкретный момент времени может быть только одно устройство. Это устройство имеет право инициировать операции пересылки данных с помощью команд чтения и записи. Согласно терминологии PCI хозяин шины называется инициатором. Им может быть либо процессор, либо контроллер ПДП. Адресуемое устройство, отвечающее на команды чтения и записи, называется целевым.

Основные сигналы шины, используемые для пересылки данных, перечислены в табл. 1. Для сигналов, имена которых начинаются с символа «#», активным является низковольтное состояние.

Таблица 1 – Сигналы пересылки данных по шине PCI

Имя	**Описание**
CLK	Тактовый сигнал с частотой 33 или 66 МГц
FRAME#	Активизируется инициатором с целью определения длительности транзакции
AD	Представляет 32 линии для пересылки адресов и данных (количество линий при необходимости может быть увеличено до 64)

Окончание табл. 1

Имя	Описание
C/BE#	Представляет 4 линии для команды считывания (Command) и массив, указывающий, какие байты подлежат считыванию (Byte enable)
IRDY#,	Сигналы готовности инициатора и целевого устройства
DEVSEL#	Ответ устройства, указывающий, что оно распознало свой адрес и готово к операции пересылки данных
IDSEL#	Сигнал выбора инициализируемого устройства

Полная операция пересылки данных по шине – это транзакция. Пересылка отдельного слова в ходе транзакции называется фазой.

SCSI (Small Computer Systems Interface) – означает интерфейс малых компьютерных систем. Стандарт определен Национальным институтом стандартизации США (American National Standards Institute, ANSI) под номером X3.131. Согласно основной спецификации этого стандарта такие устройства, как диски, должны соединяться с компьютером при помощи 50-проводного кабеля длиной до 25 м, по которому данные могут передаваться со скоростью до 5 Мбайт/с.

Стандарты SCSI-2 и SCSI-3, каждый из которых имеет несколько опций. Узкая (narrow SCSI) шина SCSI имеет восемь линий данных и передает данные по одному байту. Широкая шина SCSI (wide SCSI) состоит из 16 линий данных и передает информацию по 16 бит. Существует несколько вариантов электрических сигнальных схем. Передача данных по шине SCSI выполняется в асимметричном режиме (Single-Ended SCSI, SE) при использовании для каждого сигнала проводника с общим замыканием через «землю» для всех сигналов. Для каждого сигнала также может предназначаться отдельный обратный провод. В этом случае возможно использование двух уровней напряжения. Применяется уровень напряжения 3,3 В.

Для различных версий SCSI используются разъемы: 50-, 68- и 80-контактные. Максимальная скорость передачи данных современных устройств варьируется от 5 до 60 Мбайт/с. Последняя версия стандарта поддерживает скорость передачи до 320 Мбайт/с, а на подходе версия с поддержкой 640 Мбайт/с. Максимальная скорость передачи по конкретной шине зависит от дли-

ны кабеля и количества подключенных к нему устройств. Для достижения максимальной скорости обычно используют кабель длиной не более 1,6 м для сигнальной схемы SE и не более 12 м для сигнальной схемы LVD. Для подключения удаленных устройств предоставляют специальные расширители шины. Максимальная «вместимость» шины составляет 8 устройств для Narrow SCSI и 16 устройств для Wide SCSI.

Шина SCSI соединяется с шиной процессора через SCSI-контроллер. Для пересылки пакетов данных от главной памяти к устройству и в обратном направлении применяется технология прямого доступа к памяти. Пакет может содержать блок данных, команды, направляемые процессором устройству, или информацию о состоянии устройства.

Принцип взаимодействия с дисками отличается от принципа взаимодействия с основной памятью. Данные хранятся на диске блоками, называемыми секторами, каждый из которых может содержать несколько сот байтов. Данные не всегда записываются в последовательно расположенные секторы, потому что в одних секторах могут уже храниться ранее записанные данные; другие могут быть дефектными, а следовательно, должны быть пропущены. Поэтому для обслуживания запроса чтения или записи может потребоваться доступ к обязательно последовательным секторам диска.

Протокол SCSI ориентирован на режим работы с задержками из-за механической природы диска. Обращение к первому сектору, из которого учитываются или в который записываются данные, выполняется с довольно значительной задержкой, порядка нескольких миллисекунд. После этого некоторый объем данных пересылается с очень высокой скоростью, но затем может произойти еще одна задержка и т. д. В ходе обслуживания одного запроса чтения или записи может произойти несколько таких задержек.

Контроллер, подключенный к шине SCSI, может быть инициатором или целевым устройством. Инициатор обладает способностью выбирать конкретное целевое устройство и направлять ему команды, определяющие выполняемую операцию. Контроллер со стороны процессора функционирует как инициатор. Дисковый контроллер является целевым устройством и выполня-

ет команды, получаемые от инициатора. Инициатор устанавливает логическое соединение с выбранным целевым устройством. Соединение может временно разрываться и возобновляться по мере возникновения необходимости в пересылке команд и пакетов данных. При временном разрыве соединения шина используется другими устройствами. Эта способность чередовать запросы пересылки данных, определяет высокую производительность шины.

Пересылка данных по шине SCSI управляется контроллером. Для отправки ему команды, инициатор запрашивает управление шиной, выиграв арбитраж, выбирает контроллер, с которым хочет взаимодействовать, и передает ему управление шиной. После этого целевой контроллер начинает операцию передачи данных для получения команды от инициатора.

Процессор направляет SCSI-контроллеру команду, в ответ на которую происходят следующие события:

1. Контроллер SCSI как инициатор запрашивает управление шиной.

2. Выиграв арбитраж, он выбирает целевой контроллер и передает ему управление шиной.

3. Целевой контроллер начинает операцию вывода, а инициатор направляет ему в ответ команду, определяющую операцию чтения.

4. Целевой контроллер, который вначале должен выполнить операцию поиска данных на диске, отсылает инициатору сообщение, указывающее, что он временно разрывает соединение. После этого он освобождает шину.

5. Целевой контроллер направляет диску команду переместить считывающую головку к первому сектору, содержащему запрошенные данные. Затем он считывает хранящиеся в этом секторе данные и сохраняет их в буфере данных. Когда контроллер готов начать пересылку данных инициатору, он запрашивает управление шиной. Выиграв арбитраж, он снова выбирает инициирующий контроллер, возобновляя тем самым временно разорванное соединение.

6. Целевой контроллер пересылает инициатору содержимое буфера данных и еще раз временно разрывает соединение. Дан-

ные пересылаются параллельно по 8 или 16 бит, в зависимости от ширины шины.

7. Целевой контроллер направляет диску команду выполнить еще одну операцию поиска. Затем он пересылает инициатору содержимое второго сектора диска. Когда пересылка завершается, логическое соединение между двумя контроллерами разрывается.

8. Получив данные, инициирующий контроллер сохраняет их в основной памяти с использованием ПДП.

9. Контроллер SCSI направляет процессору запрос прерывания, для того чтобы проинформировать его о завершении операции.

Шина USB

Современные компьютерные системы включают множество устройств: клавиатуры, микрофоны, цифровые видеокамеры, динамики, дисплеи. Для их подключения применяется универсальная последовательная шина – Universal Serial Bus (USB), являющаяся промышленным стандартом. USB поддерживает два режима функционирования, получивших названия низкоскоростной (1,5 Мбит/с) и полноскоростной (12 Мбит/с). В последней версии этой спецификации, USB 2.0 введен третий режим, названный высокоскоростным (480 Мбит/с).

В USB реализованы следующие преимущества:

- простая, дешевая и удобная система соединения, позволяющая преодолевать сложности, возникающие из-за ограниченного числа имеющихся в компьютерах портов ввода-вывода;
- обеспечен широкий диапазон параметров пересылаемых данных, присущих различным устройствам ввода-вывода, в том числе модемам (скорость, объемы и временные характеристики процесса обмена данными);
- облегчен процесс подключения устройств за счет поддержки режима plug-and-play.

Система обнаруживает новое устройство, идентифицируеит его и соответствующее ему программное обеспечение (драйвер, а также другие необходимые для его работы средства), назначает

адреса и устанавливает логические соединения для взаимодействия с другими устройствами.

Для шины USB выбран последовательный формат пересылки данных. Тактирующий сигнал и данные кодируются вместе и передаются как единый сигнал. Шина USB поддерживает три скорости пересылки данных, а именно, 1,5; 12 и 480 Мбит/с. Шина имеет древовидную структуру, (рис. 7.9). В узлах дерева располагаются хабы. Корневой хаб соединяет все дерево с хост-компьютером. Листьями дерева являются устройства ввода-вывода. Это – функции.

Древовидная структура обеспечивает соединения «точка-точка». Каждый хаб имеет ряд портов. В нормальном режиме хаб копирует полученное входное сообщение в свои выходные порты, на которое отвечает только адресуемое устройство. Сообщение от устройства ввода-вывода пересылается только вверх, в направлении корневого узла, и другие устройства его не получают.

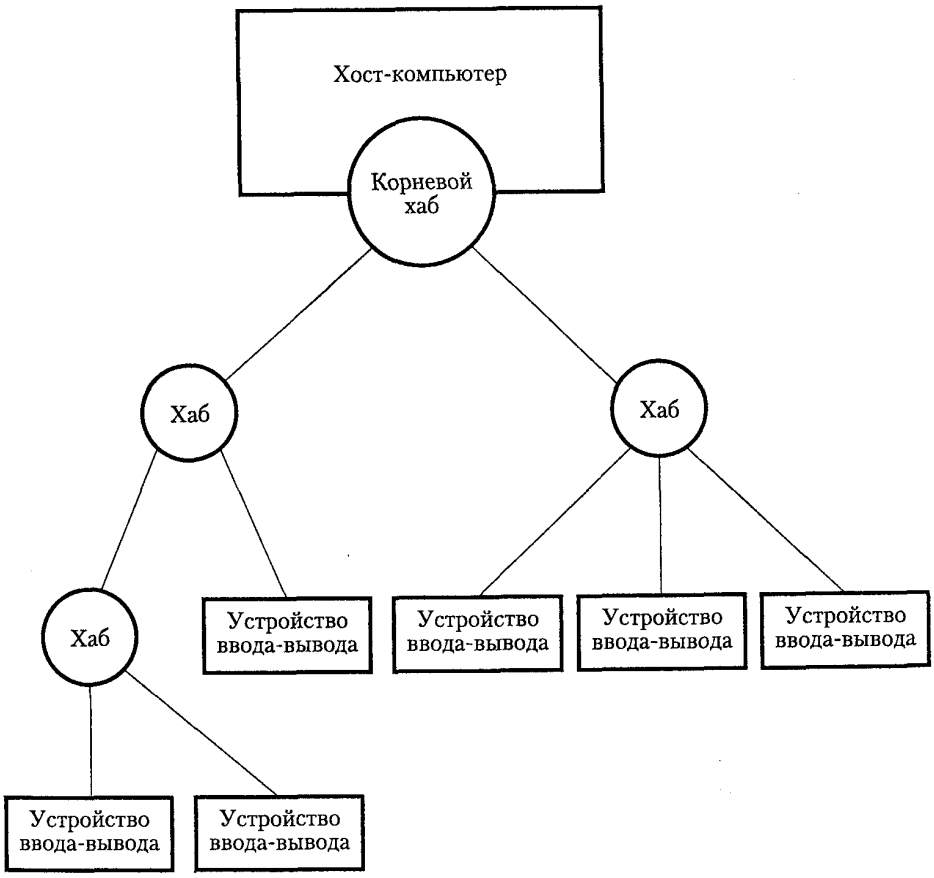


Рис. 7.9 – Структура дерева USB

В основе функционирования шины USB лежит принцип опроса устройств и устройство отсылает сообщение только в ответ на запрос хоста. Шина может работать в разделенном режиме, чтобы не задерживать сообщения, направляемые по высокоскоростным линиям.

6.8 Глава 8. Организация памяти

Основные понятия

К числу основных относятся следующие понятия: запоминающий элемент (ЗЭ), ячейка памяти, запоминающее устройство (ЗУ), память ЭВМ.

Определение: память – это часть ЭВМ, предназначенная для запоминания и выдачи информации.

Функции памяти: 1) хранение информации; 2) прием информации по запросу – запись; 3) выдача информации по запросу – чтение.

Операции чтения и записи информации в память принято называть **обращением к памяти**.

Запоминающий элемент – это место хранения бита информации. На основе ЗЭ организуется хранение более крупных единиц информации – слов.

Ячейка памяти – это фиксированная совокупность ЗЭ, обращение к которым производится одновременно как к единому целому. Ячейка памяти – это место хранения слова информации.

Адресный принцип доступа к информации, хранимой в ячейках памяти: ячейки памяти нумеруются числами 0, 1, …, E–1, номер ячейки называют ее **адресом**.

Количество ячеек памяти E – **емкость памяти** – и длина адреса связаны отношением $E = 2^m$, m – длина адреса в битах.

Совокупность ЗЭ, предназначенная для хранения блока информации, это также ячейка. За одно обращение можно записать или прочитать один блок. Блок является единицей обмена с внешней памятью. Блоки как ячейки памяти нумеруются номерами 0, 1, …, E–1 и рассматриваются как адреса блоков.

Доступ к ячейкам памяти с заданным адресом A обеспечивается схемой селекции, которая выбирает одну из ячеек при обращении с целью записи или чтения слова (блока) информации.

Определение ЗУ. Совокупность ячеек памяти, объединенных в единое целое схемой селекции.

Организация и основные характеристики ЗУ

ЗУ состоит из двух частей: запоминающего блока ЗЧ и блока управления БУ (рис. 8.1).

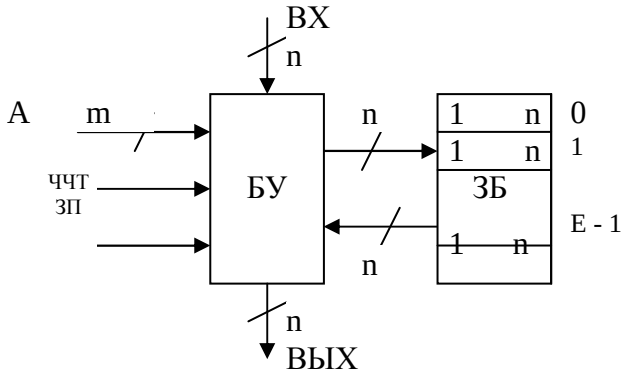


Рис. 8.1 – Структура ЗУ

Блок управления БУ обеспечивает выбор ячейки с адресом А и управление операциями чтения или записи по запросам ЧТ или ЗП:

Операция чтения обеспечивает выдачу слова информации из ячейки с адресом А на выходную шину: ЧТ: ВЫХ:= [А].

Операция записи обеспечивает прием слова со входной шины и запись его в ячейку с адресом А: ЗП: [А]: = ВХ.

Основные характеристики ЗУ:

- емкость,
- быстродействие, надежность,
- стоимость.

Емкость ЗУ определяется количеством ячеек, т.е. максимальным количеством информации, которая одновременно может храниться в ЗУ.

Быстродействие ЗУ определяется количеством операций обращения в единицу времени и зависит от продолжительности одной операции обращения: Vзу = 1/tобр.

Время обращения при выполнении операций ЧТ и ЗП:

$$t_{обр}^{чт} = \tau_д + \tau_{чт} + {}_{рег},$$

$$t_{обр}^{зп} = \tau_д + \tau_{подг} + \tau_{зп},$$

где $t_д$ – время доступа к информации (к ячейке ЗУ);

$\tau_{чт}$ – время выдачи содержимого ячейки на выходную шину ЗУ;

$\tau_{рег}$ – время регенерации – повторной записи в ячейку информации, разрушенной при чтении (это время равно нулю, если чтение производится без разрушения информации);

$\tau_{подг}$ – время на подготовку ячейки к записи новой информации. Подготовка, если она требуется, осуществляется путем стирания старой информации.

Время $\tau_{подг}$ =0, если стирания старой информации перед записью новой не требуется делать (например, в ЗУ на триггерах);

$\tau_{зп}$ – время собственно записи слова с входной шины ЗУ в выбранную схемой селекции ячейку ЗУ.

В зависимости от времени обращения (от быстродействия) ЗУ делятся на три класса: сверхоперативные, оперативные и внешние.

Надежность ЗУ обычно характеризуется временем наработки на отказ.

Стоимость ЗУ – интегральная характеристика. Она зависит от емкости, быстродействия, надежности ЗУ. Чем лучше эти технические характеристики, тем выше стоимость.

Применительно к ЗУ используется понятие удельной стоимости: $\delta=S/E$, где S – стоимость ЗУ. Под удельной стоимостью понимается стоимость хранения единицы информации: байта, КВ, МВ, ГВ.

Классификация запоминающих устройств

Деление ЗУ на классы осуществляется по различным признакам.

По способу доступа к информации ЗУ делят на ЗУ с произвольным (прямым) доступом и ЗУ с последовательным доступом к информации (рис. 8.2).

ЗУ с последовательным доступом в свою очередь делятся на ЗУ с периодическим (циклическим) доступом и ЗУ с апериодическим (нециклическим) доступом к информации.

В последовательных ЗУ с апериодическим доступом выбор ячейки с адресом A = n + N осуществляется путем последовательного перебора ячеек с адресами n + 1, n + 2, …, n + (N–1),

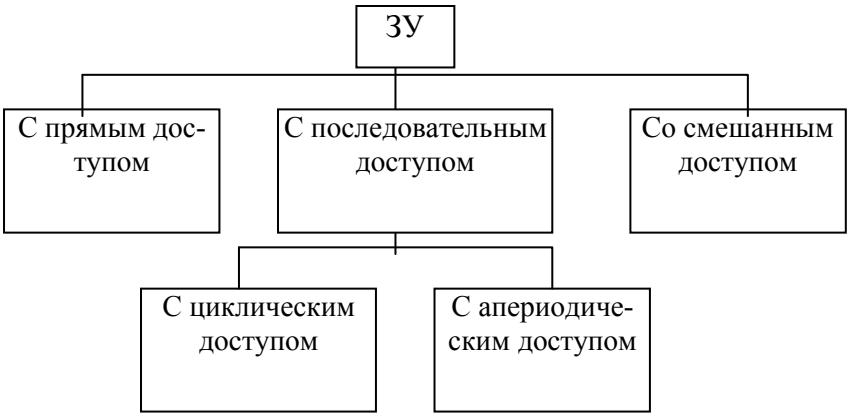


Рис. 8.2 – Классификация ЗУ по способу доступа

где n – номер ячейки, к которой производилось последнее обращение. Время доступа $\tau_д = (N–1)\ \tau_{яч}$, где $\tau_{яч}$ – время обращения к одной ячейке.

В последовательных ЗУ с циклическим доступом к информации доступ к адресуемой ячейке предоставляется периодически с периодом T = 1/w, где w – угловая скорость вращения носителя информации относительно неподвижных средств чтения/записи. Время доступа к ним является случайной величиной, которая лежит в пределах от 0 до T, среднее время доступа $t_{д.ср}=T/2$.

Накопители на магнитных дисках (НМД) относятся и к ЗУ со смешанным типом доступа.

В зависимости от принципа (средства) хранения информации ЗУ делятся на:

1) полупроводниковые ЗУ – на основе электронных ЗЭ;

2) ЗУ на основе магнитных носителей информации – магнитных «запоминающих элементов»; 3) ЗУ на основе оптических носителей информации – оптических ЗЭ.

В зависимости от положения информации относительно средств ЧТ/ЗП различают ЗУ статического и динамического типа.

К ЗУ статического типа относятся электронные ЗУ, в которых и информация и средства ЧТ/ЗП неподвижны, статичны.

В ЗУ динамического типа в движении находится либо информация, либо средства ЧТ/ЗП.

ЗУ динамического типа в свою очередь делятся на ЗУ с подвижным и неподвижным носителем информации. К ЗУ с подвижным носителем информации относятся НМЛ, НМД, НОД.

К ЗУ с неподвижным носителем информации относятся ЗУ на цилиндрических магнитных доменах (ЦМД). В них движется информация относительно неподвижного носителя и неподвижных средств ЧТ/ЗП (информация хранится в динамике).

По способу организации схемы селекции ЗУ делятся на адресные и безадресные. Безадресные в свою очередь делятся на ассоциативные, стековые и магазинные.

В зависимости от специфики использования информации ЗУ делятся на 3 типа:

1) ЗУ с частой сменой информации в процессе функционирования ЭВМ; 2) ЗУ с редкой сменой и 3) ЗУ без смены информации (постоянные ЗУ).

К ЗУ с частой сменой относятся СОЗУ, ОЗУ, ОВЗУ (НМД).

К ЗУ с редкой сменой относятся перепрограммируемые ПЗУ (ППЗУ).

К ЗУ без смены относятся постоянные ЗУ (ПЗУ, ПВЗУ – CD ROM).

В зависимости от того, требуется питание для хранения информации или нет, различают ЗУ энергозависимые и энергонезависимые.

Полупроводниковая память

Для запоминания электрических сигналов используют полупроводниковые структуры, на основе которых создаются биполярные транзисторы, МОП-транзисторы (металл-оксид полупроводники), МНОП-транзисторы (металл-нитрид-оксид полупроводники) и приборы с зарядовой связью (ПЗС). Блоки памяти на транзисторах организованы аналогично блокам памяти на ферритовых сердечниках. Основным недостатком полупроводниковой памяти следует считать значительное потребление электроэнергии и потерю информации при отключении электропитания. Биполярный транзистор представляет собой прибор с двумя p-n-переходами. Под действием напряжения база-коллектор изменяется состояние транзистора: он может быть открыт или заперт. Эти состояния используются как 0 и 1. Транзистор с металло-

ксидной стружкой является разновидностью полевого транзистора. Название этого транзистора происходит от трех составляющих: металлический затвор, слой изолирующего окисла и полупроводниковая подложка. Он представляет собой полупроводниковый прибор, у которого сопротивление между двумя его выводами управляется потенциалом, подаваемым на третий вывод (затвор). Под действием управляющего напряжения МОП-транзистор может находиться в закрытом или открытом состояниях. На биполярных транзисторах, полевых МОП- и МНОП-транзисторах, ПЗС собирают интегральные запоминающие устройства. Технология изготовления полупроводниковых структур позволяет создавать на их базе интегральные запоминающие устройства. Основу всех полупроводниковых элементов составляет кремниевая пластина, на которой собирается весь логический блок памяти. Так, один запоминающий блок на МОП-структуре представляет собой матрицу из 256 запоминающих элементов. Из упомянутых нами устройств ПЗС считаются новой страницей в развитии микроэлектроники, им специалисты прочат будущее и полагают, что они могут быть лучше, чем запоминающие устройства на цилиндрических магнитных доменах и магнитных дисках средних размеров.

Статические ОЗУ

Память ЭВМ – функциональная часть ЭВМ, предназначенная для записи, хранения и выдачи данных. В соответствии с этим различают три режима работы памяти: 1) хранения; 2) записи; 3) считывания.

Запоминающее устройство – устройство, реализующее функцию памяти данных. В ЗУ хранятся числа, над которыми должны быть произведены определенные действия, и числа – коды команд, определяющие характер этих действий.

Производительность и вычислительные возможности ЭВМ в значительной степени определяются составом и характеристиками ее ЗУ. В составе ЭВМ используется одновременно несколько типов ЗУ, отличающихся принципом действия, характеристиками и назначением.

Запись в ЗУ или считывание из него информации называют обращением к ЗУ. Важнейшими характеристиками памяти ЭВМ являются ее емкость и быстродействие.

Емкость памяти определяет максимальное количество слов, которые могут храниться в памяти, и выражается в битах или байтах. Например, если память ЭВМ составляет 64 Кбайт, это означает, что она может хранить 64 * 1024 восьмиразрядных слов.

Быстродействие памяти (ЗУ) определяется продолжительностью операции обращения к ЗУ. Время обращения $t_о$ при записи информации складывается из времени поиска числа $t_{п,}$ стирания ранее записанной информации $t_{ст}$ (при необходимости) и записи нового числа $t_{зп}$, т.е. $t_о = t_п + t_{ст} + t_{зп.}$ При считывании информации время цикла обращения складывается из из времен поиска $t_п$, считывания $t_{сч}$ и восстановления $t_{восст}$ считанных кодов т.е. $t_о = t_п + + t_{сч} + t_{воост}$. За продолжительность цикла обращения к памяти принимается величина $t_{обр}=\max(t_о^{зап},t_о^{счит})$.

Оперативной или основной памятью (ОП) называют устройство, которое служит для хранения информации, непосредственно используемой в процессе выполнения операций в арифметико-логическом устройстве и устройстве управления процессора.

В процессе обработки информации осуществляется тесное взаимодействие процессора и ОП. Из ОП в процессор поступают команды программы и операнды, над которыми производятся предусмотренные командой операции, а из процессора в ОП направляются для хранения промежуточные и конечные результаты обработки.

Характеристики ОП непосредственно влияют на основные показатели ЭВМ и в первую очередь на скорость ее работы. Оперативная память высокопроизводительных ЭВМ должна иметь емкость несколько миллионов байт и цикл обращения около 1мкс (и менее).

По способу хранения информации ОЗУ делят на статические и динамические.

Лучшими скоростными характеристиками обладает статическая память SRAM (Static Random Access Memory), но стоимость ее существенно выше, т.к. для хранения 1 бита здесь уже требуется триггер, включающий в себя 4 транзистора. Схемы статиче-

ской памяти к тому же менее компактны, чем динамической, и размещение в приемлемом объеме требуемых сегодняшними приложениями многих десятков мегабайт SRAM оказывалось проблематичным, особенно в мобильных и полупрофильных настольных компьютерах. В настоящее время схемы SRAM находят широкое применение в системе кэш-памяти.

В статических ЗУ цифровой код остается неподвижным относительно носителя информации в течение всего времени хранения информации.

ОЗУ могут быть выполнены на различной элементно-технологической основе.

Запоминающие элементы на биполярных транзисторах

Запоминающий элемент – триггер на биполярных транзисторах с непосредственными связями представлен на рис. 8.3. Будем считать, что в ЗЭ записан 0, если транзистор VT1 открыт, а транзистор VT2 закрыт. От $+E_k$ к $–E_д$ (к «земле») через открытый транзистор триггера ток может проходить через цепи эмиттеров. Верхние по схеме эмиттеры подключены к разрядным шинам, нижние соединены и выведены на адресную шину. При выборе ячейки, в которую входят рассматриваемые ЗУ, данная адресная шина возбуждается – ее потенциал возрастает, и цепь тока через нижние эмиттеры обрывается.

В режиме хранения ЗЭ не выбраны – адресная шина имеет низкий потенциал, а разрядные – наиболее высокий. Поэтому ток открытого транзистора протекает через нижний эмиттер на адресную шину и по ней на «землю».

В режиме записи происходит выборка ЗЭ ячейки – данная адресная шина приобретает высокий потенциал, и ток отпертого транзистора ЗЭ может протекать только на разрядную шину. При появлении на входе двоичной переменной, записываемой в данный ЗЭ, и сигнала разрешения записи одна разрядная шина получает высокий потенциал, а другая – благодаря инвертору – низкий. Если перед этим триггер находился в состоянии 0 (VT1 отперт), то он переключается в состояние 1 (отперт VT2).

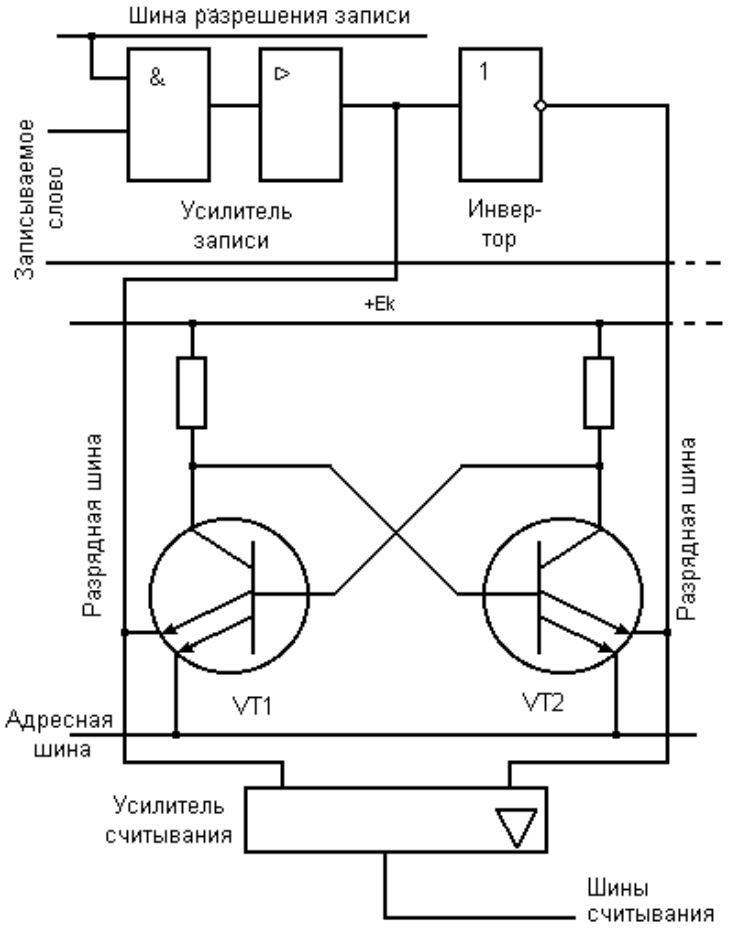


Рис. 8.3 – Запоминающий элемент – триггер
на биполярных транзисторах

В режиме считывания вновь происходит выборка ЗЭ ячейки – адресная шина получает высокий потенциал, и ток открытого транзистора протекает по разрядной шине на один из входов усилителя считывания. На его выходе появляется «лог. 1», если отперт VT2, и «лог. 0», если отперт VT1.

В случае необходимости микросхемы ЗУ можно объединить, увеличивая тем самым емкость памяти. Для этого они имеют специальный вывод – «Выбор корпуса» (ВК).

Запоминающие элементы на МОП-транзисторах

В зависимости от типа ЗЭ на основе МОП-транзисторов можно построить ЗУ статические или динамические. В первом случае в качестве ЗЭ служит статический триггер, во втором случае информация запоминается на емкости затвора МОП-транзистора. Запоминающее устройство на МОП-транзисторах, так

же как и на биполярных транзисторах, может быть с пословной и двухкоординатной произвольной выборкой.

Пример простейшей схемы ЗЭ-триггера на МОП-транзисторах для ЗУ с пословной выборкой приведен на рис. 8.4,*а*. Триггер образован транзисторами VT1–VT4. Управление триггером для записи и считывания осуществляется переключением транзисторов VT5 и VT6. Временные диаграммы работы такого ЗЭ представлены на рис. 8.4,*б*. В исходном состоянии напряжение на обоих разрядных шинах P_1 и P_0 равно нулю, а на шине слова А потенциал равен напряжению питания схемы. При этом транзисторы VT5 и VT6 закрыты, так как разность потенциалов между затворами и истоками по абсолютной величине меньше порогового напряжения. Триггер, находится в одном из устойчивых состояний.

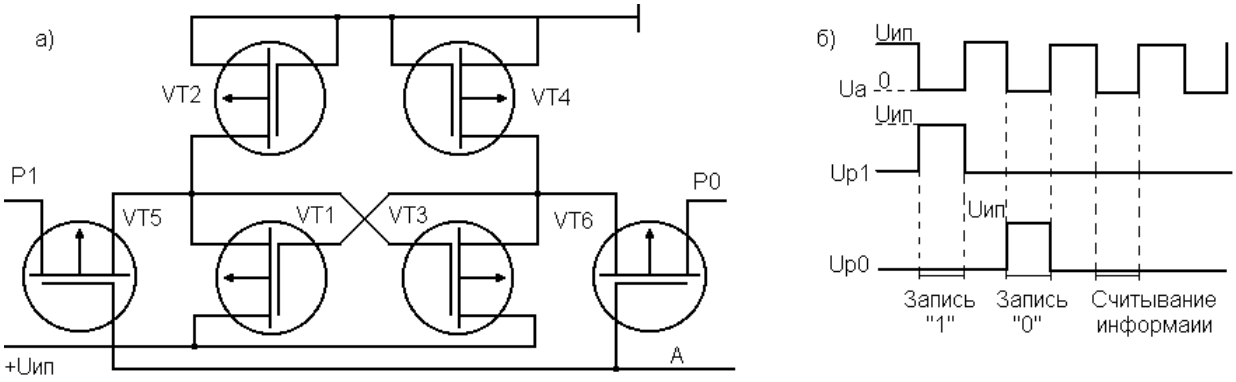


Рис. 8.4 – Запоминающий элемент – триггер на основе МОП структуры:
а – схема; *б* – временная диаграмма работы

Пусть транзистор VT3 открыт, а транзистор VT1 закрыт. При записи «1» в шину слова подается отрицательный сигнал, изменяющий напряжение в ней до нуля, одновременно в разрядную шину P_1 подается положительный сигнал, изменяющий напряжение в ней до напряжения питания $U_{ип}$. При этом транзистор VT5 открывается, так как разность потенциалов между затвором и истоком становится отрицательной. Положительный сигнал поступает на сток транзистора VT1 и на затвор транзистора VT3. Разность потенциалов между затвором и истоком транзистора VT3 становится меньше порогового напряжения, и этот транзистор закрывается. После запирания транзистора VT3 открывается

транзистор VT1 и на его стоке устанавливается положительное напряжение, что соответствует состоянию «1». Напряжение на стоке транзистора VT3 становится равным нулю.

При записи «0» в ЗЭ необходимо при нулевом напряжении на шине слова подать напряжение $U_{ип}$ в разрядную шину P_0, при этом через открытый транзистор VT6 положительное напряжение, попадая на затвор транзистора VT1, запирает его, что приводит к открыванию транзистора VT3. Для считывания информации, предварительно записанной в ЗЭ, необходимо подать отрицательный сигнал только на шину слова, изменяя в ней напряжение до нуля. При этом транзисторы VT5 и VT6 оказываются открытыми, и через транзистор, присоединенный к точке триггера с положительным потенциалом, протекает ток, поступающий в соответствующую разрядную шину и далее на усилитель считывания.

Структура ЗУ

Они представляют сбой матрицу запоминающих элементов, каждый из которых может быть установлен в одно из двух устойчивых состояний. Таким элементом обычно является триггер. Из ЗЭ строится матрица памяти – основа ЗУ. Построение (организация) матрицы определяется способом выборки (опроса) ЗЭ при записи или считывании.

В структурной схеме матрицы с пословной выборкой и одной ступенью дешифрации (рис. 8.5,*а*) одна строка образует слово из m разрядов. На схеме символами A_1, A_2,…, A_n обозначены адресные, а P_1, P_2,…, Pm – разрядные шины. Как видно из схемы, адресные шины связаны с каждым ЗЭ одного слова, в то время как разрядные шины имеют связь с ЗЭ одноименного разряда всех слов при наличии в адресной шине A_i сигнала выбора i-го слова, соответствующего высокому уровню. Состояние каждого из ЗЭ в этом слове может быть считано по разрядным шинам P_1 – P_m. Если необходимо записать информацию по выбранному адресу A_i, на разрядные шины P_1, P_2,…, P_m подаются электрические сигналы «1» и «0», которые попадут на каждый из ЗЭ i-ой строки: $ЗЭ_{i1}$, $ЗЭ_{i2}$, $ЗЭ_{i3}$,…, $ЗЭ_{im}$.

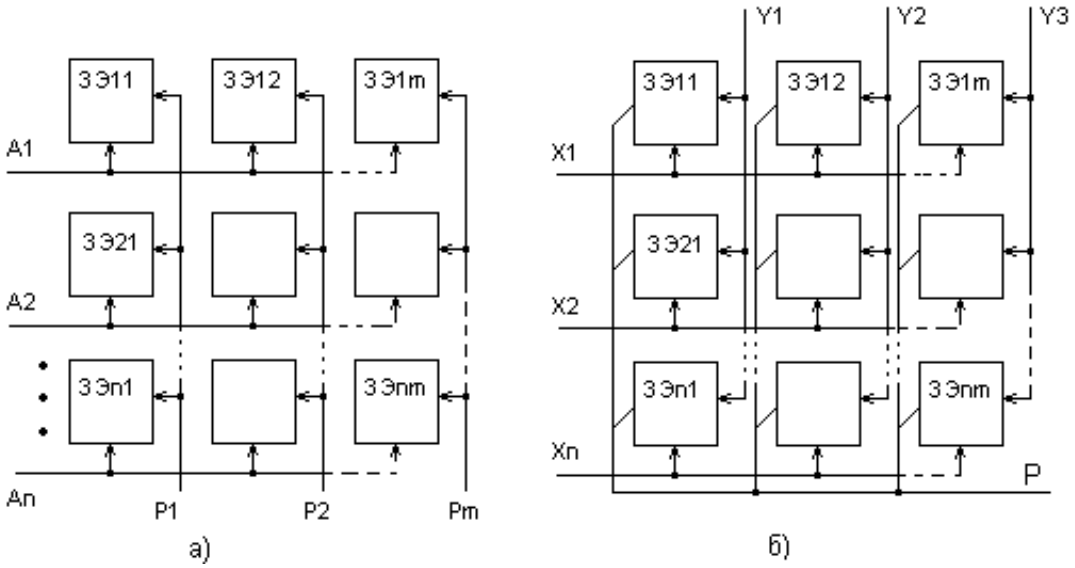


Рис. 8.5 – Структурные схемы матриц (накопителей информации):
а – с пословной выборкой и одной ступенью дешифрации;
б – двухкоординатной с двумя ступенями дешифрации

На рис. 8.5,*а* не показаны устройства управления матрицей (дешифратор с адресными формирователями, усилители считывания записи), которые для повышения надежности работы ОЗУ изготовляются на одном кристалле с матрицей.

В структурной схеме двухкоординатной матрицы информации с двумя ступенями дешифрации (рис. 8.5,*б*) ЗЭ выбирается с помощью двух адресных шин. Например, при наличии сигнала, соответствующего высокому уровню, на адресных шинах будет выбран только $ЗЭ_1$. Его состояние можно считывать по общей для всех элементов разрядной шине Р. Чтобы записать «1» в выбранный ЗЭ, по разрядной шине необходимо подать сигнал, соответствующий высокому уровню. Эта организация матрицы позволяет оперировать mn одноразрядными словами.

Регистрация и хранение информации берут свое начало от высеченных на камне изображений в эпоху неолита и бронзового века. Прошли века, пока к человеку пришли письменность, а затем книгопечатание**.** Только в XIX в. была изобретена фотография (1839 г.) и кинематограф (1895 г.). Эти два замечательных изобретения позволили регистрировать и запоминать информацию в виде изображений и звука. Интересный способ хранения дискретной информации предложил французский механик Ж. Вакансон, создавший в 1741 г. ткацкий станок с программным

управлением. Для запоминания программы он использовал механический перфорированный барабан. Лишь 60 лет спустя барабан был заменен перфорированным картоном, явившимся прообразом перфокарт и перфолент. Принципиально важным событием явилось изобретение записи электрических сигналов на магнитной ленте, положившее начало многим разновидностям устройств магнитной записи. Производство магнитной ленты началось сравнительно недавно с 1928 г., хотя принцип записи звука с помощью магнитного поля известен уже более ста лет.

Память на цилиндрических магнитных доменах

Физический эффект: в некоторых магнитных материалах при воздействии внешнего магнитного поля возникают отдельные области, отличающиеся направлением намагниченности. Это – домены (domain – управляемая область).

Под действием слабого внешнего магнитного поля домены могут перемещаться в пластине ферромагнитного материала по заранее заданным направлениям с высокой скоростью (рис. 8.6).

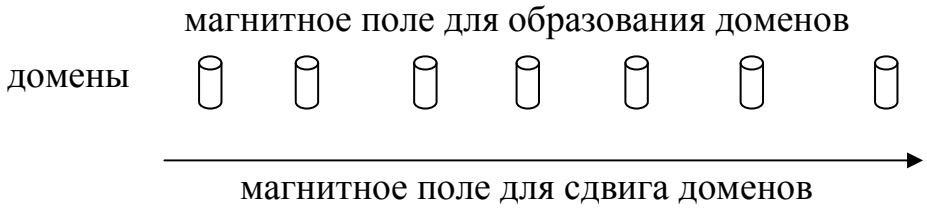


Рис. 8.6 – Принцип формирования памяти на ЦМД

Это свойство перемещения доменов позволяет создавать запоминающие устройства. Хорошим доменообразующим материалом является пленка ферритграната. Доменные структуры могут быть полосковыми, кольцевыми,

Размер домена составляет от 0,01 до 0,1 мм, поэтому на одном квадратном сантиметре материала можно разместить несколько миллионов доменов. Домены можно генерировать или уничтожать, их перемещение позволяет создавать логические операции, потому что наличие ил отсутствие домена в определенной точке магнитного кристалла можно считать за 1 или 0. При отключении домены сохраняются.

На базе доменосодержащего кристалла выпускаются полупроводниковые модули – чипы. Для образования в чипе цилиндрических доменов его помещают в постоянные и вращающиеся магнитные поля, образованные постоянным магнитом и электромагнитом.

Доменный регистр состоит из устройства ввода доменов (генератор доменов), вывода (резистивный датчик) и пермаллоевой пленки. Генерация доменов производится путем непосредственного зарождения доменов в той или иной точке кристалла. Генерация и ввод доменов в регистр сдвига производятся токопроводящей петлей из пермаллоевой пленки. При появлении тока в генераторе создается локальное магнитное поле. Под действием этого поля в области, ограниченной контуром петли, зарождается домен, который затем под действием поля постоянного смещения принимает цилиндрическую форму. В таком сформированном виде домен поступает в сдвиговый регистр. Один чип способен запомнить до 150 бит, а весь накопитель – 10 Мбит. Существовали накопители на 16 Мбит. Запоминающее устройство такой емкости имеет размеры небольшого чемодана.

Память на магнитных дисках

Накопитель НМД имеет три физических узла:

- кассеты с диском,
- привода диска,
- электронной части.

Информация записывается на магнитный слой по концентрическим дорожкам; стандартные диаметры 88,9; 133,35 мм, толщина примерно 2 мм; обе поверхности являются рабочими.

Диск устанавливается на вал, который приводится во вращение электромотором. Зазор между поверхностью диска и магнитной головкой составляет 2,5–5,0 мкм и должен сохраняться постоянным в процессе работы. С этой целью производят тщательную обработку поверхности диска и используют специальные головки аэростатического типа, плавающие над диском. Головки для записи и считывания перемещаются в зазоре между дисками с помощью суппорта, управляемого сервоприводом специальными командами. Средняя емкость дорожки достаточно велика (примерно 40 Кбайт), поэтому каждая дорожка разбивается

на секторы для более быстрого поиска. При аппаратном разделении диска на секторы на внутренней окружности имеются 32 отверстия, отмечающие начала секторов. Емкость дисков может достигать сотен Гбит, а время доступа к информационному блоку – от 1 до 10 мс.

Основное преимущество дисковой памяти – сравнительно быстрый поиск нужного информационного блока и возможность смены дисков, что позволяет считывать с дисков данные, записанные на другом компьютере. Особенность жестких дисков – герметизация носителя, что позволяет уменьшить зазоры между головками и диском, существенно увеличить плотность записи. Герметизация повышает также надежность устройства. Постоянные (жесткие) диски емкостью 2000 Мбайт с неперемещаемыми головками имеют время доступа 10–15 мс; гибкие (Floppy) диски (дискеты) емкостью 1440 Кбайт имеют время доступа 0,1 с (рис. 8.7). Гибкие диски обеспечивают удобную загрузку программ и данных, могут использоваться и как центральное устройство хранения и поиска, поскольку имеют прямой доступ к информации, что позволяет исключить из системы накопители на жестких дисках.

Пути данных между различными системами компьютера

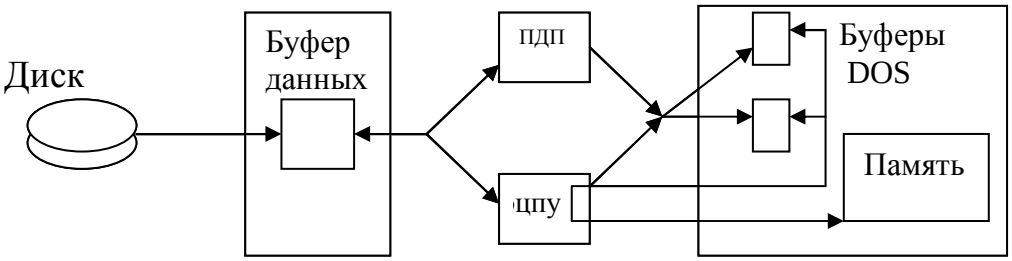


Рис. 8.7 – Память на оптических дисках

Излучение лазера в оптическом диапазоне волн может иметь не расходящийся пучок с очень малым сечением и достаточно большой мощностью и использоваться для записи и считывании информации.

Основа записи информации: модуляции дискретными значениями 0 и 1 лазерного излучения, которое оставляет на поверхности диска метки, вызванные воздействием луча на металл. Поверхность диска покрывается тонким слоем отражающего металла – теллура. Луч записывающего лазера модулируется сле-

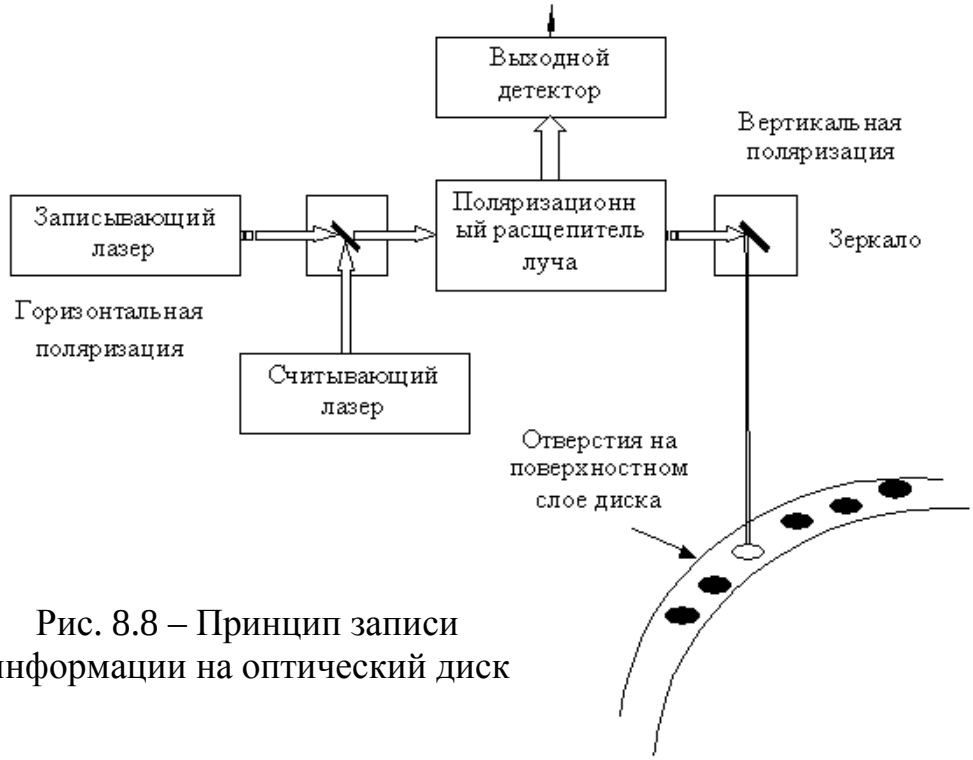


Рис. 8.8 – Принцип записи информации на оптический диск

дующим образом: при модуляции «единицей» луч прожигает в пленке теллура микроскопическое отверстие. Если 1 следует одна за другой, отверстие оказывается вытянутым за счет вращения диска. Луч считывающего лазера отражается от поверхности диска, кроме мест, выжженных записывающим лучом, и попадает в фотодетектор. Такой метод позволяет записывать информацию один раз (рис. 8.8).

Запись информации происходит по концентрическим дорожкам. Для удобства поиска нужного информационного блока диск разбивается на секторы, каждому сектору присваивается адрес. Скорость записи определяется мощностью луча записывающего лазера, скоростью вращения диска и скоростью перемещения лазерного луча. Скорость записи современных пишущих CD-ROM'ов составляет около 600 Кбайт/с.

Для различения излучений записывающего и считывающего лазеров применяют разделение сигналов с помощью поляризации. Электромагнитная волна с вертикальной поляризацией не воспринимается детектором (приемником), настроенным на прием волны с горизонтальной поляризацией, и наоборот. Детектором служит фоточувствительный приемник, воспроизводящий изменение мощности луча, отраженного от поверхности диска.

Диск, покрытый пленкой теллура, может сохранять записанную информацию в течение примерно 10 лет. Оптический диск обладает огромной плотностью записи. Если элемент записи на магнитном диске имеет размер около 2 мкм, то размер оптического элемента записи определен размерами дифракции луча, что составляет 0,4 мкм. В результате на оптический диск можно записать до 6 Гбит информации. В последнее время распространение получили оптические диски стандартного диаметра 5,25 дюйма (133,35 мм). Современная техника записи позволяет зафиксировать на диске такого диаметра 420 тыс. страниц.

6.9 Глава 9. Многопроцессорные системы

Необходимость появления – решение крупномасштабных задач по обработке информации в научных исследованиях, экономике, экологии, управлении большими системами, прогнозе.

Главная отличительная особенностью многопроцессорной вычислительной системы – ее производительность, т.е. количество операций, производимых системой за единицу времени.

Производительность

Пиковая	Реальная
величина, равная произведению пиковой производительности одного процессора на число таких процессоров в данной машине	зависит от взаимодействия программной модели, в которой реализовано приложение, с архитектурными особенностями машины, на которой приложение запускается

Под архитектурой многопроцессорной системы понимают способ параллельной обработки данных, используемый в системе, организацию памяти, топологию связи между процессорами, и способ исполнения системой арифметических операций.

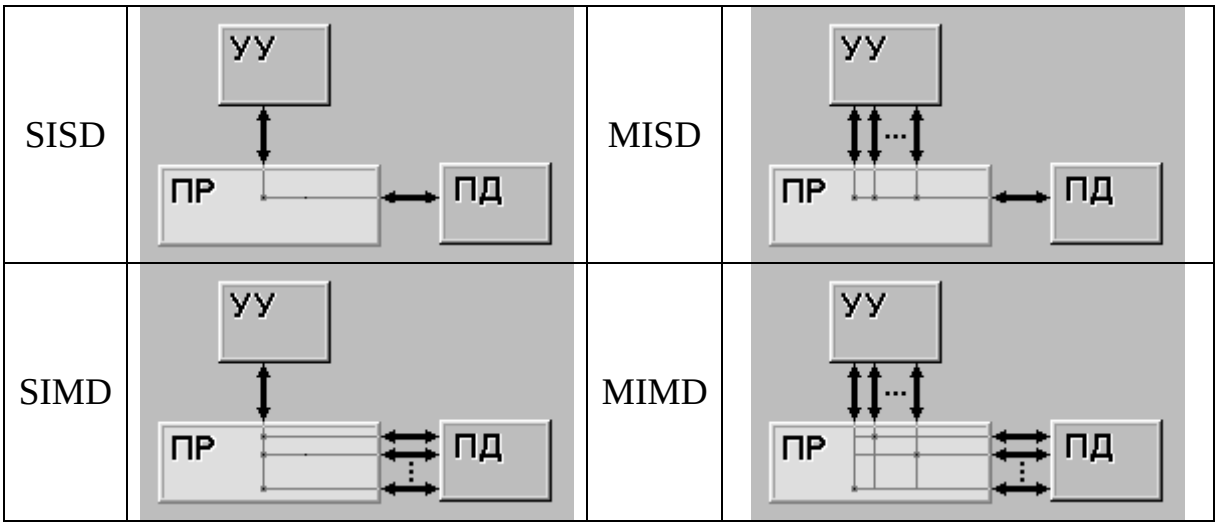


Рис. 9.1 – Классификация Флинна

Классификация Флинна

SISD (single instruction stream / single data stream) – одиночный поток команд и одиночный поток данных. Один центральный процессор, способный обрабатывать только один поток последовательно исполняемых инструкций,

MISD (multiple instruction stream / single data stream) – множественный поток команд и одиночный поток данных. Множество инструкций выполняется над единственным потоком данных.

SIMD (single instruction stream / multiple data stream) – одиночный поток команд и множественный поток данных. Большое количество процессоров, имеющих собственную память, в пределах от 1024 до 16384 шт., выполняющих одну и ту же инструкцию относительно разных данных в жесткой конфигурации.

MIMD (multiple instruction stream / multiple data stream) – множественный поток команд и множественный поток данных. Параллельно выполняется несколько потоков инструкций над различными потоками данных.

Современные многопроцессорные системы – это MIMD.

Основной параметр классификации – это отношение памяти:

- общая память (SMP архитектура).

• Многопроцессорные системы с общей (разделяемой) памятью, где каждый процессор компьютера обладает возможностью прямого доступа к общей памяти, используя общую шину (возможно, реализованную на основе высокоскоростной);

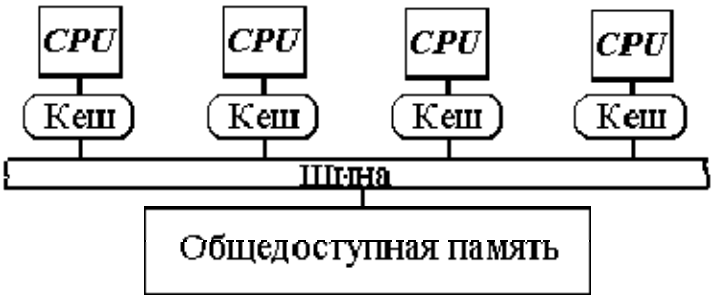


Рис. 9.2 – Общая память

- распределенная память (ММР архитектура) (рис. 9.3).

Многопроцессорные системы с **распределенной** памятью, где каждый процессор имеет доступ только к локальной собственной памяти. Процессоры объединены в сеть. Доступ к удаленной памяти возможен только с помощью системы обмена сообщениями, память физически распределена, но логически общедоступна (NUMA-архитектура).

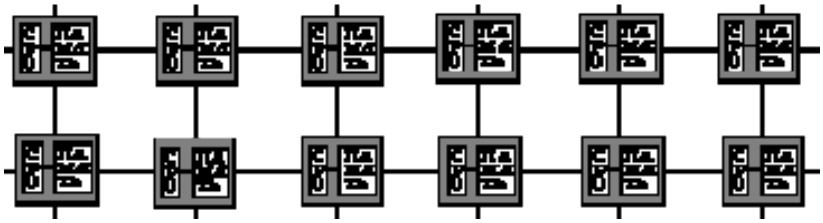


Рис. 9.3 – Распределенная память

Количество таких архитектур велико. Например, в некоторых архитектурах каждый процессор имеет как прямой доступ к общей памяти, так и собственную локальную память (рис. 9.4).

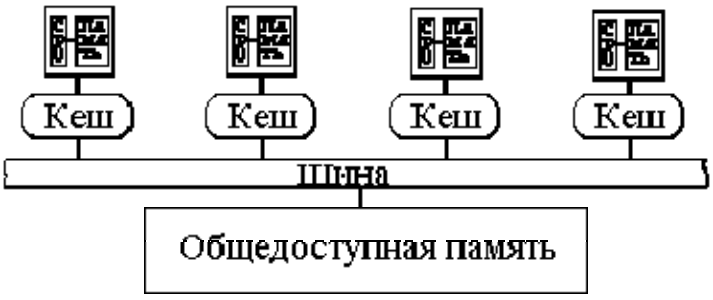


Рис. 9.4 – Система с неоднородной памятью

Векторно-конвейерные суперкомпьютеры

Первый векторно-конвейерный компьютер Cray-1 (1976 год) имел удачную архитектуру и положил начало целому семейству компьютеров.

Два принципа заложены в архитектуре процессоров:

- конвейерная организация обработки потока команд;
- введение в систему команд набора векторных операций, которые позволяют оперировать с целыми массивами данных.

Конвейерная обработка эффективна при загрузке конвейера близкой к полной, а скорость подачи новых операндов соответствует максимальной производительности конвейера.

При выполнении векторной команды одна и та же операция применяется ко всем элементам вектора (или чаще всего к соответствующим элементам пары векторов).

Главный принцип вычислений на векторной машине состоит в выполнении некоторой элементарной операции или комбинации из нескольких элементарных операций, которые должны повторно применяться к некоторому блоку данных. Таким операциям в исходной программе соответствуют небольшие компактные циклы.

SMP-архитектура

SMP-архитектура (symmetric multiprocessing) – симметричная многопроцессорная архитектура. Главной особенностью систем с архитектурой SMP является наличие общей физической памяти, разделяемой всеми процессорами (рис. 9.5).

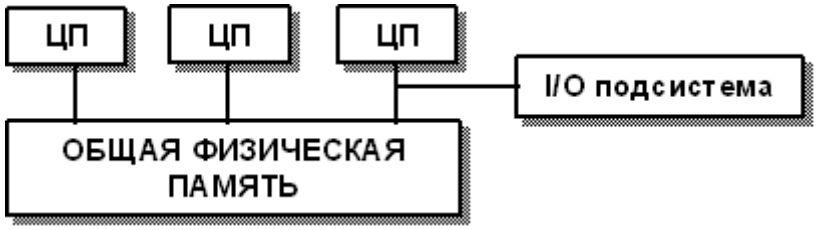


Рис. 9.5 – Схематический вид SMP-архитектуры

Память является способом передачи сообщений между процессорами, при этом все вычислительные устройства при обращении к ней имеют равные права и одну и ту же адресацию для

всех ячеек памяти. Поэтому SMP-архитектура называется симметричной. Последнее обстоятельство позволяет эффективно обмениваться данными с другими вычислительными устройствами. SMP-система строится на основе высокоскоростной системной шины (SGI PowerPath, Sun Gigaplane, DEC TurboLaser), к слотам которой подключаются функциональные блоки трех типов: процессоры (ЦП), операционная система (ОС) и подсистема ввода/вывода (I/O). Для подсоединения к модулям I/O используются медленные шины (PCI, VME64).

MPP-архитектура (рис. 9.6)

MPP-архитектура (massive parallel processing) – массивно-параллельная архитектура или, как иногда говорят, – система с массовым параллелизмом. Главная особенность такой архитектуры состоит в том, что память физически разделена. В этом случае система строится из отдельных модулей, содержащих один или несколько процессоров, локальный банк операционной памяти (ОП), два коммуникационных процессора (рутера) или сетевой адаптер, иногда – жесткие диски и/или другие устройства ввода/вывода.

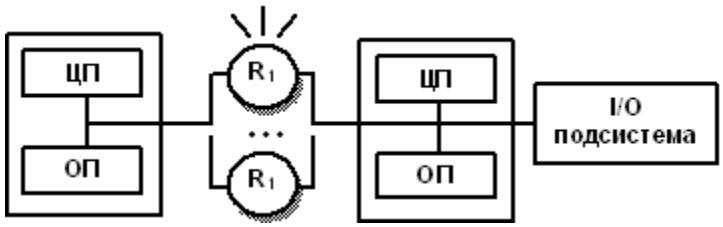


Рис. 9.6 – Схема модуля MPP системы

Один маршрутизатор используется для передачи команд, другой – для передачи данных. Доступ к банку ОП из данного модуля имеют только процессоры (ЦП) из этого же модуля. Оперативная память в MPP-системах имеет 3-х уровневую структуру:

- кэш-память процессора;
- локальная оперативная память узла;
- оперативная память других узлов.

Гибридная архитектура (NUMA) (рис. 9.7)

Особенность архитектуры – неоднородный доступ к памяти. Это удобства систем с общей памятью и относительную дешевизну систем с раздельной памятью. Память является физически распределенной по различным частям системы, но логически разделяемой, так что пользователь видит единое адресное пространство. Система состоит из однородных базовых модулей (плат), состоящих из небольшого числа процессоров и блока памяти. Модули объединены с помощью высокоскоростного коммутатора. Поддерживается единое адресное пространство, аппаратно поддерживается доступ к удаленной памяти, т.е. к памяти других модулей. При этом доступ к локальной памяти осуществляется в несколько раз быстрее, чем к удаленной. По существу архитектура NUMA является MPP архитектура, где в качестве отдельных вычислительных элементов берутся SMP узлы.

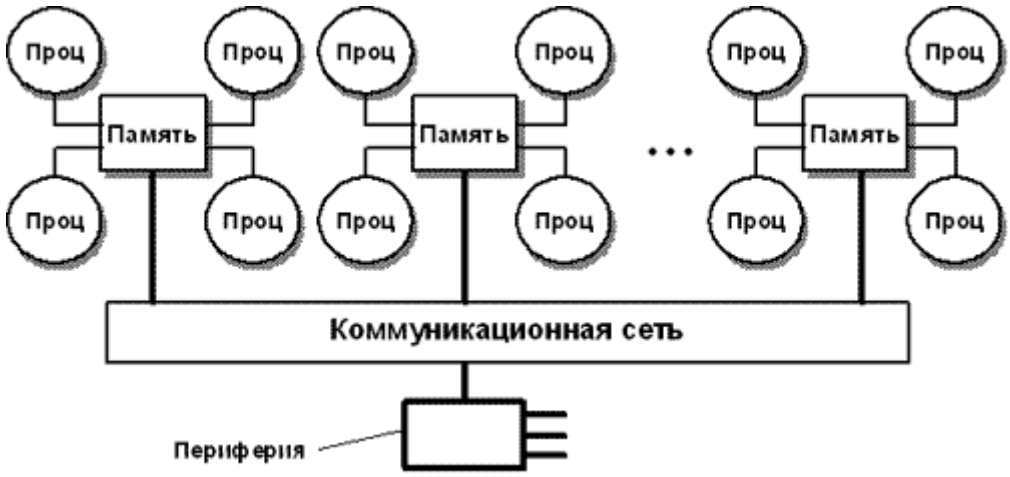


Рис. 9.7 – Структурная схема компьютера с гибридной сетью: четыре процессора связываются между собой при помощи кроссбара в рамках одного SMP-узла. Узлы связаны сетью типа «бабочка» (Butterfly)

Кластерные системы

Кластерные технологии стали логическим продолжением развития идей, заложенных в архитектуре MPP-систем. В качестве активных элементов используются обычные компьютеры. Развитие коммуникационных технологий, появление высокоскоро-

стного сетевого оборудования и специального программного обеспечения, такого как система MPI, реализующего механизм передачи сообщений над стандартными сетевыми протоколами, сделали кластерные технологии общедоступными.

Кластерные технологии позволяют для достижения необходимой производительности объединять компьютеры самого разного типа, начиная от персональных компьютеров и заканчивая мощными суперкомпьютерами. В частности, одним из первых был реализован проект COCOA, в котором на базе 25 двухпроцессорных персональных компьютеров общей стоимостью порядка $100000 была создана система с производительностью, эквивалентной 48-процессорному Cray T3D стоимостью несколько миллионов долларов США.

Производительность систем с распределенной памятью зависит от производительности коммуникационной среды. Коммуникационную среду можно достаточно полно охарактеризовать двумя параметрами: *латентностью* – временем задержки при посылке сообщения и *пропускной способностью* – скоростью передачи информации.

Суперкомпьютеры

Одним из современных представителей векторно-конвейерных суперкомпьютеров является компьютер NEC SX-6 линейки SX компании NEC. К основным компонентам архитектуры NEC SX относятся центральный процессор, подсистема оперативной памяти и подсистема ввода-вывода. Данные компоненты объединяются в узлы SMP-архитектуры, которые, в свою очередь, связаны через межсоединение Internode Crossbar Switch (IXS). При этом вся память всех узлов является общей; иными словами, многоузловые модели SX обладают архитектурой NUMA.

Архитектура PowerScale – реализация SMP-системы

В современных системах SMP необходима высокопроизводительная подсистема памяти для обеспечения скоростных RISC-процессоров данными и командами. Решение – это высокоскоростная кэш-память. В архитектуре PowerScale применена новая организация управления кэш-памятью и доступа к памяти.

Обычно выполнение прикладных систем связано с необходимостью манипулирования огромными объемами данных и разделения доступа к этим данным между многими пользователями или программами. Такого рода рабочая нагрузка характеризуется наличием больших рабочих наборов данных с низким уровнем локализации. При моделировании прикладных систем подобного профиля на системах SMP были замечены два особых эффекта.

Процессор PowerPC определяет слабо упорядоченную модель памяти, позволяющую оптимизировать использование пропускной способности памяти системы. Это достигается за счет того, что аппаратуре разрешается переупорядочивать операции загрузки и записи так, что требующие длительного времени операции загрузки могут выполняться ранее определенных операций записи. Такой подход дает возможность уменьшить действительную задержку операций загрузки. Архитектура PowerScale полностью поддерживает эту модель памяти как на уровне процессора за счет набора команд PowerPC, так и глобально путем реализации различных ограничений.

9.10 Глава 10. Нейрокомпьютерные системы

Нейрокомпьютерные системы это информационные системы, использующие нейронные сети для обработки информации. Достоинство этого способа обработки информации заключается в новом подходе, связанном с использованием простейших элементов – нейронов.

Сравнение традиционного и нейросетевого способов обработки информации.

Традиционный способ: сложные элементы (микропроцессоры) в узлах, связи простые.

Нейросетевой подход: простые элементы в узлах (нейроны), связи сложные.

Поэтому применение нейросетевой технологии перспективно для обработки информации, в отличии от искусственного интеллекта.

Вид нейрона приведен на рис 10.1.

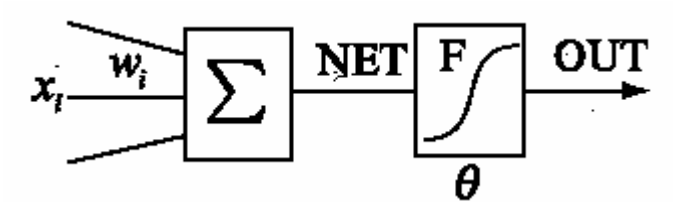


Рис. 10.1 – Математическая модель нейрона:
W ij _ _ синаптические коэффициенты, X i – входной вектор, Net – выход сумматора, F –активационная фукнкция, Out – выход нейрона

Классификация нейронных парадигм по структурной и функциональной организации (НС-нейронные сети):

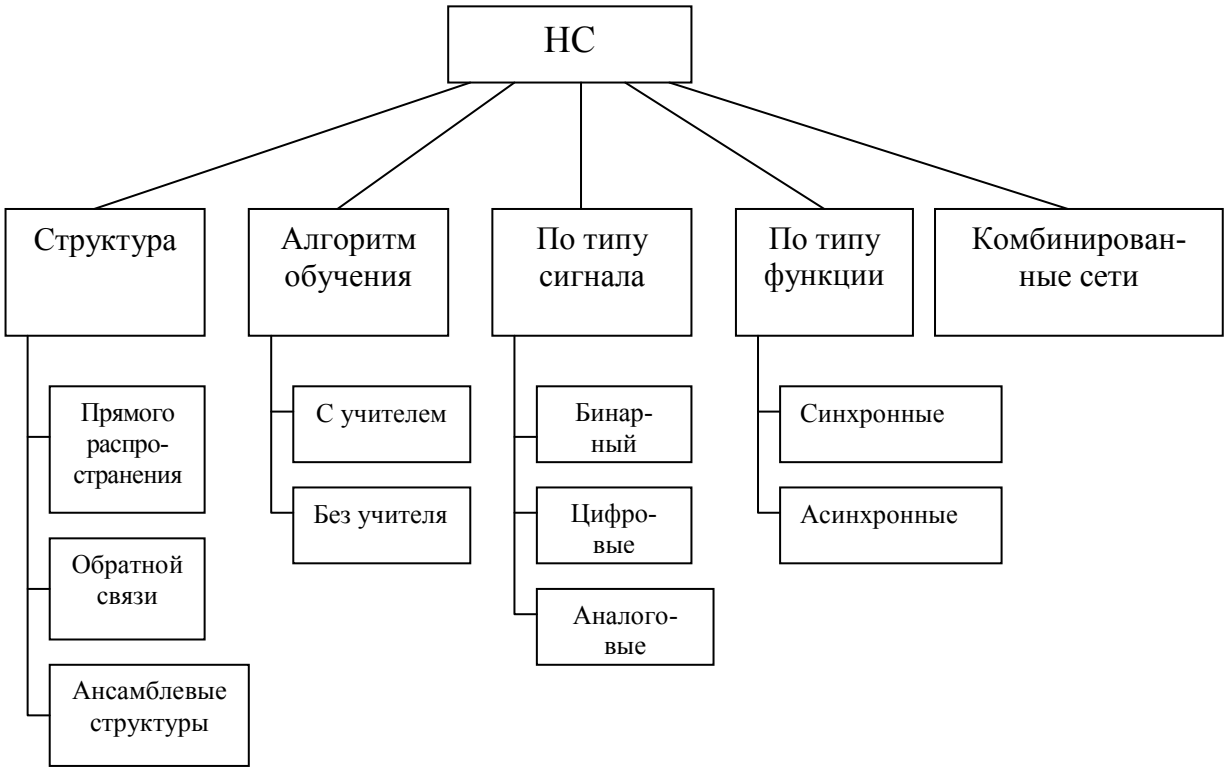


Рис. 10.2 – Классификация нейросетевых парадигм

Классификация по виду нейронных сетей:

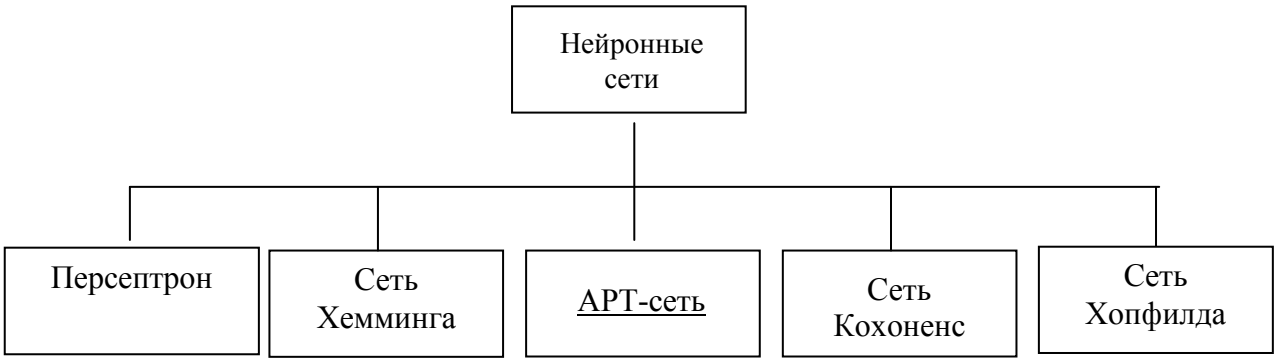


Рис. 10.3 – Классификация нейронных сетей по виду

Виды активационных функций

1. **Жесткая ступенька** (рис.10.4,*а*):

$$Y = \begin{cases} 0, & S > Q \\ 1, & S < Q \end{cases}$$

Используется в классическом формальном нейроне. Эта функция чрезмерно упрощена и не позволяет моделировать схемы с непрерывными сигналами. Отсутствие первой производной затрудняет применение градиентных методов для обучения таких нейронов.

2. Логистическая функция (сигмоида, функция Ферми, рис. рис. 10.4,*б*):

$$Y = \frac{1}{1 + e^{-NET}}.$$

Применяется очень часто для многослойных персептронов и других сетей с непрерывными сигналами. Гладкость, непрерывность функции – важные положительные качества. Непрерывность первой производной позволяет обучать сеть градиентными методами, например метод обратного распространения ошибки. Функция симметрична относительно точки (S=0, Y=1/2), это делает равноправными значения Y=0 и Y=1,что существенно в работе сети. Тем не менее диапазон выходных значений от 0 до 1 несимметричен, из-за этого обучение значительно замедляется. Данная функция – сжимающая, т.е. для малых значений S коэффициент передачи K=Y/S велик, для больших значений он сни-

жается. Поэтому диапазон сигналов, с которыми нейрон работает без насыщения, оказывается широким. Значение производной легко выражается через саму функцию. Быстрый расчет производной ускоряет обучение.

3. Гиперболический тангенс (рис.10.4,*в*):

$$OUT = \text{th}(NET) = \frac{e^{NET} - e^{-NET}}{e^{NET} + e^{-NET}} .$$

Тоже применяется часто для сетей с непрерывными сигналами. Функция симметрична относительно точки (0,0), это – преимущество по сравнению с сигмоидой. Производная также непрерывна и выражается через саму функцию.

4. **Пологая ступенька** (рис. 10.4,*г*):

$$OUT = \begin{cases} 0, & NET \le \theta \\ \dfrac{(NET - \theta)}{\Delta}, & \theta \le NET < \theta + \Delta \\ 1, & NET \ge \theta + \Delta \end{cases}$$

Рассчитывается легко, но имеет разрывную первую производную в точках: $NET = \theta, NET = \theta + \Delta,$ что усложняет алгоритм обучения.

5. **Экспонента:**

$$OUT = e^{-NET} .$$

Применяется в специальных случаях.

6. **SOFTMAX функция:**

$$OUT = \frac{e^{NET}}{\sum_i e^{NET_i}} .$$

Здесь суммирование производится по всем нейронам данного слоя сети. Такой выбор функции обеспечивает сумму выходов слоя, равную единице при любых значениях сигналов NET_i данного слоя. Это позволяет трактовать OUT_i как вероятности собы-

тий, совокупность которых (все выходы слоя) образует полную группу. Это полезное свойство позволяет применить SOFTMAX функцию в задачах классификации, проверки гипотез, распознавания образов и во всех других, где требуются выходы вероятности.

8. **Гауссова кривая** (рис. 10.4,*д*):

$$\hat{V} = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(\quad -m)}{2\sigma^2}}.$$

Применяется в случаях, когда реакция нейрона должна быть максимальной для некоторого определенного значения S .

9. **Линейная функция.**

Y =K S, K =const. Применяется для тех моделей сетей, где не требуется последовательное соединение слоев нейронов друг за другом.

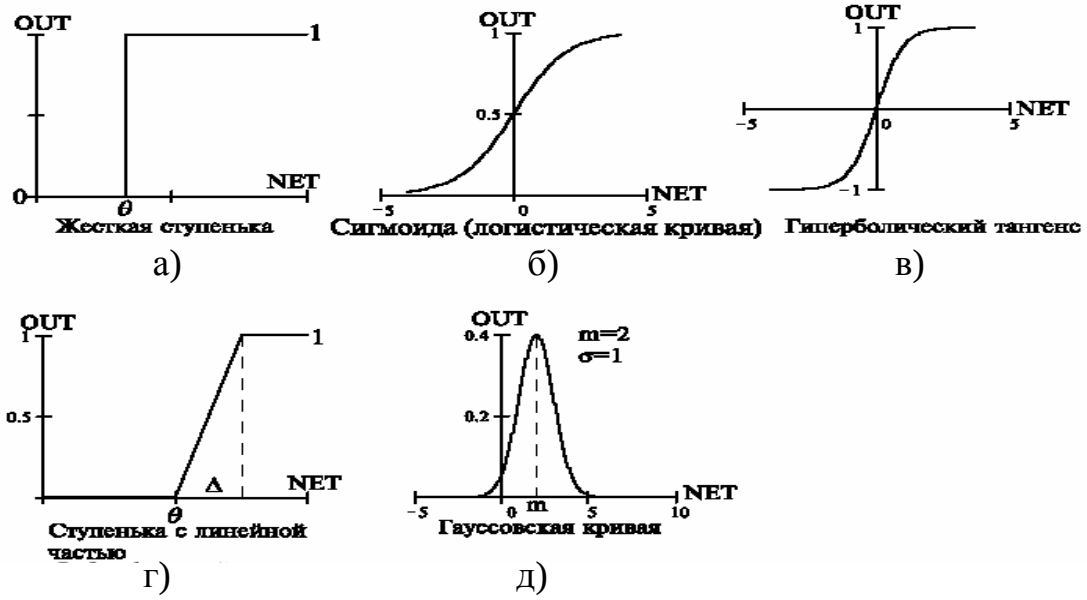


Рис. 10.4 – Виды функций активации

Выбор функции активации определяется:

1. Спецификой задачи.

2. Удобством реализации на ЭВМ, в виде электрической схемы или другим способом.

3. Алгоритмом обучения: некоторые алгоритмы накладывают ограничения на вид функции активации, их нужно учитывать.

Чаще всего вид нелинейности не оказывает принципиального влияния на решение задачи. Однако удачный выбор может сократить время обучения в несколько раз.

6.11 Глава 11. Перспективы развития преобразователей информации

При всех достижениях современных способов преобразования информации существуют ограничения, сдерживающие дальнейшее развития информатики.

Ограничения:

- по плотности размещения,
- по времени преобразования,
- по производительности,
- по объему памяти,
- по способу представления информации.

Выход: поиск и применение новых принципов преобразования информации.

Новые технологии преобразования информации:

- оптические,
- квантовые,
- молекулярные,
- биологические,
- нанотехнологии.

Оптические технологии

В основе – использование в качестве носителя информации оптического излучения.

Достоинства:

- преобразование без затрат энергии
- параллельная обработка информации,
- высокая скорость коммутации,
- многопараметричность (представление информации в зависимости от пространства, времени, длины волны, поляризации),
- широкая полоса частот преобразования (до 10^{13} герц),
- дешевизна материала (песок),

- низкие потери при передаче,
- высокая помехозащищенность,
- высокая технологичность (многослойность без ограничения влияния слоев друг на друга).

Оптическое излучение широко используется для передачи информации по волоконно-оптическим линиям связи (рис. 11.1).

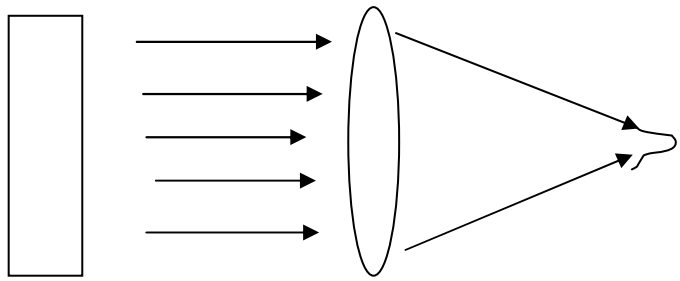


Рис. 11.1 – Преобразование света оптической линзой

Пример преобразования света оптической линзой: выполнение математической операции преобразования Гаусса.

Проблемы, сдерживающие применение в средствах преобразования информации: отсутствуют микроэлектронные средства усиления и управления световыми потоками.

Оптический процессор состоит:

- из линейки светоизлучающих диодов (LED), расположенных на фокальной линии цилиндрической линзы L1;
- оптического транспаранта T с записанной на нем матрицей пропускания T(i, j), строки которой параллельны образующей первой линзы;
- цилиндрической линзы L2, расположенной параллельно столбцам матрицы транспаранта, и собирающей лучи, прошедшие через элементы одной строки, на одном пикселе многоэлементного линейного фотоприемника D.

Входной X и выходной Y вектора связаны линейным преобразованием

$$Y=TX.$$

Пример: оптический процессор Enlight 256 – это DSP-процессор, выполняющий простой набор команд. Свет позволяет процессору выполнять до 8 триллионов операций в секунду (в тысячи раз быстрее DSP-процессора) (рис. 11.2).

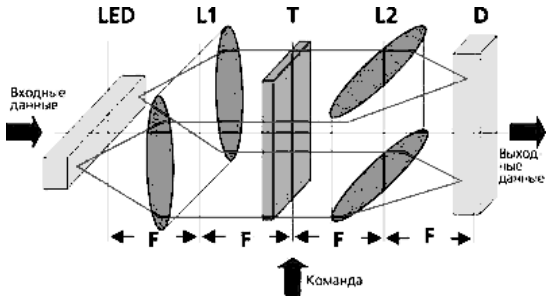


Рис. 11.2 – Оптический процессор Enlight 256

Оптический процессор (рис. 11.3) реализует операции свертки двух изображений для работы устройств ассоциативной памяти и распознавания образов. S – плоский однородный источник света, L1 и L2 – сферические линзы, D – матричный фотоприемник, T1 и T2 – транспаранты, пропускание которых соответствует двум обрабатываемым изображениям. Распределение интенсивности излучения на матричном фотоприемнике пропорционально интегралу свертки.

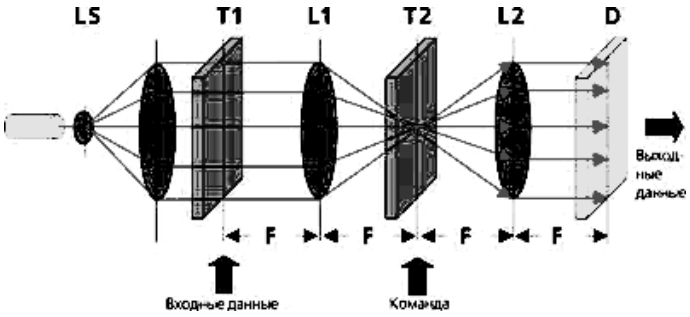


Рис. 11.3 – Схема оптического когерентного процессора

Голографическая обработка информации позволяет сохранить информацию об интенсивности и фазе световой волны, что позволяет повысить объем записываемой информации. Объемная плотность записи информации может превышать величину 10^{11} бит/см^3, а скорость ввода информации с голограмм – несколько гигабит в секунду (рис. 11.4).

Ввод информации осуществляется с помощью управляемого оптического транспаранта. Адресацией при записи-считывании управляет опорный луч. Считываемая информация фокусируется в плоскости многоэлементного матричного фотоприемника D.

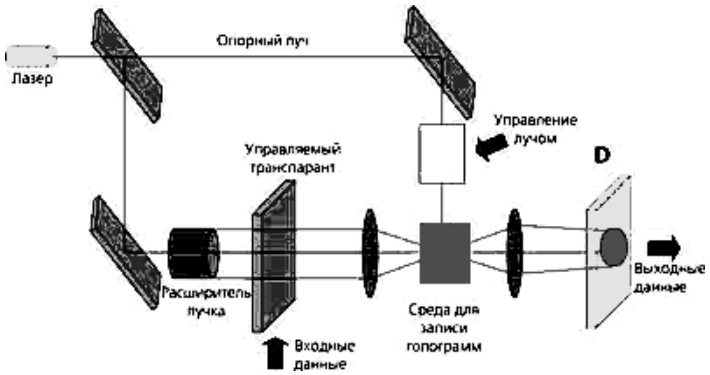


Рис. 11.4 – Оптическая память с объемной голографической средой

Квантовые компьютеры

Квантовый компьютер использует квантово-механические явления (*квантовую интерференцию*), чтобы осуществлять совершенно новый способ обработки информации. В квантовом компьютере фундаментальная единица информации представляется в виде квантового бита (*кубит*), который существует одновременно в состоянии, соответствующем логическим 0 или 1, как классический бит, но также и в состояния *суперпозиции* этих классических состояний. Другими словами, кубит может существовать как ноль как единица, и как одновременно 0 и 1.

Квантовый компьютер обрабатывает кубиты, выполняя последовательность операций квантовыми логическими элементами, каждый из которых представляет собой *унитарное преобразование*, действующее на единичный кубит или пару кубитов. Последовательно выполняя эти преобразования, квантовый компьютер может выполнить сложное унитарное преобразование над всем набором кубитов, приготовленных в некотором начальном состоянии. После этого можно произвести измерение над кубитами, которое и даст конечный результат вычислений.

Квантовый регистр, составляется из физических битов. Квантовый регистр, составленный из трех кубитов, может хранить в любой момент все 8 чисел в квантовой суперпозиции. L кубитов могут хранить 2^L чисел одновременно. Как только регистр приготовлен, в суперпозиции разных чисел можно выполнять математические операции сразу над всеми числами.

Пример: кубиты – это атомы. Лазерные импульсы воздействуют на атомные электронные состояния и заставляют начальную суперпозицию чисел изменяться к некоторой другой суперпозиции. Во время такого изменения обрабатывается каждое число в суперпозиции, и таким образом производятся массовые параллельные вычисления, но только в одном элементе квантового компьютера. Это означает, что квантовый компьютер может всего за один вычислительный шаг выполнять одну и ту же вычислительную операцию над 2^L разных входных числах, закодированных в когерентной суперпозиции L кубитов.

Система из 500 кубитов представляет собой квантовую суперпозицию из 2^{500} состояний. Каждое значение такой суперпозиции классически эквивалентно списку из 500 единиц и нулей. Любая квантовая операция над такой системой будет одновременно воздействовать на 2^{500} состояний. Таким образом, за один тик компьютерных часов квантовая операция вычисляет не одно машинное состояние, как обычные компьютеры, а 2^{500} состояний сразу.

Существует три основных эффекта, реализующих квантовые суперпозиции:

- ядерно-магнитный резонанс;
- ионные ловушки;
- запирание электронов в сверхпроводящих резонаторах.

Квантовые компьютеры (2–3 кубита) уже созданы. Необходимо работать с 10–12 кубитами.

Трудности: явление декоренции, т.е. возникновение ошибок и распадение кубитов при их измерении (переход к детерминированному состоянию). Эти задачи решаются.

Молекулярный компьютер

Молекулярный компьютер представляет устройство, в котором вместо кремниевых чипов используются молекулы и молекулярные ансамбли, которые могут существовать в двух термодинамически устойчивых состояниях. Переводить молекулу из одного состояния в другое (переключать) можно с помощью света, тепла, химических реагентов, электрического и магнитного поля и т.д.

Переключаемые бистабильные молекулы – это наноразмерная двухбитовая система, воспроизводящая на молекулярном уровне функцию классического транзистора.

Память молекулярного компьютера будет основана на тех же принципах, что и переключатели, в её основе – бистабильные молекулярные структуры и их же превращения.

Переключающие молекулярные элементы

Молекулярные транзисторы, память и проводники – три составные части будущего молекулярного компьютера.

Задача – собрать все компоненты в работающее устройство по принципу молекулярного распознавания (самосборка и самоорганизация сложных ансамблей и агрегатов молекул). Этот же принцип лежит в основе происхождения жизни, и именно его использует природа для создания таких сложных структур, как двойная спираль ДНК, жидкие мембраны и глобулярные протеины.

Биологические компьютеры

Достижения в области биологии и генетики привели к возможности создания биологических компьютеров. Гигантской информационной мощностью обладает молекула ДНК. Если обычный компьютер манипулирует сочетаниями значений «0» и «1», то в ДНК имеются четыре базовых состояния (А, Г, Т, Ц), соответственно многократно возрастает число сочетаний. Информационный потенциал ДНК-компьютеров – 10^{21} бит/грамм, то есть один 1 бит/нм^3, тогда как современный компьютер имеет 1 бит на 10^{12} нм^3. ДНК-компьютер способен рассчитывать 10^{19} операций в секунду, а последний суперскоростной компьютер обеспечивает не более 10^{13} операций в секунду. ДНК-компьютеры нецелесообразно использовать для повседневной работы, но их возможности позволяют уже сейчас решать ряд сложных задач.

Применение в вычислительной технике биологических материалов позволит со временем уменьшить компьютеры до размеров живой клетки. Пока это чашка Петри, наполненная спиралями ДНК, или нейроны, взятые у пиявки и подсоединенные к электрическим проводам. По существу, наши собственные клетки – это не

что иное, как *биомашины молекулярного размера*, а примером биокомпьютера, конечно, служит наш мозг.

Современные достижения в области сборки молекул позволяют создавать устройства клеточного размера, которые можно применять для биомониторинга.

7 ГЛОССАРИЙ

2D – структура ЗУ с однокоординатной выборкой слов путем возбуждения линии выборки от дешифратора адреса.

2DM – структура ЗУ (модификация структуры 2D), в которой слова выбираются поэтапно – вначале выбираются «длинные» слова с помощью дешифрации одной части адреса, а затем из них собираются слова нужной разрядности с помощью дешифрации другой части адреса.

3D – структура ЗУ с двухкоординатной выборкой запоминающих элементов на пересечении двух линий выборки, возбуждаемых выходами двух дешифраторов адреса.

Автомат Мура – автомат с памятью, выходные сигналы которого зависят только от состояния автомата.

Адресация абсолютная – адресация, при которой ячейке памяти или внешнему устройству соответствует один-единственный адрес.

Адресация неабсолютная – адресация, при которой ячейке памяти или внешнему устройству соответствует некоторая зона адресов.

Адресное ЗУ – ЗУ, в котором доступ к единицам хранения информации осуществляется по их адресу (местоположению в памяти).

Адресное пространство – диапазон адресов, к которым может обращаться процессор.

Асинхронные установочные входы – входы сброса и установки триггеров, действие которых не зависит от тактирования и доминирует над воздействиями других входов.

Ассоциативное ЗУ (CAM, Content Addressable Memory) – ЗУ, в котором доступ к единицам хранения информации осуществляется не по их адресу, а по специальному признаку (ключу).

совокупность аппаратных и программных средств, унифицирующих процессы обмена между модулями системы.

Интерфейс с общей шиной – интерфейс, в котором адреса ячеек памяти и вешние устройства имеют общее адресное пространство.

Интерфейс с раздельной шиной – интерфейс, в котором для адресов внешних устройств имеется отдельное адресное пространство.

Канал трассировки – свободная зона на кристалле БМК, выделенная для гализации межсоединений ячеек.

Канальный БМК – базовый матричный кристалл, в конструкции которого предусмотрены определенные каналы трассировки.

Код «1 из N» – код, в кодовых комбинациях которого один разряд активен, все остальные пассивны. Кодирование этим способом в английской терминологии именуется ONE, One-Hot Encoding. Активным может считаться значение логической 1 или логического 0.

Код Грея – код, в котором соседние кодовые комбинации отличаются друг от друга только в одном разряде.

Код Хемминга – код, кодовые комбинации которого содержат несколько контрольных разрядов для проверки на четность/нечетность весов определенных групп разрядов. Обладает свойствами не только обнаружения, но и вправления ошибок единичной кратности.

Кодовая комбинация – набор из символов принятого алфавита.

Командный цикл – интервал времени, соответствующий выполнению одной команды программы.

Комбинационная цепь – схема, в которой установившиеся значения выходных сигналов зависят только от текущих значений входных сигналов.

Компаратор (цифровой) – устройство, определяющее отношения между двумя словами.

Конвейеризация – способ повышения частоты тактирования в тракте обработки данных, для реализации которого комбинационные цепи тракта разделяют на ступени.

Контроллер ПДП – контроллер прямого доступа к памяти, устройство, управляющее обменом данными между памятью и внешними устройствами при участии процессора.

Контроль по четности/нечетности – контроль с проверкой четности кодовых комбинаций. Обладает свойством обнаружения ошибок единичной кратности.

Контрольный разряд – дополнительный разряд, вводимый в информационное слово для обеспечения четности/нечетности его веса или веса групп разрядов при контроле по модулю два или с помощью кода Хемминга.

Конфигурируемый логический блок (Configurable Logic Block) – логический блок микросхем программируемой логики, настраиваемый (программируемый) на воспроизведение требуемых функций.

Кратность ошибки – число неверных разрядов в данной кодовой комбинации.

Кратчайшая ДНФ – дизъюнктивная нормальная форма, представленная переключательной функций, содержащая минимальное число конъюнкт термов.

Кэш-память – особо быстродействующая память, хранящая копии и информацию, используемые в текущих операциях обмена с процессором.

Кэш-память наборно-ассоциативного типа – вариант кэш-памяти, с полной ассоциацией и прямым отображением.

Кэш-память с полной ассоциацией – ассоциативная кэш-память с произвольной загрузкой данных.

Кэш-память с прямым размещением – кэш-память, в которой одна или несколько страниц основной памяти строго соответствуют одной строке памяти.

Кэш первого уровня (L1) – внутрипроцессорная кэш-память, размещенная на одном кристалле с процессором.

Кэш второго уровня (L2) – кэш-память, расположенная вне кристалла, на котором размещен процессор. Емкость кэш-памяти второго уровня, как правило, превышает емкость кэш-памяти первого уровня.

ЛИЗМОП – МОП-транзистор с лавинной инжекцией заряда, «плавающий затвор», т. е. изолированная область над каналом, в котором можно создавать или не создавать электрический заряд, отображая логические состояния 1 и 0. Кроме того, может иметь необычный управляющий затвор (варианты «с плавающим затвор «с двойным затвором»).

Магистрально-модульная структура – структура микропроцессорной системы, в которой к одним и тем же шинам подключаются различные модули.

Мажоритарный элемент – логический элемент с нечетным числом входов, входная величина которого определяется тем, какие сигналы (0 или 1) составляют большинство среди входных сигналов.

Маскирование запросов – воздействие на сигналы запросов прерывания, прямого доступа к памяти и др., запрещающее обслуживание этих запросов.

Масочное программирование – запись данных в ПЗУ или задание межсоединений в БМК, осуществляемые при производстве кристаллов методами интегральной технологии (с помощью шаблонов металлизации).

Матричная базовая ячейка – базовая ячейка внутренней области БМК, предназначенная для реализации на ее основе функциональных ячеек.

Машинный цикл – интервал времени, составляющий часть командного цикла, соответствующий в основном обращению процессора к памяти или внешнему устройству и передаче байта (слова) в процессор или из него.

Метастабильное состояние – аномальное состояние триггера, в котором он длительное время находится вблизи равновесного состояния. Вызывается тушением условий предустановки и выдержки информационных сигналов относительно тактирующего или другими факторами, вводящими триггер в режим, близкий к равновесному (симметричному).

Микроконтроллер – однокристальная микроЭВМ, ориентированная на выполнение относительно простых алгоритмов управления техническими объектами и технологическими процессами.

Микропроцессор – реализованное на одном или нескольких кристаллах программно-управляемое устройство, осуществляющее процесс обработки формации и управление им.

Микропроцессорный комплект БИС – набор микросхем, пригодных для совестного применения при построении микропроцессорной системы.

Микропроцессорная система – система, в которой реализован законченный процесс выполнения заданной программы, содержащая в качестве основных блоков (модулей) процессор, память, внешние устройства и интерфейсные схемы.

Минимальное кодовое расстояние – минимальное кодовое расстояние между двумя любыми кодовыми комбинациями, принадлежащими данному коду.

Минимизация логических функций – такое преобразование логических функций, которое упрощает их в смысле заданного критерия.

Модуль счета – число состояний, которое может иметь счетчик.

Мультиплексор – схема, передающая на выход одну из нескольких входных величин под управлением адресующего кода.

Нейрон – логический элемент, состоящий из входов, средств перемножения входных значений на синаптические коэффициенты, сумматора и функции активации,

Однофазная синхронизация – система синхронизации, в которой на элементы памяти (триггеры) подаются одни и те же тактирующие сигналы.

Операция монтажной логики – логическая операция, реализуемая путем объединения в одной точке выходов нескольких логических элементов с коллектором или эмиттером.

Организация ЗУ – параметр ЗУ, выражаемый произведением максимального возможного числа хранимых слов на их разрядность.

Основная память – память, работающая в режиме оперативного обмена данными с процессором и, в отличие от кэш-памяти, хранящая весь объем, требуемых для этого данных. В ЭВМ в качестве основной используется, как правило, память динамического типа.

Открытый коллектор – тип выходной цепи логических элементов, допускающих подключение к магистрали. Может быть использована для реализации операций монтажной логики.

Параллельный периферийный адаптер – это устройство, обслуживающее обмен параллельными данными между процессором и внешними устройствами.

Перекрестная помеха – помеха, порождаемая взаимным влиянием сигнальных линий.

Полиномиальный счетчик – сдвигающий регистр с линейными обратными связями, т. е. связями, реализованными с

помощью элементов сложения по модулю два. Используются в качестве генераторов псевдослучайных последовательностей.

Третье состояние – состояние «отключено», в котором выход логического элемента практически отсоединяется от нагрузки. Элементы с тремя состояниями выхода (0, 1 и «отключено») могут подключаться к магистралям систем с магистрально-модульной структурой.

Триггер – элементарный автомат, содержащий элемент памяти с емкостью один бит и схему управления записью в этот элемент памяти.

Триггер асинхронный – триггер, воспринимающий воздействия информационных входных сигналов непосредственно в моменты их изменений.

Триггер-защелка – триггер типа D, имеющий режим «прозрачности» при одном уровне управляющего сигнала и режим хранения при другом.

Триггер синхронный – тактируемый триггер, воспринимающий воздействия информационных сигналов только при разрешении их приема специальным тестовым сигналом.

Триггер, управляемый уровнем – триггер, для которого сигналом разрешения приема информации является тот или иной уровень управляющего (тактирующего) сигнала. Такой триггер называют также синхронным триггером со статическим управлением.

Триггер, управляемый фронтом – триггер, для которого сигналом разрешения приема информации является перепад управляющего (тактирующего) сигнала. Такой триггер называют также синхронным триггером с динамическим управлением.

Триггер D – синхронный триггер с одним информационным входом, принимающий состояние, соответствующее входному сигналу, по разрешению тактирующего сигнала.

Триггер JK – триггер, имеющий информационные входы установки и сброса, а также режим счетного триггера.

Триггер RS – триггер, имеющий информационные входы установки и сброса.

Универсальный логический модуль – устройство, воспроизводящее любую функцию заданного числа аргументов.

Фиксированный приоритет – приоритет, присвоенный данному запросу (входу) и не изменяющийся в процессе работы системы.

Флэш-память – высококачественная перепрограммируемая память на элементах типа EEPROM, в которой стирание данных производится электрическими сигналами для всего кристалла либо для отдельных блоков (симметричных или несимметричных).

Функция активации – это функция принятия решения в нейронах.

Функции возбуждения триггера – функции, определяющие такие воздействия на триггеры автомата, которые переводят автомат из одного состояния в другое согласно требуемому графу переходов.

Функция прозрачности – вспомогательная функция, используемая при синтезе сумматоров и некоторых других устройств, в которых используются сигналы переноса. Принимает единичное значение для тех разрядов и групп разрядов, на выходах которых сигнал переноса возникает только при наличии входного переноса.

Цикл ЗУ – минимальный интервал времени между соседними однотипными обращениями к ЗУ. Соответственно по типу обращения различают цикл чтения, записи и др.

Циклический (круговой) приоритет – порядок обслуживания запросов (прерывания, прямого доступа к памяти и др.), для которого источники запросов равноправны. Равноправность источников запросов достигается і, что их приоритеты изменяются при работе системы – после обслуживания источник получает низший приоритет, который постепенно повышается по мере обслуживания других источников запросов.

8 ЛИТЕРАТУРА

1. Замятин Н.В. Организация ЭВМ и систем: Учебное пособие. – Томск: ТМЦДО, 2005.

2. Коган Б.М. Электронные вычислительные машины и системы. – М.: Энергоиздат, 1995.

3. Ровдо А.А. Микропроцессоры от 8086 до Pentium III Xeon и AMD-K6-3. – М., 2000. – 592 с.

4. Фролов А.В., Фролов Г.В. Аппаратное обеспечение IBM PC. – М.: Мифи-диалог, 1992.

5. Лю-ю жен, Гибсон. Микропроцессоры семейства 8086/88. – М.: Мир, 1998.

6. Морс С. Архитектура микропроцессора 80286. – М.: Мир, 1992.

7. Григорьев В.П. Архитектура микропроцессора i-486. – М., 1993.

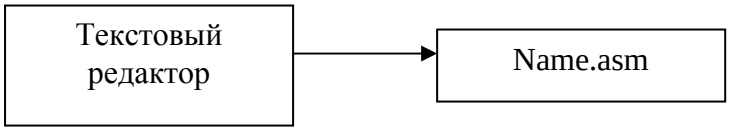
8. Уоссерман Ф. Нейрокомпьютерная техника.– М.: Мир, 1990.

ПРИЛОЖЕНИЕ 1
Процесс разработки программы на Ассемблере

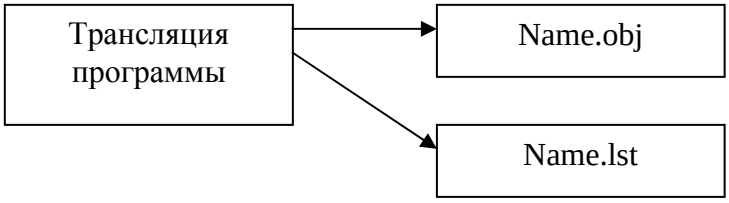
Для разработки программы на ассемблере (контрольная работа №2 и контрольно-лабораторный практикум) необходимо следовать общей схеме процесса разработки, приведенной на рис. П.1.1.

На первом этапе для ввода и редактирования исходного текста программы с расширением asm, можно использовать любой текстовый редактор (far, notepack).

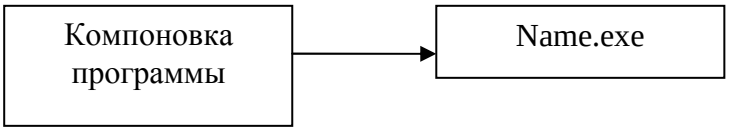
1. Ввод исходного текста программы:



2. Создание объектного модуля:



3. Создание загрузочного модуля:



4. Отладка программы:

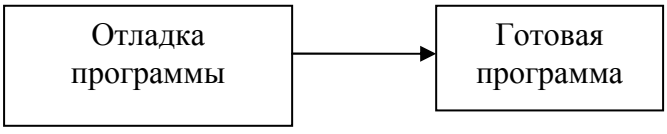


Рис. П.1.1 – Процесс разработки программы на ассемблере

На втором этапе необходимо выполнить процедуру трансляции посредством транслятора tasm.exe и создать объектный модуль с расширением obj. Объектный модуль включает в себя представление исходной программы в машинных кодах и дру-

гую информацию для отладки программы. Для получения информации о tasm.exe необходимо запустить его без параметров. При наличии ошибок необходимо, просматривая номера строк из сообщений транслятора, устранить ошибки и вновь выполнить трансляцию.

На третьем этапе после устранения ошибок и получения объектного модуля помощью компоновщика tlink.exe выполняется компоновка программы и создается загрузочный модуль с расширением типа com или exe. Главная цель этого этапа преобразовать код и данные в объектных файлах в их перемещаемое выполняемое отображений в машинных кодах.

Лучше всего для отладки и просмотра выполнения программы использовать отладчик turbo debugger (td.exe).

ПРИЛОЖЕНИЕ 2
Отладчик turbo debugger

Обязательным этапом процесса разработки программного модуля является его *отладка*. Специфика программ на ассемблере состоит в том, что они интенсивно работают с аппаратными ресурсами компьютера. Это обстоятельство заставляет программиста постоянно отслеживать содержимое определенных регистров и областей памяти. Человеку трудно следить за этой информацией с большой степенью детализации. Поэтому для локализации логических ошибок в программах используют специальный тип программного обеспечения – программные *отладчики*.

Лучше всего использовать отладчик Turbo Debugger (TD), разработанный фирмой Borland International. Он представляет собой оконную среду отладки программ на уровне исходного текста и позволяет решить две главные задачи:

1) определить место логической ошибки;

2) определить причину логической ошибки.

Перечислим некоторые возможности TD:

1) выполнение *трассировки* программы в прямом направлении, то есть последовательное исполнение программы, при котором за один шаг выполняется одна машинная инструкция;

2) выполнение *трассировки* программы в обратном направлении, то есть выполнение программы по одной команде, но в обратном направлении;

3) просмотр и изменение состояния аппаратных ресурсов микропроцессора во время покомандного выполнения программы.

Это позволяет определить место и источник ошибки в программе. Нужно сразу оговориться, что TD не позволяет вносить исправления в исходный текст программы. После определения причины ошибочной ситуации можно, при необходимости, не завершая работу отладчика, внести исправления прямо в машинный код и запустить программу на выполнение. После завершения работы отладчика эти изменения не будут сохранены, и нужно внести их повторно, но уже в исходный текст, и повторно создать загрузочный модуль.

Запуск отладчика удобнее производить из командной строки с указанием исполняемого модуля программы, которая подлежит отладке:

td имя_исполняемого_модуля.

При правильном выполнении перечисленных выше действий откроется окно отладчика TD под названием Module (Рис. 1). В этом окне вы видите исходный текст программы с расширением asm. (Если Turbo Debugger не найдет файл с расширением ASM, он не сможет показать окно с исходным текстом программы.) Здесь вы видите так называемый *курсор выполнения* (в виде треугольника (1)). Он указывает на первую команду, подлежащую выполнению. Этой команде предшествует имя метки. Это так называемая *точка входа* в программу. Если вы внимательно посмотрите конец исходного текста программы, то увидите, что это же имя записано в качестве операнда в заключительной директиве END. Это единственный способ сообщить загрузчику ОС о том, где в исходном тексте программы расположена точка входа в нее. В более сложных программах обычно вначале могут идти описания процедур, макрокоманд, и в этом случае без такого явного указания на первую исполняемую команду вам не обойтись. Просмотреть текст программы можно с помощью клавиш управления курсором, <PgUp>, <PgDn>, <Home>, <End>.

Кроме окна Module, существует ещ[cedilla] ряд окон, которые позволяют управлять отладкой программы. Основную часть экрана отладчика обычно занимают одно или несколько окон. В каждый момент времени активным может быть только одно из них. Активизация любого окна производится щелчком мышью в любой видимой точке окна.

Управление работой отладчика ведется с помощью системы меню. Имеется два типа таких меню:

1. *Глобальное меню* (рис. П.2.1 (1)) – находиться в верхней части экрана и доступно постоянно. Вызов меню осуществляется нажатием клавиши F10, после чего следует выбрать нужный пункт меню.

2. *Локальное меню* (рис. П.2.1 (2)) – для каждого окна отладчика можно вызвать его собственное меню, которое учитывает особенности этого окна. Вызвать данное меню можно щелкнув в окне правой кнопкой мыши (либо клавишей Alt-F10).

Рис. П.2.1 – Окно Module

Специфика программ на ассемблере в том, что делать выводы о правильности их функционирования можно, только отслеживая работу на уровне микропроцессора. При этом нас интересует, прежде всего, то, как программа использует микропроцессор и изменяет состояние его ресурсов и компьютера в целом. Для получения информации о состоянии микропроцессора последовательно нажмите комбинацию клавиш <Alt+V>+<C>, выбирающую команду View / CPU (просмотр окна центрального процессора). Нажмите <F5> для распахивания окна на весь экран (рис. П.2.2 (1)). Кроме чисто внешних деталей, между этим и предыдущим окном существует принципиальное различие. В окне исходного кода, также называемом окном просмотра модуля, вы видите копию текста программы. В CPU-окне вы находитесь непосредственно в памяти, видите действительные значения расположенных там байтов.

Это окно отражает состояние микропроцессора и состоит из 5 подокон:

- окна (рис. П.2.2 (1)) с исходной программой в дизассемблированном виде. Это та же самая программа, что и в окне Module, но уже в машинном виде. Пошаговую отладку можно производить прямо в этом окне; строка с текущей командой подсвечивается;

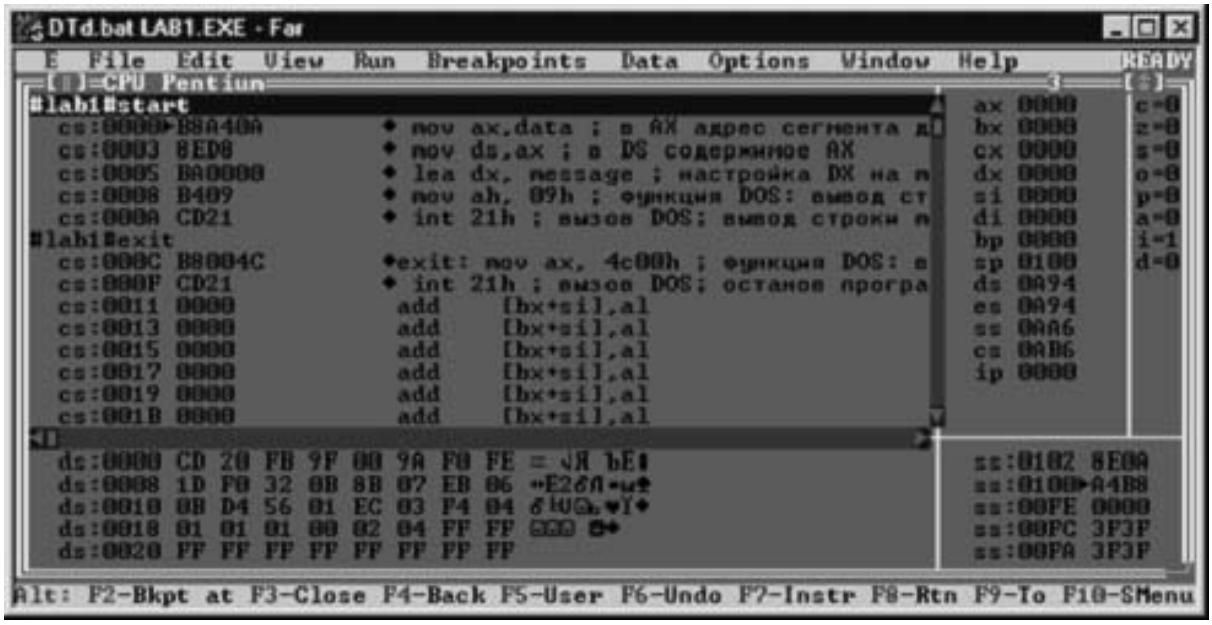


Рис. П.2.2 – Окно CPU

- Registers (рис. П.2.2 (2)) – окна регистров микропроцессора, отражающего текущее содержимое регистров. Заметьте, что по умолчанию отображаются регистры только i8086. Для того чтобы воспользоваться всеми регистрами i486 или Pentium, нужно задать режим их отображения. Для этого щелкните правой кнопкой мыши в области подокна регистров для вызова локального меню. В меню выберите команду Registers 32-bit – Yes;
- окна флагов (рис. П.2.2 (3)), которые отражает текущее состояние флагов микропроцессора в соответствии с их мнемоническими названиями;
- окна стека Stack (рис. П.2.2 (4)), отражающего содержимое памяти, выделенной для стека. Адрес области стека определяется содержимым регистров S3 и SP;
- окна с дампом памяти Dump (рис. П.2.2 (5)), отражающего содержимое области памяти по адресу, который формируется из компонентов, указанных в левой части окна. В окне можно увидеть содержимое произвольной области памяти. Для этого нужно в локальном меню, вызываемом по щелчку правой кнопки мыши, выбрать нужную команду.

Замечание. Некоторые из этих подокон можно использовать по-другому. Совместить возможности окон Module и CPU можно посредством пункта глобального меню View, в котором необхо-

димо выбрать нужные имена подокон CPU (эти имена мы указали при перечислении подокон выше).

Для передвижения выделенного прямоугольного курсора по различным командам используйте клавиши <Up> и <Down>. Инструкции, принадлежащие вашим программам, отмечаются ромбами. Для передвижения курсора из одного подокна в другое нажимайте клавишу <Tab>. Переходя между подокнами, можно изменять значения регистров и модифицировать байты памяти. (В данный момент не вносите никаких изменений.)

Нажимайте клавишу <Tab> до тех пор, пока прямоугольный курсор снова не проявится в большей секции. Для изменения вида этого окна нажмите <Alt+F10> или выберите команду Mixed (для этого нажмите клавишу <M> или переместите курсор на команду Mixed и нажмите <Enter>). Эту же команду можно выбрать, просто нажав <Ctrl+M>. Она имеет три установки: No, Yes и Both, которые позволяют изменять тип просмотра вашей программы:

1. No показывает дизассемблированные машинные коды, аналогичные командам языка ассемблера. Это удобно для просмотра кода в случае, если у вас нет соответствующего ASM-файла. Многие программисты предпочитают использовать это представление, так как по сравнению с другими оно наиболее практично.

2. Yes показывает исходный текст вашей программы вместе с дизассемблированным машинным кодом. Используется для отображения строк в языках программирования высокого уровня вместе с соответствующим им откомпилированным машинным кодом. Как правило, мы не пользуемся этой установкой при просмотре ассемблерных программ.

3. Both устанавливается по умолчанию и, возможно, является наилучшим способом просмотра в CPU-окне, показывая в левой колонке байты машинного кода вместе с создающими их строками исходного текста. Пустые строки при этом не отображаются.

Запустить программу на выполнение в отладчике можно в одном из четырех режимов:

- режим безусловного выполнения;

- выполнение по шагам;
- выполнение до текущего положения курсора;
- выполнение с установкой точек прерывания.

Режим безусловного выполнения целесообразно применять, когда требуется посмотреть на общее поведение программы. Для запуска программы в этом режиме необходимо нажать клавишу F9. В точках, где необходимо ввести данные, отладчик, в соответствии с логикой работы применяемого средства ввода, будет осуществлять определенные действия. Аналогичные действия отладчик выполняет для вывода данных. Для просмотра или ввода этой информации можно открыть окно *пользователя* (Window|User screen), например, нажав клавиши Alt+F5.

Выполнение по шагам применяется для детального изучения работы программы. В этом режиме вы можете выполнить программу по командам. При этом можно наблюдать результат исполнения каждой команды. Для активизации этого режима нужно нажать клавиши F7 (Run|Trace into) или F8 (Run|Step over). Обе эти клавиши активизируют пошаговый режим; отличие их проявляется в том случае, когда в потоке команд встречаются команды перехода в процедуру или на прерывание. При использовании клавиши F7 отладчик осуществит переход на процедуру или прерывание и выполнит их по шагам. Если же используется клавиша F8, то вызов процедуры или прерывания отрабатывается как одна обычная команда и управление возвращается следующей команде программы. *Выполнение до текущего положения курсора* позволяет выполнить программу по шагам, начиная с произвольного места программы. Этот режим целесообразно использовать в том случае, если вас интересует только правильность функционирования некоторого участка программы. Для активизации этого режима необходимо установить курсор на нужную строку программы и нажать клавишу F4. Программа начнет выполнение и остановится на отмеченной команде, не выполнив ее.

Выполнение с установкой точек прерывания позволяет выполнить программу с остановкой ее в строго определенных *точках прерывания* (breakpoints). Перед выполнением программы необходимо установить эти точки в программе, для чего следует перейти в нужную строку и нажать клавишу F2. Выбранные

строки подсвечиваются другим цветом. Установленные ранее точки прерывания можно убрать – для этого нужно повторно выбрать нужные строки и нажать клавишу F2. После установки точек прерывания программа запускается на выполнение клавишей F9. На первой точке прерывания программа остановится. Теперь можно посмотреть состояние микропроцессора и памяти, а затем продолжить выполнение программы. Сделать это можно или в пошаговом режиме, или выполнить программу до следующей точки прерывания.

Прервать выполнение программы в любом из этих режимов можно, нажав Ctrl+F2.

Работа с меню

Команды глобальных меню Turbo Debugger выводятся в верхней части экрана в строке меню. Для открытия меню Turbo Debugger, нажмите F10, с помощью стрелой переместитесь в нужному пункту и нажмите Enter. После F10 для перехода к нужному пункту можно также нажать его подсвеченную букву, либо сразу нажмите Alt+буква (без F10). Системное меню выбирается по Alt+пробел. Меню открывается также щелчком «мышью» на соответствующем пункте.

Окна Turbo Debugger

Для вывода информации об отлаживаемой программе в Turbo Debugger используется набор окон. Для облегчения отладки служат команды управления окнами, которые находятся в меню Window и System. Каждое открываемое окно имеет номер, указанный в его правом верхнем углу. Нажатием клавиши Alt в сочетании с номером окна вы можете активизировать любое из первых 9 окон. Список открытых окон содержится в нижней половине меню Window. Чтобы открыть конкретное окно, нажмите в меню Window цифру номера окна. Если окон больше 9, в этом меню выводится команда Window Pick, выводящая меню окон.

Клавиша F6 (или команда Window Next) позволяет циклически перемещаться по открытым на экране окнам. Окно может иметь несколько областей. Для перемещения между областями используйте клавиши Tab или Shift+Tab, либо Window Next. Кур-

сор в областях перемещается с помощью стандартных клавиш перемещения курсора.

При открытии нового окна оно выводится в месте текущего расположения курсора. Переместить его в другое место можно с помощью команды Window Size/Move и клавиш стрелок, либо сразу нажмите Shift и сдвигайте окно стрелками. Для быстрого увеличения или уменьшения окна выберите Window Zoom (F5) или щелкните «мышью» на кнопке минимизации/максимизации в верхнем правом углу окна.

Если окно по ошибке закрыто, то вернитесь в последнее окно можно с помощью команды Window Undo Close (Alt+F6). Когда программа затирает своим выводом экран операционной среды (при выключенном переключении экрана), можно очистить его с помощью System Repaint Desktop. Для возврата к используемой по умолчанию схемы окон Turbo Debugger выберите System Restore Standard.

Каждое окно Turbo Debugger имеет специальное оперативное меню SpeedMenu, содержащее команды, относящиеся к данному окну. Области окон также могут иметь свои меню. Для доступа к SpeedMenu активного окна или области вы можете нажать в окне правую кнопку «мыши», либо нажать клавиши Alt+F10, либо нажать Ctrl и подсвеченную букву команды SpeedMenu (для этого должно быть разрешено действие команд-сокращений).

Окна меню View

Меню View является точкой входа в большинство окон Turbo Debugger. Перечислим их кратко. С помощью команды View Another вы можете дублировать на экране окна Dump, File и Module.

Окно Breakpoints

Используется для установки, модификации или удаления точек останова. Точка останова определяет то место в программе, где отладчик приостанавливает выполнение программы. Это окно имеет две области. Справа перечислены условия и действия точек останова, слева – все точки останова.

Окно Stack

Показывает текущее состояние программного стека. Первая вызванная функция показывается в нижней части окна, а выше ее – каждая последующая. Подсвечивая эти функции и нажимая Ctrl+I вы можете проверять исходный код. Кроме того, можно открыть окно Variables и вывести все локальные переменные и аргументы функции (Ctrl+L).

Окно Log

Выводит содержимое журнала сообщений с прокручиваемым списком сообщений и информацией, сгенерированной при работе с отладчиком. Это окно можно также использовать для получения информации об использовании памяти, модулях и оконных сообщения приложения Windows.

Окно Watches

Показывает значения переменных и выражений. Введя в это окно выражения, вы можете отслеживать их значения при выполнении программы. Окно добавляется с помощью клавиш Ctrl+W при установке курсора на переменной в окне Module.

Окно Variables

Выводит все переменные в данном контексте программы. В верхней области окна перечисляются глобальные переменные, а в нижней – локальные. Это полезно использовать для поиска функции или идентификатора, имени которых вы точно не помните.

Окно Module

Одно из важнейших окон Turbo Debugger, показывающее исходный код отлаживаемого программного модуля (включая DLL). Модуль должен компилироваться с отладочной информацией.

Окно File

Выводит содержимое любого файла на диске. В нем можно просматривать шестнадцатиричные байты или текст ASCII и искать нужные байтовые последовательности.

Окно CPU

Выводит текущее состояние процессора. Окно имеет 6 областей, где выводятся дизассемблированные инструкции, селекторы Windows (только в TDW), шестнадцатиричные данные, стек в шестнадцатиричном виде, регистры ЦП и флаги процессора. Это окно полезно использовать при отладке программ на ассемблере или просмотре точно последовательности инструкций.

Окно Dump

Выводит в шестнадцатиричном виде содержимое любой области памяти (аналогично области окна CPU). Команды SpeeMenu этого окна позволяют вам модифицировать данные и работать с блоками памяти.

Окно Registers

Показывает содержимое регистров (в области регистров) и флагов ЦП (в области флагов). С помощью команд SpeedMenu вы можете изменить их значения.

Окно Numeric Processor

Показывает текущее состояние сопроцессора и имеет три области: содержимого регистров с плавающей точкой, значений флагов состояния и значений управляющего флага. Это позволяет вам диагностировать проблемы в использующих сопроцессор подпрограммах. См. файл TD_ASM.TXT.

Окно Execution History

Выводит последние выполненные машинные инструкции или исходные строки программы, номер строки исходного кода и следующую выполняемую инструкцию или строку кода. Используется для обратного выполнения.

Окно Hierarchy

Выводит на экран дерево иерархии всех используемых текущим модулем классов. Имеет область списка классов и дерева иерархии. Это окно показывает взаимосвязь используемых в модуле классов.

Окно Windows Messages

Показывает список оконных сообщений программы Windows. Области этого окна показывают задание режима отслеживания сообщений, тип перехватываемых сообщений и перехваченные сообщения.

Окно Clipboard

Буфер Clipboard отладчика используется для для вырезания и вставки элементов из одного окна отладчика в другое. Оно показывает вырезанные элементы и их типы. Скопированные в буфер элементы динамически обновляются.

Окна Inspector

Выводят текущее содержимое выбранной переменной. Его можно открыть с помощью команды Data Inspect или Inspect меню SpeedMenu. Закрывается оно обычно по Esc или щелчком «мышью» на блоке закрытия. При последовательном открытии нескольких окон Inspector нажатием Alt+F3 или командой Window Close вы можете закрыть сразу все эти окна. Окна Inspector выводят простые скалярные величины, указатели, массивы, объединения, структуры, классы и объекты. Выбором команды Inspect в этом окне вы можете создать дополнительные окна Inspector.

Экран пользователя

Экран пользователя показывает полный экран вывода вашей программы. Этот экран имеет такой же вид, как при выполнении программы без Turbo Debugger. Чтобы переключиться в этот экран, выберите команду Window User Screen. Для возврата в экран отладчика нажмите любую клавишу.

Некоторые окна позволяют вам начать набор нового значения, не выбирая сначала команду SpeedMenu. Выбор по набору обычно применяется к наиболее часто используемым командам SpeedMenu.

Получение справочной информации

Turbo Debugger предлагает несколько способов получения в ходе отладки справочной информации. С помощью клавиши F1

вы можете получить доступ к развитой контекстной справочной системе. По данной клавише на экран выводится список тем, из которых вы можете выбрать необходимую.

Индикатор активности в левом правом углу экрана всегда показывает текущее состояние. Например, если курсор находится в окне, в индикаторе активности выводится READY. Если выводится меню, в нем указывается MENU, а если вы находитесь в диалоговом окне PROMPT. Если вы, запутаетесь и не можете понять, что происходит в отладчике, взгляните на индикатор активности.

В строке состояния в нижней части экрана всегда дается краткая информация об используемых клавиатурных командах. При нажатии клавиши Alt или Ctrl данная строка изменяется. Когда вы находитесь в системе меню, эта строка предлагает вам оперативное описание текущей команды меню.

Оперативная помощь

В отладчик встроен контекстно-зависимый оперативный справочник. Он доступен как при работе в системе меню, так и при выводе сообщения об ошибке или подсказки. Для вывода справочного экрана с информацией, относящийся к текущему контексту (окну или меню) нажмите клавишу F1. При наличие «мыши» вы можете вывести справочный экран, выбрав F1 в строке состояния. Некоторые справочные экраны содержат подсвеченные слова, которые позволяют вам получить дополнительную информацию по данной теме. Для перемещения к нужным ключевым словам используйте клавиши Tab или Shift+Tab и нажмите клавишу Enter. Для перемещения к первому или последнему слову на экране используйте клавиши Home и End. Доступ к оперативным справочным средствам можно получить также с помощью команды Help из строки меню (оперативные клавиши Alt+H).

Если нужно вернуться к предыдущему справочному экрану, нажмите клавиши Alt+F1 или выберите команду Previous из меню Help. В справочной системе для просмотра последних 20 экранов можно пользоваться клавишей PgUp (клавиша PgDn работает, когда вы находитесь в группе связанных экранов). Для доступа к индексному указателю справочной системы нажмите Shift+F1

(или F1 в справочной системе) или выберите команду Index в меню Help. Для получения информации о самой справочной системе выберите в меню Help команду Help Help. Для выхода из справочной системы нажмите клавишу Esc.

При работе в отладчике в нижней части экрана выводится краткая справочная строка. В этой строке состояния кратко описаны клавиши или команды меню для текущего контекста.