Департамент образования Вологодской области
бюджетное профессиональное образовательное учреждение
Вологодской области
«ВОЛОГОДСКИЙ СТРОИТЕЛЬНЫЙ КОЛЛЕДЖ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к практическим работам

ПО ДИСЦИПЛИНЕ ОП.01. ОСНОВЫ АРХИТЕКТУРЫ, УСТРОЙСТВО И ФУНКЦИОНИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Специальность 09.02.04 Информационные системы (по отраслям)

2017г.

Рассмотрено на заседании предметной цикловой комиссии общепрофессиональных, специальных дисциплин и дипломного проектирования по специальностям 08.02.01 «Строительство и эксплуатация зданий и сооружений», 08.02.07 «Монтаж и эксплуатация внутренних сантехнических устройств, кондиционирования воздуха и вентиляции», 43.02.08 «Сервис домашнего и коммунального хозяйства» и рекомендована для внутреннего использования.

Данные методические указания предназначены для студентов обучающихся по специальности 09.02.04 Информационные системы (по отраслям) БПОУ ВО «Вологодский строительный колледж» при выполнении практических работ при изучении дисциплины ОП.01. Основы архитектуры, устройство и функционирование вычислительных систем.

Настоящие методические указания включают в себя краткий теоретический материал, практические задачи, указания к их выполнению.

Авторы:
Н.Л.Ингеройнен - преподаватель
Т. А. Габриэлян - преподаватель
Н. А. Исакова - преподаватель

Содержание

Введение

В методических указаниях представлены практические задания для выполнения их в тетради или на компьютере по всем разделам рабочей программы и календарно-тематического плана по дисциплине ОП.01. Основы архитектуры, устройство и функционирование вычислительных систем.

Задания разделены по разделам курса дисциплины: «Информационно-логические основы ЭВМ», «Архитектура и принципы работы основных логических блоков вычислительных систем», «Архитектура учебной ЭВМ». Раздел 1. Базовые сведения об ЭВМ» практических работ не содержит.

В методических указаниях предложены задания для самостоятельного закрепления новых знаний и умений. Методические указания используются для стимулирования самостоятельного изучения нового материала.

ТЕХНИКА БЕЗОПАСНОСТИ
в компьютерном классе

Находясь за компьютером, рекомендуется периодически отдыхать, отвлекаться от экрана монитора, смотреть в окно, однако во время работы надо быть предельно внимательным.

Во избежание несчастного случая, поражения электрическим током, поломки оборудования, рекомендуется выполнять следующие правила:

1) не входить в помещение, где находится вычислительная техника без разрешения преподавателя;
2) не включать без разрешения оборудование;
3) при несчастном случае, или поломке оборудования позвать преподавателя; знать, где находится пульт выключения оборудования (выключатель, красная кнопка, рубильник);
4) не трогать провода и разъемы (возможно поражение электрическим током);
5) не допускать порчи оборудования;
6) не работать в верхней одежде;
7) не прыгать, не бегать (не пылить);
8) не шуметь.

Строго запрещается:

1) трогать разъемы соединительных кабелей;
2) прикасаться к питающим проводам и устройствам заземления;
3) прикасаться к экрану и тыльной стороне монитора;
4) включать и отключать аппаратуру без указания преподавателя;
5) работать во влажной одежде и влажными руками;
6) класть диск, книги, тетради на монитор и клавиатуру.

Работать следует на расстоянии 60-70 см, допустимо не менее 50 см, соблюдая правильную посадку, не сутулясь, не наклоняясь; учащимся, имеющим очки для постоянного ношения, - в очках.

Уровень глаз при вертикальном расположении экрана должен приходиться на центр экрана или 2/3 его высоты. Оптимальное расстояние глаз учащихся до экрана монитора должно быть в пределах 0,6 - 0,7 м, допустимое - не менее 0,5 м.

Нельзя работать при недостаточном освещении и при плохом самочувствии. Все задания выполнять только **с разрешения преподавателя.**

ЧЕМ ОПАСЕН ДЛЯ НАС КОМПЬЮТЕР	**ЧЕМ ОПАСНЫ МЫ ДЛЯ КОМПЬЮТЕРА**
Компьютер - высокотехнологичное технически хорошо продуманное устройство, но вместе с тем очень опасное. Иногда опасность реальна, а иногда, он незаметно воздействует на Ваше здоровье и психику. <u>Возможные воздействия:</u> • На зрение. Для профилактики следует чаще моргать, периодически отвлекаться (смотреть в окно, вдаль), делать гимнастику для глаз. При наборе текста стараться, как можно меньше смотреть на монитор. • Излучение микроволновое (радиация) и электромагнитное. • Высокое напряжение от 110 до 50000В в неисправных блоках может сохраняться длительное время, поэтому не следует касаться токоведущих частей под напряжением и не использовать компьютер в сырых помещениях. • Воздействие на осанку, неправильная организация рабочего места может привести к быстрому утомлению, искривлению позвоночника (необходима правильная организация рабочего места и времени, гимнастика). • Артрит (при работе с мышкой и клавиатурой более всего задействованы - указательный и средний пальцы, мышцы запястья и предплечья, что может вызвать болезнь суставов) – необходимо распределение нагрузки на все пальцы (десятипальцевый - слепой метод печати). • Ионизированная (наэлектризованная) пыль - сильный канцероген – необходимо проветривать помещение и содержать в чистоте.	Не только компьютерная техника может повредить нашему здоровью, но и мы при несоблюдении элементарных правил гигиены и труда можем испортить оборудование. <u>Возможные повреждения:</u> • Блоков компьютера - это царапины, вмятины, трещины. • Механические повреждения клавиатуры. Стираются надписи на клавишах, от сильного удара клавиши "залипают" (в особенности пробел и ввод). • Механическое повреждение тонкого защитного слоя экрана, касание поверхности экрана пальцем, указкой, ручкой, карандашом... Не желательно протирать экран грубой тканью. • Внутренние механические повреждения, которые могут возникнуть от удара или попадания постороннего предмета вовнутрь. (Категорически запрещается переносить, передвигать блоки компьютера во включенном состоянии.) • Токопроводящая пыль, загрязнения, влага нарушают теплопроводность блоков и могут вывести из строя блоки компьютера. • Крошки, кофе, чай, скрепки... могут попасть в компьютерные блоки и вывести их из строя. • Бумага, положенная на вентиляционные отверстия блоков (монитора) нарушает их тепловой режим. • Частое включение / выключение компьютера создает дополнительную нагрузку на блоки компьютера. Правильная организация рабочего места и рабочего времени, соблюдение правил техники безопасности превратят Ваш

• Компьютерные игры и Интернет иногда перерастают в психологическую (компьютерную) зависимость, поэтому следует развивать чувство самоконтроля.	компьютер в настоящего друга и безопасного помощника.

Тест по технике безопасности.

Проверь себя, насколько ты усвоил «Правила безопасности при работе в кабинете информатики».

Критерии теста:

90-100% (15-14 вопросов) - оценка **5**

80-89% (13-12 вопросов) - оценка **4**

70-79% (11-10 вопросов) - оценка **3**

69% и менее (9 вопросов и менее) - оценка **2**

№1. Разрешено ли входить в кабинет в грязной обуви и верхней одежде?

1) да

2) нет

№2. При появлении запаха гари или странного звука необходимо

1) продолжить работу за компьютером
2) сообщить об этом преподавателю
3) немедленно покинуть кабинет

№3. Как следует нажимать на клавиши?

1) с усилием и ударом
2) плавно

№4. Разрешается ли приносить в кабинет информатики продукты питания и напитки?

1) да, только в том случае, если сильно хочется, есть или пить
2) нет
3) да

№5. Разрешается ли включать или подключать какое-либо оборудование в кабинете информатики без разрешения преподавателя?

1) нет
2) да

№6. Что нужно сделать по окончании работы за компьютером?

1) привести в порядок рабочее место, закрыть окна всех программ, задвинуть кресло, сдать преподавателю все материалы, при необходимости выключить компьютер
2) расписаться в журнале учета работы пользователей за компьютером
3) покинуть кабинет
4) выключить компьютер

№7. Ваши действия при пожаре

1) прекратить работу, под руководством преподавателя покинуть кабинет
2) немедленно покинуть кабинет информатики
3) выключить компьютер и покинуть здание
4) вызвать пожарную охрану

№8. Разрешается ли касаться экрана монитора?

1) нет
2) да

№9. Нужно ли выключать компьютер по окончании работы?

1) да, при необходимости
2) да
3) нет

№10. Какое воздействие на человека оказывают компьютеры?

1) Вызывают усталость и снижение работоспособности
2) Плохо влияет на зрение
3) Человек получает определенную дозу излучения
4) Вызывает расстройство желудка

№11. На каком расстоянии от монитора должен работать студент за компьютером?

1) 15-20 см
2) 50-70 см
3) Меньше 40 см
4) 90-110 см

№12. Каким огнетушителем нужно пользоваться при загорании аппаратуры?

1) Воздушно-пенный огнетушитель
2) Пенный огнетушитель
3) Углекислотный огнетушитель
4) Порошковый огнетушитель

№13. Что обязан сделать студент, если в кабинете вычислительной техники возникла чрезвычайная ситуация?

1) Делать то же, что все делают
2) Спокойно ожидать указания преподавателя
3) Медленно покинуть кабинет
4) Сообщить учителю о ситуации

№14. Если студент неоднократно нарушает инструкцию по технике безопасности, то…

1) Не допускается до занятий
2) Должен пройти снова инструктаж
3) Получает двойку
4) Восстанавливает ущерб, который он причинил

№15. Физические упражнения при работе за компьютером рекомендуется делать через каждые…

1) 25 минут
2) 45 минут
3) 1 час
4) Можно не делать

РАЗДЕЛ 2. ИНФОРМАЦИОННО-ЛОГИЧЕСКИЕ ОСНОВЫ ЭВМ

Практическая работа №1. Перевод чисел из одной системы счисления в другую. Выполнение операций над числами в системах счисления

Система счисления, или просто счисление, или нумерация,— набор конкретных знаков–цифр вместе с системой приемов записи, которая представляет числа этими цифрами.

Цель работы – приобретение навыков выполнения операций в различных системах счисления.

1. Основные понятия систем счисления

Система счисления — это совокупность правил и приемов записи чисел с помощью набора цифровых знаков. Количество цифр, необходимых для записи числа в системе, называют основанием системы счисления. Основание системы записывается в справа числа в нижнем индексе: 5_{10}; 1110110_2; $AF178_{16}$.

Различают два типа систем счисления:

– позиционные, когда значение каждой цифры числа определяется ее позицией в записи числа;

– непозиционные, когда значение цифры в числе не зависит от ее места в записи числа.

Примером непозиционной системы счисления является римская: числа IX, IV, XV и т.д. Примером позиционной системы счисления является десятичная система, используемая повседневно.

Любое целое число в позиционной системе можно записать в форме многочлена:

$$X_S = \{A_n A_{n-1} ... A_2 A_1\} = A_n \cdot S^{n-1} + A_{n-1} \cdot S^{n-2} + ... + A_2 \cdot S^1 + A_1 \cdot S^0,$$

где S — основание системы счисления;

A_n — цифры числа, записанного в данной системе счисления;

n — количество разрядов числа.

Пример. Число 6293_{10} запишется в форме многочлена следующим образом:

$$6293_{10} = 6 \cdot 10^3 + 2 \cdot 10^2 + 9 \cdot 10^1 + 3 \cdot 10^0$$

Десятичная система счисления – в настоящее время наиболее известная и используемая. неправильное название удерживается и поныне.

Десятичная система использует десять цифр —– 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9, а также символы “+” и “–” для обозначения знака числа и запятую или точку для разделения целой и дробной частей числа.

В вычислительных машинах используется двоичная система счисления, её основание — число 2. Для записи чисел в этой системе используют только две цифры — 0 и 1.

Таблица 1. Соответствие чисел, записанных в различных системах счисления

Десятичная	Двоичная	Восьмеричная	Шестнадцатеричная
1	001	1	1
2	010	2	2

3	011	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

2. Правила перевода чисел из одной системы счисления в другую

Перевод чисел из одной системы счисления в другую составляет важную часть машинной арифметики. Рассмотрим основные правила перевода.

1. Для перевода двоичного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 2, и вычислить по правилам десятичной арифметики:

$$X_2 = A_n \cdot 2^{n-1} + A_{n-1} \cdot 2^{n-2} + A_{n-2} \cdot 2^{n-3} + ... + A_2 \cdot 2^1 + A_1 \cdot 2^0$$

При переводе удобно пользоваться таблицей степеней двойки:

Таблица 2. Степени числа 2

n	0	1	2	3	4	5	6	7	8	9	10
2^n	1	2	4	8	16	32	64	128	256	512	1024

Пример. Число 11101000_2 перевести в десятичную систему счисления.

$$11101000_2 = 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 232_{10}$$

2. Для перевода восьмеричного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и соответствующей степени числа 8, и вычислить по правилам десятичной арифметики:

$$X_8 = A_n \cdot 8^{n-1} + A_{n-1} \cdot 8^{n-2} + A_{n-2} \cdot 8^{n-3} + ... + A_2 \cdot 8^1 + A_1 \cdot 8^0$$

При переводе удобно пользоваться таблицей степеней восьмерки:

Таблица 3.4. Степени числа 8

n	0	1	2	3	4	5	6
8^n	1	8	64	512	4096	32768	262144

Пример. Число 75013_8 перевести в десятичную систему счисления.

$$75013_8 = 7 \cdot 8^4 + 5 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 + 3 \cdot 8^0 = 31243_{10}$$

3. Для перевода шестнадцатеричного числа в десятичное необходимо его записать в виде многочлена, состоящего из произведений цифр числа и

соответствующей степени числа 16, и вычислить по правилам десятичной арифметики:

$$X_{16} = A_n \cdot 16^{n-1} + A_{n-1} \cdot 16^{n-2} + A_{n-2} \cdot 16^{n-3} + ... + A_2 \cdot 16^1 + A_1 \cdot 16^0$$

При переводе удобно пользоваться таблицей степеней числа 16:

Таблица 3. Степени числа 16

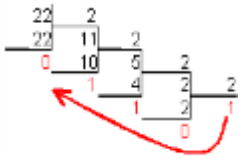
n	0	1	2	3	4	5	6
16^n	1	16	256	4096	65536	1048576	16777216

Пример. Число $FDA1_{16}$ перевести в десятичную систему счисления.

$$FDA1_{16} = 15 \cdot 163 + 13 \cdot 16^2 + 10 \cdot 16^1 + 1 \cdot 16^0 = 64929_{10}$$

4. Для перевода десятичного числа в двоичную систему его необходимо последовательно делить на 2 до тех пор, пока не останется остаток, меньший или равный 1. Число в двоичной системе записывается как последовательность последнего результата деления и остатков от деления в обратном порядке.

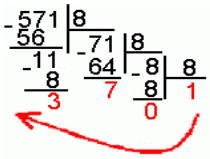
Пример. Число 22_{10} перевести в двоичную систему счисления.



$$22_{10} = 10110_2$$

5. Для перевода десятичного числа в восьмеричную систему его необходимо последовательно делить на 8 до тех пор, пока не останется остаток, меньший или равный 7. Число в восьмеричной системе записывается как последовательность цифр последнего результата деления и остатков от деления в обратном порядке.

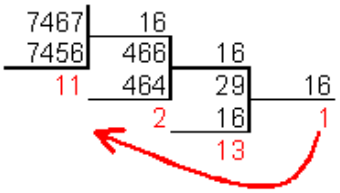
Пример. Число 571_{10} перевести в восьмеричную систему счисления.



$$571_{10} = 1073_8$$

6. Для перевода десятичного числа в шестнадцатеричную систему его необходимо последовательно делить на 16 до тех пор, пока не останется остаток, меньший или равный 15. Число в шестнадцатеричной системе записывается как последовательность цифр последнего результата деления и остатков от деления в обратном порядке.

Пример. Число 7467_2 перевести в шестнадцатеричную систему счисления.



$$7467_{10} = 1D2B_{16}$$

7. Чтобы перевести число из двоичной системы в восьмеричную, его нужно разбить на триады (тройки цифр), начиная с младшего разряда, в случае необходимости дополнив старшую триаду нулями, и каждую триаду заменить соответствующей восьмеричной цифрой (табл. 3).

Пример. Число 1001011_2 перевести в восьмеричную систему счисления.

$$001\ 001\ 011_2 = 113_8$$

8. Чтобы перевести число из двоичной системы в шестнадцатеричную, его нужно разбить на тетрады (четверки цифр), начиная с младшего разряда, в случае необходимости дополнив старшую тетраду нулями, и каждую тетраду заменить соответствующей восьмеричной цифрой (табл. 3).

Пример. Число 1011100011_2 перевести в шестнадцатеричную систему счисления.

$$0010\ 1110\ 0011_2 = 2E3_{16}$$

9. Для перевода восьмеричного числа в двоичное необходимо каждую цифру заменить эквивалентной ей двоичной триадой.

Пример. Число 531_8 перевести в двоичную систему счисления.

$$531_8 = 101011001_2$$

10. Для перевода шестнадцатеричного числа в двоичное необходимо каждую цифру заменить эквивалентной ей двоичной тетрадой.

Пример. Число $EE8_{16}$ перевести в двоичную систему счисления.

$$EE8_{16} = 111011101000_2$$

11. При переходе из восьмеричной системы счисления в шестнадцатеричную и обратно, необходим промежуточный перевод чисел в двоичную систему.

Пример 1. Число FEA_{16} перевести в восьмеричную систему счисления.

$$FEA_{16} = 111111101010_2$$

$$111\ 111\ 101\ 010_2 = 7752_8$$

Пример 2. Число 6653_8 перевести в шестнадцатеричную систему счисления.

$$6653_8 = 110110101011_2$$

$$1101\ 1010\ 1011_2 = DAB_{16}$$

3. Арифметические действия над целыми числами в 2-ой системе счисления:

1.Операция сложения выполняется с использованием таблицы двоичного сложения в одном разряде:

+	0	1
0	0	1
1	1	10_2

Пример.

а) $+1001_2$ б) $+1101_2$ в) $+11111_2$

1010_2 1011_2 1_2

10011_2 11000_2 100000_2

2.Операция вычитания выполняется с использованием таблицы вычитания, в которой 1 обозначается заем в старшем разряде.

Пример.

а) -101110011_2 б) -110101101_2

100011011_2 101011111_2

001011000_2 001001110_2

3.Операция умножения выполняется по обычной схеме, применяемой в десятичной с/с с последовательным умножением множимого на очередную цифру множителя.

×	0	1
0	0	0
1	0	1

Пример.

а) $\times 11001_2$ б) $\times 101_2$

1101_2 11_2

11001 101

11001 101

11001 1111_2

101000101_2

4.Операция деления выполняется по алгоритму, подобному алгоритму выполнения операции деления в 10–ой с/с.

Пример.

101000101_2 | 1101_2 100011000_2 | 1111_2

1101 11001_2 1111 10010_2

1110 10100

1101 1111

001101 1010_2 –остаток

1101

0

3.1. Сложение и вычитание в восьмеричной системе счисления.

При выполнении сложения и вычитания в 8-ой с/с необходимо соблюдать следующие правила:

1) в записи результатов сложения и вычитания могут быть использованы только цифры восьмеричного алфавита;

2) десяток восьмеричной системы счисления равен 8, т.е. переполнение разряда наступает, когда результат сложения больше или равен 8.

В этом случае для записи результата надо вычесть 8, записать остаток, а к старшему разряду прибавить единицу переполнения;

3)если при вычитании приходится занимать единицу в старшем разряде, эта единица переносится в младший разряд в виде восьми единиц.

Пример

$+\ 770_8$	$+\ 750_8$
236_8	236_8
1226_8	512_8

3.2. Сложение и вычитание в шестнадцатеричной системе счисления.

При выполнении этих действий в 16–ой с/с необходимо соблюдать следующие правила:

1)при записи результатов сложения и вычитания надо использовать цифры шестнадцатеричного алфавита: цифры, обозначающие числа от 10 до 15 записываются латинскими буквами, поэтому, если результат является числом из этого промежутка, его надо записывать соответствующей латинской буквой;

2)десяток шестнадцатеричной системы счисления равен 16, т.е. переполнение разряда поступает, если результат сложения больше или равен 16, и в этом случае для записи результата надо вычесть 16, записать остаток, а к старшему разряду прибавить единицу переполнения;

3)если приходится занимать единицу в старшем разряде, эта единица переносится в младший разряд в виде шестнадцати единиц.

Примеры.

$+\ B09_{16}$	$+\ B09_{16}$
TFA_{16}	$7FA_{16}$
$1A03_{16}$	$30F_{16}$

Задание

1. Выполнить перевод чисел

а) из 10–ой с/с в 2–ую систему счисления: 165; 541; 600; 720; 43,15; 234,99.

б) из 2–ой в 10–ую систему счисления: 110101_2; 11011101_2; 110001011_2; $1001001{,}111_2$

в) из 2–ой с/с в 8–ую ,16–ую с/с:
100101110_2; 100000111_2; 111001011_2; 1011001011_2; 110011001011_2; $10101{,}10101_2$; $111{,}011_2$

г) из 10–ой с/с в 8–ую, 16–ую с/с: 69; 73; 113; 203; 351; 641; 478,99; 555,555

д) из 8–ой с/с в 10–ую с/с: 35_8 ; 65_8 ; 215_8 ; 327_8 ; 532_8 ; 751_8; $45{,}454_8$

е) из 16–ой с/с в 10–ую с/с: $D8_{16}$; $1AE_{16}$; $E57_{16}$; $8E5_{16}$; FAD_{16}; $AFF{,}6A7_{16}$

2. Выпишите целые десятичные числа, принадлежащие следующим чловым промежуткам:

$[10101_2; 110000_2]$; $[14_8; 20_8]$; $[18_{16}; 30_{16}]$

3. Выполнить операции:

а) сложение в двоичной системе счисления

$+ 10010011_2$ $+ 1011101_2$ $+ 10110011_2$ $+10111001,1_2$

1011011_2 11101101_2 1010101_2 $10001101,1_2$

б) вычитание в 2–ой системе счисления

$– 100001000_2$ $– 110101110_2$ $– 11101110_2$ $-10111001,1_2$

10110011_2 10111111_2 1011011_2 $10001101,1_2$

в) умножение в 2–ой системе счисления

$\times 100001_2$ $\times 100101_2$ $\times 111101_2$ $\times 11001,01_2$

111111_2 111011_2 111101_2 $11,01_2$

г) деление в 2–ой системе счисления

1) $111010001001_2 / 111101_2$
2) $100011011100_2 / 110110_2$
3) $10000001111_2 / 111111_2$

д) сложение 8–ых чисел

$+ 715_8$ $+ 524_8$ $+ 712_8$ $+ 321_8$ $+ 5731_8$ $+ 6351_8$

73_8 57_8 763_8 765_8 1376_8 737_8

е) вычитание 8–ых чисел

$– 137_8$ $– 436_8$ $– 705_8$ $– 538_8$ $– 7213_8$

72_8 137_8 76_8 57_8 537_8

ж) сложение 16–ых чисел

$+ A13_{16}$ $+ F0B_{16}$ $+ 2EA_{16}$ $+ ABC_{16}$ $+ A2B_{16}$

$16F_{16}$ $1DA_{16}$ FCE_{16} $C7C_{16}$ $7F2_{16}$

з) вычитание 16–ых чисел

$– À17_{16}$ $– DFA_{16}$ $– FO5_{16}$ $– DE5_{16}$ $– D3C1_{16}$

$1FC_{16}$ $1AE_{16}$ AD_{16} AF_{16} $D1F_{16}$

4. Вычислите выражение:

$(1111101_2 + AF_{16}) / 36_8$; $125_8 + 11101_2 \times A2_{16} / 1417_8$

Шкала оценки образовательных достижений

Процент результативности (правильных ответов)	Оценка уровня подготовки	
	балл (отметка)	вербальный аналог
90 ÷ 100	5	отлично
80 ÷ 89	4	хорошо
70 ÷ 79	3	удовлетворительно
менее 70	2	неудовлетворительно

Практическая работа №2. Арифметические операции в прямом, обратном, дополнительном кодах

Целые числа в ЭВМ могут быть представлены в виде:

- прямого кода. Прямой код двоичного числа совпадает по изображению с записью самого числа.
- обратного кода. Обратный код для положительного числа совпадает с прямым кодом. Для отрицательного числа все цифры числа заменяются на противоположные (1 на 0, 0 на 1), а в знаковый разряд заносится единица.
- дополнительного кода. Дополнительный код положительного числа совпадает с прямым кодом. Для отрицательного числа дополнительный код образуется путем получения обратного кода и добавлением к младшему разряду единицы.

Прямой код числа кодирует только знаковую информацию и используется для хранения положительных и отрицательных чисел в ЭВМ. Прямой код двоичного числа совпадает по изображению с записью самого числа, но в знаковом разряде ставится 0, если число положительное и, 1 если число отрицательное.
Обратный и дополнительный коды используются для выполнения всех арифметических операций через операцию сложения.
Следует помнить, что положительные числа в обратном и дополнительном коде совпадают с прямым кодом.
1) Прямой код числа (кодируется только знаковая информация), “+”=0; ”-”=1.
Для прямого кода возможны два представления нуля, машинный положительный ноль, т.е. +0,110=0,110, машинный отрицательный ноль, т.е. -0,111=1,111.

ПРИМЕР ПЕРЕВОДА

$$x_1 = 10101 \; - \; [x_1]_{пр} = 010101$$

$$x_2 = -11101 \; - \; [x_2]_{пр} = 111101$$

$$x_3 = 0{,}101 \; - \; [x_3]_{пр} = 0{,}101$$

$$x_4 = -0{,}111 \; - \; [x_4]_{пр} = 1{,}111$$

2) Обратный код числа, используется для выполнения арифметических операций вычитания, умножения, деления, через сложение. Обратный код положительного числа совпадает с его прямым кодом, обратный код отрицательного числа формируется по правилам: в знаковом разряде записывается “1”; цифровые значения меняются на противоположные.

ПРИМЕР ПЕРЕВОДА

$$x_1 = 10101 \; - \; [x_1]_{обр} = 010101$$

$$x_2 = -11101 \; - \; [x_2]_{обр} = 100010$$

$$x_3 = 0{,}101 \; - \; [x_3]_{обр} = 0{,}101$$

$$x_4 = -0{,}111 \; - \; [x_4]_{обр} = 1{,}000$$

3) *Дополнительный код числа*, имеет такое же назначение, как и обратный код числа. Формируется по следующим правилам: положительные числа в дополнительном коде выглядят также как и в обратном и в прямом коде, т.е. не изменяются. Отрицательные числа кодируются следующим образом: к обратному

коду отрицательного числа (к младшему разряду) добавляется 1, по правилу двоичной арифметики.

ПРИМЕР ПЕРЕВОДА

$x_1 = 10101 \; - \; [x_1]_{доп} = 010101$

$x_2 = -11101 \; - \; [x_2]_{обр} = 100010 + 1 \; - \; [x_2]_{доп} = 100011$

$x_3 = 0{,}101 \; - \; [x_3]_{доп} = 0{,}101$

$x_4 = -0{,}111 \; - \; [x_4]_{обр} = 1{,}000 + 1 \; - \; [x_4]_{доп} = 1{,}001$

Для выявления ошибок при выполнении арифметических операций используются также модифицированные коды: модифицированный прямой; модифицированный обратный; модифицированный дополнительный, для которых под код знака числа отводится два разряда, т.е. “+”=00; ”-”=11. Если в результате выполнения операции в знаковом разряде появляется комбинация 10 или 01 то для машины это признак ошибки, если 00 или 11 то результат верный.

Как определить, положительное или отрицательное число? Знак числа определяет старший бит: 0 - положительное число, 1 - отрицательное число. Например, для числа *1,001* сразу можно определить, что оно отрицательное (меньше нуля).

Текст задания:

1) Записать число в прямом, обратном и дополнительном кодах:
а) 11010; б) -11101; в) -101001; г) -1001110.

2) Перевести X и Y в прямой, обратный и дополнительный коды. Сложить их в обратном и дополнительном кодах. Результат перевести в прямой код. Проверить полученный результат, пользуясь правилами двоичной арифметики.

а) X= -11010; Y= 1001111;	б) X= -11101; Y= -100110;	в) X= 1110100; Y= -101101;
г) X= -10110; Y= -111011;	д) X= 1111011; Y= -1001010;	е) X= -11011; Y= -10101.

3) Сложить X и Y в модифицированном обратном и модифицированном дополнительном восьмиразрядных кодах. В случае появления признака переполнения увеличить число разрядов в кодах и повторить суммирование. Результат перевести в прямой код и проверить, пользуясь правилами двоичной арифметики.

а) X= 10110; Y= 110101;	б) X= 11110; Y= -111001;	в) X= -11010; Y= -100111;
г) X= -11001; Y=-100011;	д) X= -10101; Y= 111010;	е) X= -1101; Y= -111011.

За правильный ответ на вопросы или верное решение задачи выставляется положительная оценка – 1 балл.

За не правильный ответ на вопросы или неверное решение задачи выставляется отрицательная оценка – 0 баллов.

Шкала оценки образовательных достижений

Процент результативности (правильных ответов)	Оценка уровня подготовки	
	балл (отметка)	вербальный аналог
90 ÷ 100	5	отлично
80 ÷ 89	4	хорошо
70 ÷ 79	3	удовлетворительно
менее 70	2	неудовлетворительно

РАЗДЕЛ 3. АРХИТЕКТУРА И ПРИНЦИПЫ РАБОТЫ ОСНОВНЫХ ЛОГИЧЕСКИХ БЛОКОВ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Практическая работа №3. Моделирование арифметических и логических команд

Логические выражения и логические операции

Логическое выражение - это символическая запись, состоящая из логических величин (констант или переменных), объединенных логическими операциями (связками).

В булевой алгебре простым высказываниям ставятся в соответствие *логические переменные*, значение которых равно 1, если высказывание истинно, и 0, если высказывание ложно. Обозначаются логические переменные буквами латинского алфавита.

Существуют разные варианты обозначения истинности и ложности переменных:

Истина	И	True	T	1
Ложь	Л	False	F	0

Связки "НЕ", "И", "ИЛИ" заменяются логическими операциями *инверсия, конъюнкция, дизъюнкция*. Это основные логические операции, при помощи которых можно записать любое логическое выражение.

Логическое отрицание (инверсия).

В обыденной речи мы часто пользуемся словом "НЕ", или словами "НЕВЕРНО, ЧТО", когда хотим что-то отрицать. Пусть, например, кто-то сказал: "Тоска зеленая." (Обозначим это высказывание А). Если Вы не согласны, Вы скажете:" Тоска НЕ зеленая." Или:" Неверно, что тоска зеленая." (Ваше высказывание обозначим **В**). Нетрудно заметить, что значения истинности высказываний **А** и **В** находятся в определенной связи: если **А** истинно, то **В** ложно, и наоборот. Операция, с помощью которой из высказывания **А** получается

высказывание **B**, называется логическим отрицанием и само высказывание **B** называется отрицанием высказывания **A** и обозначается ¬ **A.**

Таким образом, отрицанием ¬ A некоторого высказывания A называется такое высказывание, которое истинно, когда A ложно, и ложно, когда A истинно. Отрицание высказывания **A** обозначим ¬**A**. Определение отрицания может быть записано с помощью так называемой таблицы истинности:

A	¬ A
И	Л
Л	И

В ней указано, какие значения истинности (Истина, Ложь) принимает отрицание ¬ A в зависимости от значений истинности исходного высказывания A.

Логическое умножение (конъюнкция) от латинского conjunctio - союз, связь.

Если два высказывания соединены союзом "И", то полученное сложное высказывание обычно считается истинным тогда и только тогда, когда истинны оба составляющие его высказывания. Если хотя бы одно из составляющих высказываний ложно, то и полученное из них с помощью союза "И" сложное высказывание также считается ложным. Например, возьмем два высказывания: "У кота есть хвост" (**A**), "У зайца есть хвост" (**B**). Сложное высказывание "У кота есть хвост и у зайца есть хвост" истинно, т.к. истинны оба высказывания **A** и **B.** Но если взять другие высказывания: "У кота длинный хвост" (**C**), "У зайца длинный хвост" (**D**), то сложное высказывание "У кота длинный хвост и у зайца длинный хвост" будет ложным, т.к. ложно высказывание (**D**). Таким образом, исходя из обычного смысла союза "И", приходим к определению соответствующей логической операции - конъюнкции.

Таким образом, конъюнкцией двух высказываний A и B называется такое высказывание, которое истинно тогда и только тогда, когда истинны оба высказывания A и B.

Конъюнкцию высказываний **A** и **B** мы обозначим: **A** & **B**. Знак & - амперсент - читается как английское "and" (помните Procter & Gamble или Wash & Go?). Часто встречается обозначение **A** Λ **B**. Иногда, для краткости, пишут просто **AB.**

Определение конъюнкции может быть записано в виде таблицы истинности, в которой для каждого из четырех возможных наборов значений исходных высказываний **A** и **B** задается соответствующее значение конъюнкции **A & B**:

A	B	A&B
и	и	и
и	л	л
л	и	л
л	л	л

Определение конъюнкции двух высказываний естественным образом распространяется на любое конечное число составляющих: конъюнкция A_1 & A_2 & A_3 &...& A_N истинна тогда и только тогда, когда истинны все высказывания A_1, A_2, A_3, ...A_N (а, следовательно, ложна, когда ложно хотя бы одно из этих высказываний).

Логическое сложение (дизъюнкция) от латинского disjunctio - разобщение, различие.

Если два высказывания соединены союзом "ИЛИ", то полученное сложное высказывание обычно считается истинным, когда истинно хотя бы одно из составляющих высказываний. Например, возьмем два высказывания: "Мел черный." (**A**), "Доска черная." (**B**). Высказывание "Мел черный или доска черная" будет истинным, т.к. одно из исходных высказываний (B) истинно.

Таким образом, дизъюнкцией двух высказываний называется такое новое высказывание, которое истинно тогда и только тогда, когда истинно хотя бы одно из этих высказываний.

Дизъюнкцию высказываний **A** и **B** мы обозначим символом **A** V **B** и будем читать: **A** или **B**. Определение дизъюнкции может быть записано в виде таблицы истинности:

A	B	AVB
И	И	И
И	Л	И
Л	И	И
Л	Л	Л

Определение дизъюнкции двух высказываний естественным образом распространяется на любое конечное число составляющих: дизъюнкция A_1 V A_2 V A_3 V...V A_N истинна тогда и только тогда, когда истинно хотя бы одно из высказываний A1, A2, A3, ..., AN (а следовательно, ложна, когда ложны все эти высказывания).

Логическое следование (импликация) от латинского implico - тесно связываю.

В наших рассуждениях, особенно в математических доказательствах, мы часто пользуемся сложными высказываниями, образованными с помощью слов "если..., то...". Здесь высказывание, расположенное после слова "если", называется основанием или посылкой, а высказывание, расположенное после слова "то", называется следствием или заключением.

Рассмотрим пример: из арифметики. Вам должно быть известно, что утверждение "если каждое слагаемое делится на 3, то и сумма делится на 3" истинно, т.е. из высказывания "каждое слагаемое делится на 3" следует высказывание "сумма делится на 3". Посмотрим, какие наборы значений истинности посылки и заключения возможны, когда истинно все утверждение. Возьмем, например, в качестве слагаемых числа 6 и 9. В этом случае истинны и посылка, и заключение, и все утверждение. Если же взять числа 4 и 5, то посылка будет ложной, а заключение истинным. Для чисел 4 и 7 и посылка и заключение

ложны. (Если Вы сомневаетесь в истинности высказывания для последнего случая попробуйте произнести его в сослагательном наклонении: если бы числа 4 и 7 делились бы на 3, то и их сумма делилась бы на 3). Очевидно, что только один случай невозможен: мы не найдем таких двух слагаемых, чтобы каждое из них делилось на 3, а их сумма не делилась на 3, т.е. чтобы посылка была истинной, а заключение ложным. Из истины не может следовать ложь, иначе логика теряет смысл. Высказывание "Если А, то В" с логической точки зрения имеет тот же смысл, что и высказывание "неверно, что А истинно и В ложно". Это означает, что функцию импликации можно заменить комбинацией двух функций (отрицания и конъюнкции). Обычно, когда мы хотим установить ложность высказывания "Если **А**, то **В**", мы стараемся показать, что возможен случай, когда **А** истинно, а **В** ложно (доказательство "от противного"). Обозначим импликацию символом => и запись "**А** => **В**" будем читать: "Из **А** следует **В**".

Таким образом, импликацией А => В называется высказывание, которое ложно тогда и только тогда, когда А истинно и В ложно.

Запишем это определение в виде таблицы истинности:

А	В	А=>В
И	И	И
И	Л	Л
Л	И	И
Л	Л	И

Логическое тождество (эквиваленция).

Интуитивно можно догадаться, что высказывания эквивалентны (равносильными), когда их значения истинности одинаковы. Например, эквивалентны высказывания: "железо тяжелое" и "пух легкий", так же как и высказывания: "железо легкое" и "пух тяжелый". Обозначим эквиваленцию символом <=> и запись "**А** <=> **В**" будем читать "**А** эквивалентно **В**", или "**А** равносильно **В**", или "**А**, если и только если **В**".

Таким образом, эквиваленцией двух высказываний А и В называется такое высказывание, которое истинно тогда и только тогда, когда оба эти высказывания А и В истинны или оба ложны.

Отметим, что высказывание типа "**А**, если и только если **В**" можно заменить высказыванием "Если **А**, то **В** и, если **В**, то **А**" (обдумайте это на досуге и обратите внимание на символ <=>). Следовательно, функцию эквиваленции можно заменить комбинацией функций импликации и конъюнкции. Запишем таблицу истинности для эквиваленции:

А	В	А<=>В
И	И	И
И	Л	Л

Л	И	Л
Л	Л	И

Приведем примеры записи сложных высказываний с помощью обозначения логических связок:

"Быть иль не быть - вот в чем вопрос." (В. Шекспир) A V ¬ A <=> B

"Если хочешь быть красивым, поступи в гусары." (К. Прутков) A => B

Построение таблиц истинности для логических функций

Логическая функция - это функция, в которой переменные принимают только два значения: логическая единица или логический ноль. Истинность или ложность сложных суждений представляет собой функцию истинности или ложности простых. Эту функцию называют булевой функцией суждений f (a, b).

Любая логическая функция может быть задана с помощью таблицы истинности, в левой части которой записывается набор аргументов, а в правой части - соответствующие значения логической функции.

При построении таблицы истинности необходимо учитывать порядок выполнения логических операций. Операции в логическом выражении выполняются слева направо с учетом скобок в следующем порядке:

1. инверсия;
2. конъюнкция;
3. дизъюнкция;
4. импликация и эквивалентность.

Для изменения указанного порядка выполнения логических операций используются круглые скобки.

Предлагается следующий *алгоритм построения таблицы истинности*.

1. Определить *количество наборов входных переменных* - всевозможных сочетаний значений переменных, входящих в выражения, по формуле: $Q=2^n$, где n - количество входных переменных. Оно определяет количество строк таблицы.
2. Внести в таблицу все наборы входных переменных.
3. Определить количество логических операций и последовательность их выполнения.
4. Заполнить столбцы результатами выполнения логических операций в обозначенной последовательности.

$$\text{Количество столбцов} = \left\{ \begin{matrix} \text{количество переменных} + \\ + \text{количество операций.} \end{matrix} \right\}$$

Чтобы не повторить или не пропустить ни одного возможного сочетания значений входных переменных, следует пользоваться одним из предлагаемых ниже способов заполнения таблицы.

Способ 1. Каждый набор значений исходных переменных есть код числа в двоичной системе счисления, причем количество разрядов числа равно количеству входных переменных. Первый набор - число 0. Прибавляя к текущему

числу каждый раз по 1, получаем очередной набор. Последний набор - максимальное значение двоичного числа для данной длины кода.

Например, для функции от трех переменных последовательность наборов состоит из чисел:

000
001
010
011
100
101
110
111

Способ 2. Для функции от трех переменных последовательность данных можно получить следующим путем:

а) разделить колонку значений первой переменной пополам и заполнить верхнюю половину нулями, нижнюю половину единицами;

б) в следующей колонке для второй переменной половинку снова разделить пополам и заполнить группами нулей и единиц; аналогично заполнить вторую половинку;

в) так делать до тех пор, пока группы нулей и единиц не будут состоять из одного символа.

Способ 3. Воспользоваться известной таблицей истинности для двух аргументов. Добавляя третий аргумент, сначала записать первые 4 строки таблицы, сочетая их со значением третьего аргумента, равным 0, а затем еще раз записать эти же 4 строки, но теперь уже со значением третьего аргумента, равным 1. В результате в таблице для трех аргументов окажется 8 строк:

000
010
100
110
001
011
101
111

Например, построим таблицу истинности для логической функции:

$F(A,B,C) = \bar{A} \& (B \vee C)$

Количество входных переменных в заданном выражении равно трем *(A,B,C)*. Значит, количество входных наборов $Q=2^3=8$.

Столбцы таблицы истинности соответствуют значениям исходных выражений *A,B,C*, промежуточных результатов $\bar{A}$ и (*B* V *C*), а также искомого окончательного значения сложного арифметического выражения $\bar{A} \& (B \vee C)$:

A	B	C	$\bar{A}$	B V C	$\bar{A} \& (B \vee C)$
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	0	1	0

Логические функции и их преобразования. Законы логики

Для операций конъюнкции, дизъюнкции и инверсии определены законы булевой алгебры, позволяющие производить *тождественные (равносильные) преобразования логических выражений*.

Законы логики

1. ¬¬ A <=> A закон двойного отрицания;
2. A&B <=> B&A коммутативность конъюнкции;
3. AVB <=> BVA коммутативность дизъюнкции;
4. A&(B&C) <=> (A&B)&C ассоциативность конъюнкции;
5. AV(BVC) <=> (AVB)VC ассоциативность дизъюнкции;
6. A&(BVC) <=> (A&B)V(A&C) дистрибутивность конъюнкции относительно дизъюнкции;
7. AV(B&C) <=> (AVB)&(AVC) дистрибутивность дизъюнкции относительно конъюнкции;
8. A&A <=> A
9. AVA <=> A
10. AV¬A <=> И закон исключенного третьего;
11. A&¬A <=> Л закон непротиворечия;
12. A&И <=> A
13. AVИ <=> И
14. A&Л <=> Л

15. AVЛ <=> A
16. ¬(A&B) <=> ¬ A V ¬ B законы де Моргана;
17. ¬(AVB) <=> ¬ A & ¬ B
18. A => B <=> ¬ A V B замена импликации.

Основываясь на законах, можно выполнять упрощение сложных логических выражений. Такой процесс замены сложной логической функции более простой, но равносильной ей, называется минимизацией функции.

Пример 1. Упростить выражения $\overline{A\bar{B}}+\bar{B}$, $\overline{\bar{B}C+C}$, $\overline{\bar{A}C}+B\bar{C}$ так, чтобы в полученных формулах не содержалось отрицания сложных высказываний.

Решение

$$X=\overline{A\bar{B}}+\bar{B}=\bar{A}+\bar{\bar{B}}+\bar{B}=\bar{A}+\bar{B};$$
$$Y=\overline{\bar{B}C+C}=\left(B+\bar{C}\right)\bar{C}=\bar{C};$$
$$\overline{\bar{A}C}+B\bar{C}=A+\bar{C}+B\bar{C}=A+\bar{C}.$$

Пример 2. Минимизировать функцию

$$F(x,y,z)=(x\vee y)z\vee xyz(\bar{z}\vee xy)\vee\bar{z}xy\vee\bar{x}y\bar{z}.$$

Решение

$$F(x,y,z)=(x\vee y)z\vee xyz(\bar{z}\vee xy)\vee\bar{z}xy\vee\bar{x}y\bar{z}=$$
$$=xz\vee yz\vee 0\vee xyz\vee y\bar{z}(x\vee\bar{x})=xz\vee y.$$

При упрощении выражения использовались формулы поглощения и склеивания.

Пример 3. Найти отрицание следующего высказывания: "Если урок будет интересным, то никто из учеников (Миша, Вика, Света) не будет смотреть в окно".

Решение

Обозначим высказывания:

Y - "Урок интересный";

M - "Миша смотрит в окно";

B - "Вика смотрит в окно";

C - "Света смотрит в окно".

$$\overline{Y\Rightarrow\bar{\bar{M}}\,\&\,\bar{\bar{B}}\,\&\,\bar{\bar{C}}}=\overline{\bar{\bar{Y}}\vee\overline{\bar{M}\,\&\,\bar{\bar{B}}\,\&\,\bar{\bar{C}}}}=$$
$$=\bar{\bar{Y}}\,\&\,\overline{\bar{\bar{M}}\,\&\,\bar{\bar{B}}\,\&\,\bar{\bar{C}}}=Y\,\&\,(M\vee B\vee C)$$

При упрощении выражения использовались формула замены операций и закон де Моргана.

Пример 4. Определить участника преступления, исходя из двух посылок:

1) "Если Иванов не участвовал или Петров участвовал, то Сидоров участвовал";

2) "Если Иванов не участвовал, то Сидоров не участвовал".

Решение

Составим выражения:

I - "Иванов участвовал в преступлении";

P - "Петров участвовал в преступлении";

S - "Сидоров участвовал в преступлении".

Запишем посылки в виде формул:

$\bar{I} \vee P \Rightarrow S$ **И** $\bar{I} \Rightarrow \bar{S}$.

Тогда

$$F(I,P,S) = (\bar{I} \vee P \Rightarrow S) \& (\bar{I} \Rightarrow \bar{S}) =$$

$$= \left(\overline{(\bar{I} \vee P)} \vee S\right) \& (I \vee \bar{S}) = (I\bar{P} \vee S) \& (I \vee \bar{S}) =$$

$$= I\bar{P} \vee IS \vee I\bar{P}\bar{S} \vee 0 = I\bar{P} \vee IS = I(\bar{P} \vee S).$$

Проверим результат, используя таблицу истинности:

I	P	S	$\bar{I} \vee P$	$\bar{I} \vee P \Rightarrow S$	$\bar{I} \Rightarrow \bar{S}$	$(\bar{I} \vee P \Rightarrow S)(\bar{I} \Rightarrow \bar{S})$
0	0	0	1	0	1	0
0	0	1	1	1	0	0
0	1	0	1	0	1	0
0	1	1	1	1	0	0
1	**0**	**0**	0	1	1	**1**
1	**0**	**1**	0	1	1	**1**
1	1	0	1	0	1	0
1	**1**	**1**	1	1	1	**1**

Ответ: Иванов участвовал в преступлении.

Построение логической функции по ее таблице истинности

Мы научились составлять таблицу истинности для логической функции. Попробуем решить обратную задачу. Пусть дана таблица истинности для некоторой логической функции Z(X,Y):

X	Y	Z
0	0	1
0	1	0
1	0	1
1	1	0

Рассмотрим строки, где значение истинности функции Z истинно (Z=1). Функцию для этой таблицы истинности можно составить следующим образом: Z(X,Y) = (¬

X& ¬Y)V(X& ¬Y).

Каждой строке, где функция истинна (равна 1), соответствует скобка, представляющая собой конъюнкцию аргументов, причем если значение аргумента О, то мы берем его с отрицанием. Все скобки соединены между собой операцией дизъюнкции. Полученную формулу можно упростить, применив законы логики:

Z(X,Y) <=> ((¬X& ¬Y) VX)&((¬X&Y)V ¬Y) <=> (XV(¬X& ¬Y)) &(¬YV(¬X&¬Y)) <=> ((XV¬X)&(XV ¬Y))&((Y¬V ¬X)&(¬YV ¬Y)) <=> (1&(XV ¬Y))&((¬YV ¬X)& ¬Y)<=> (XV ¬Y)&((¬YV ¬X)& ¬Y).

Проверьте полученную формулу: составьте таблицу истинности для функции Z(X,Y).

Запишите правила конструирования логической функции по ее таблице истинности:

1. Выделить в таблице истинности те строки, в которых значение функции равно 1.

2. Выписать искомую формулу в виде дизъюнкции нескольких логических элементов. Число этих элементов равно числу выделенных строк.

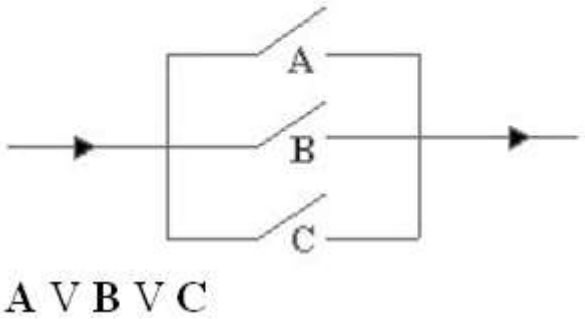
3. Каждый логический элемент в этой дизъюнкции записать в виде конъюнкции аргументов функции.

4. Если значение какого-либо аргумента функции в соответствующей строке таблице равно 0, то этот аргумент мы берем с отрицанием.

Построение логических схем

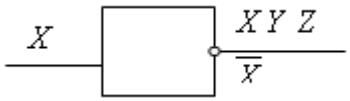
Знания из области математической логики можно использовать для конструирования электронных устройств. Нам известно, что 0 и 1 в логике не просто цифры, а обозначение состояний какого-то предмета нашего мира, условно называемых "ложь" и "истина". Таким предметом, имеющим два фиксированных состояния, может быть электрический ток. Устройства, фиксирующие два устойчивых состояния, называются бистабильными (например, выключатель, реле). Если вы помните, первые вычислительные машины были релейными. Позднее были созданы новые устройства управления электричеством - электронные схемы, состоящие из набора полупроводниковых элементов. Такие электронные схемы, которые преобразовывают сигналы только двух фиксированных напряжений электрического тока (бистабильные), стали называть *логическими элементами*.

На элементарном уровне конъюнкцию можно представить себе в виде последовательно соединенных выключателей, а дизъюнкцию - в виде параллельно соединенных выключателей:

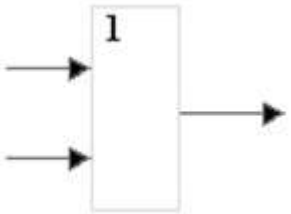


A V B V C

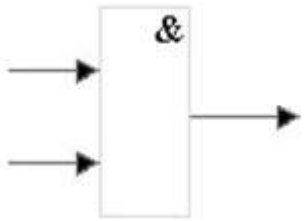
Логические элементы имеют один или несколько входов и один выход, через которые проходят электрические сигналы, обозначаемые условно 0, если "отсутствует" электрический сигнал, и 1, если "имеется" электрический сигнал. Простейшим логическим элементом является *инвертор*, выполняющий функцию отрицания. Если на вход поступает сигнал, соответствующий 1, то на выходе будет 0. И наоборот. У этого элемента один вход и один выход. На функциональных схемах он обозначается:



Логический элемент, выполняющий логическое сложение, называется *дизъюнктор*. Он имеет, как минимум, два входа. На функциональных схемах он обозначается:

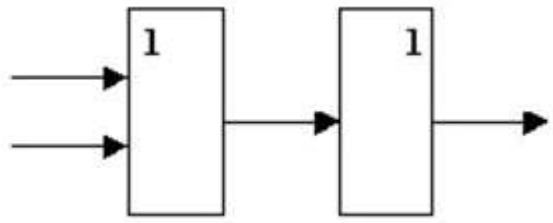


Логический элемент, выполняющий логическое умножение, называется конъюнктор. Он имеет, как минимум, два входа. На функциональных схемах он обозначается:



Специальных логических элементов для импликации и эквивалентности нет, т.к. A => B можно заменить на ¬A V B ; A <=> B можно заменить на (A & B)V(¬A & ¬B).

Другие логические элементы построены из этих трех простейших и выполняют более сложные логические преобразования информации. Сигнал, выработанный одним логическим элементом, можно подавать на вход другого элемента, это дает возможность образовывать цепочки из отдельных логических элементов. Например:

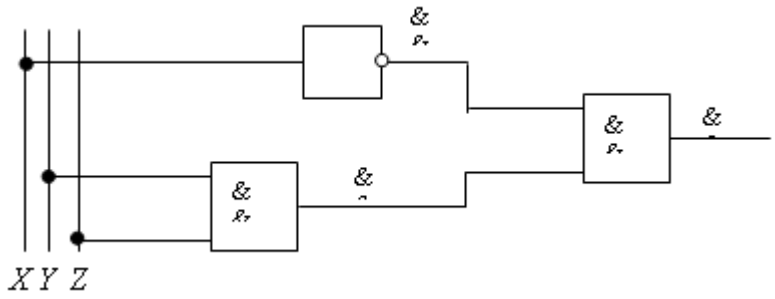


Эта схема соответствует сложной логической функции F(A,B)= ¬ (A V B).

Попробуйте проследить изменения электрического сигнала в этой схеме. Например, какое значение электрического сигнала (0 или 1) будет на выходе, если на входе: A=1 и B=0.

Такие цепи из логических элементов называются *логическими устройствами*. Логические устройства же, соединяясь, в свою очередь образуют *функциональные схемы* (их еще называют структурными или *логическими схемами*). По заданной функциональной схеме можно определить логическую формулу, по которой эта схема работает, и наоборот.

Пример 1. Логическая схема для функции $F(X,Y,Z)=\overline{X}\,\&\,(Y \vee Z)$ будет выглядеть следующим образом:



Правила составления электронных логических схем по заданным таблицам истинности остаются такими же, как для контактных схем.

Пример 2. Составить логическую схему для тайного голосования трех персон A, B, C, условия которого определяются следующей таблицей истинности:

A	0	0	0	0	1	1	1	1
B	0	0	1	1	0	0	1	1
C	0	1	0	1	0	1	0	1

F	0	0	0	1	0	1	1	1

Решение

По таблице построим СДНФ логической функции и упростим ее:

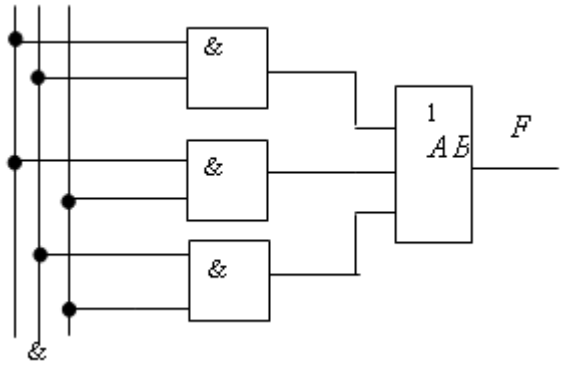
$$F = \bar{A}BC \vee A\bar{B}C \vee AB\bar{C} \vee ABC = BC(\bar{A} \vee A) \vee A(\bar{B}C \vee B\bar{C}) =$$
$$= BC \vee A\bar{B}C \vee AB\bar{C} = C(B \vee A\bar{B}) \vee AB\bar{C} = C(B \vee A) \vee AB\bar{C} =$$
$$= BC \vee AC \vee AB\bar{C} = BC \vee AC \vee AB.$$

Правильность полученной формулы можно проверить, составив для нее таблицу истинности:

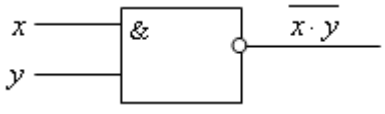
A	B	C	AB	AC	BC	F
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	0	0	0
1	0	1	0	1	0	1
1	1	0	1	0	0	1
1	1	1	1	1	1	1

Значение полученной функции совпадает с исходным, что можно заметить, сравнивая таблицы.

Логическая схема полученной функции имеет вид:



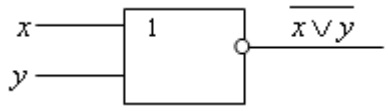
Рассмотрим еще два логических элемента, которые играют роль базовых при создании более сложных элементов и схем.



Логический элемент И-НЕ состоит из конъюнктора и инвертора:

Выходная функция выражается формулой $F(x,y)=\overline{xy}$.

Логический элемент ИЛИ-НЕ состоит из дизъюнктора и инвертора:



Выходная функция выражается формулой $F(x,y)=\overline{x \vee y}$.

Текст задания:

1. $(X_1 \vee \neg X_2) \wedge (X_3 \rightarrow X_4)$
2. $(X_1 \rightarrow X_3) \wedge X_2 \vee X_4$
3. $X_1 \wedge X_2 \vee (X_3 \rightarrow (X_1 \vee X_4))$
4. В состав жюри входят три человека. Решение принимается, если за него голосует председатель жюри, поддержанный хотя бы одним из членов жюри. В противном случае решение не принимается. Постройте логическую функцию, формализующую процесс принятия решения.
5. X выигрывает у Y, если при четырех бросаниях монеты трижды выпадает «орёл». Задайте логическую функцию, описывающую выигрыш X.
6. Слова в предложении нумеруются, начиная с единицы. Предложение считается правильно построенным, если выполняются следующие правила:
    1. Если четное в нумерации слово заканчивается на гласную, то следующее слово, если оно существует, должно начинаться с гласной.
    2. Если нечетное в нумерации слово заканчивается согласной, то следующее слово, если оно существует, должно начинаться с согласной и заканчиваться гласной.

    Какие из следующих предложений правильно построены:

    3. Мама мыла Машу мылом.
    4. Лидер всегда является образцом.
    5. Правда хорошо, а счастье лучше.

7. Сколько решений имеет уравнение:

   $(a \wedge \neg b) \vee (\neg a \wedge b) \rightarrow (c \wedge d) = 1$

8. Перечислите все решения уравнения:

   $(a \rightarrow b) \rightarrow c = 0$

9. Сколько решений имеет следующая система уравнений:

   $X_0 \rightarrow X_1 \wedge X_1 \rightarrow X_2 = 1$

   $X_2 \rightarrow X_3 \wedge X_3 \rightarrow X_4 = 1$

   $X_5 \rightarrow X_6 \wedge X_6 \rightarrow X_7 = 1$

   $X_7 \rightarrow X_8 \wedge X_8 \rightarrow X_9 = 1$

   $X_0 \rightarrow X_5 = 1$

10. Сколько решений имеет уравнение:

    $((((X_0 \rightarrow X_1) \rightarrow X_2) \rightarrow X_3) \rightarrow X_4) \rightarrow X_5 = 1$

За правильный ответ на вопросы или верное решение задачи выставляется положительная оценка – 1 балл.

За не правильный ответ на вопросы или неверное решение задачи выставляется отрицательная оценка – 0 баллов.

Шкала оценки образовательных достижений

Процент результативности (правильных ответов)	Оценка уровня подготовки	
	балл (отметка)	вербальный аналог
90 ÷ 100	5	отлично
80 ÷ 89	4	хорошо
70 ÷ 79	3	удовлетворительно
менее 70	2	неудовлетворительно

Практическая работа №4. Организация работы памяти компьютера

Всю память ЭВМ можно разделить на:

1. ОЗУ (оперативное запоминающее устройство)
2. ПЗУ (постоянное запоминающее устройство)
3. РОН (регистры общего назначения) внутренняя память процессора – его регистры.
4. CMOS (Complement Metal Oxide Semiconductor – комплементарные пары метал-оксид-полупроводник указывает на технологию изготовления данной памяти) – память системных установок(конфигурации).

5. ВЗУ (внешнее запоминающее устройство)
6. Видеопамять – электронная память, размещенная на видеокарте, используется в качестве буфера для хранения кадров динамического изображения.

1,2,3,6 – электронная память, 5 – электромеханическая память.

Характеристики оперативной памяти

Внутренняя память ПК обладает двумя основными свойствами: ***дискретностью и адресуемостью.***

Дискретность – память состоит из битов (бит — элемент памяти, частица информации, хранит двоичный код 0 или 1. Слово бит произошло от англ. «binary digit» — двоичная цифра).

Бит – наименьшая частица памяти компьютера.

Следовательно, у слова «бит» есть два смысла: это единица измерения количества информации и частица памяти компьютера. Оба эти понятия связаны между собой следующим образом:
В одном бите памяти хранится один бит информации.

Память – это упорядоченная последовательность двоичных разрядов(бит). Эта последовательность делится на группы по 8 разрядов. Каждая такая группа образует байт памяти.

Следовательно «бит» и «байт» обозначают не только названия единиц измерения количества информации, но и структурные единицы памяти ЭВМ.
1Кб = 210 байт =1024б
1Мб = 210 Кбайт =1024Кб
1Гб = 1024Мб

Ячейка памяти – группа последовательных байтов внутренней памяти, вмещающая в себе информацию, доступную для обработки отдельной командой процессора.
Содержимое ячейки памяти называется машинным словом. Байты внутренней памяти пронумерованы. Нумерация начинается с 0. Порядковый № байта называется адресом байта. **Принцип адресуемости памяти** заключается в том, что любая информация заносится в память и извлекается из нее по адресам, т.е. чтобы взять информацию из ячейки памяти или поместить ее туда, необходимо указать адрес этой ячейки. Адрес ячейки память равен адресу младшего байта, входящим в ячейку. Адресация памяти начинается с 0. Адреса ячеек кратны количеству байтов в машинном слове.

Структура оперативной памяти

Оперативная память(ОП) (ОЗУ)

Из ОП ЦП берет исходные данные для обработки, в нее записываются полученные результаты. Название «оперативная» память получила потому что работает быстро. Является энергозависимой, данные и программы сохраняются в ней только до тех пор, пока ПК включен, при выключении ПК содержимое ОП стирается. ОЗУ предназначена для хранения текущей, быстроменяющейся информации и допускает изменение своего содержимого в ходе выполнения процессором вычислений.
Используется два основных типа оперативной памяти: ***статическая память (SRAM-Static RAM — КЭШ) и динамическая память (DRAM-Dynamic RAM — ОЗУ).***
Эти две разновидности памяти различаются ***быстродействием и удельной плотностью (емкостью) хранимой информации.***

Быстродействие памяти характеризуется двумя параметрами: временем доступа (access time) и длительностью цикла памяти (cycle time).
Эти величины, как правило, измеряются в наносекундах. Чем больше эти величины, тем больше быстродействие памяти.
Время доступа представляет собой промежуток времени между формированием запроса на чтение информации из памяти и моментом поступления из памяти запрошенного машинного слова (операнда).
Длительность цикла определяется минимальным допустимым временим между двумя последовательными обращениями к памяти.

В **статической памяти** элементы построены на триггерах — схемах с двумя устойчивыми состояниями. Для построения одного триггера требуется 4-6 транзисторов. После записи информации в статический элемент памяти он может хранить информацию сколь угодно долго (пока подается электрическое питание). Статическая память имеет высокое быстродействие и низкую плотность размещения хранящихся данных. Этот вид памяти дорог и энергоемок,

следовательно, может происходить перегрев, что снижает надежность система, поэтому вся ОП не может быть построена по статическому принципу.

В **динамической памяти** элементы памяти построены на основе полупроводниковых конденсаторов, занимающих гораздо более меньшую площадь, чем триггеры в статической памяти.
Для построения динамического элемента памяти требуется 1-2 транзистора. Каждый бит ОП представляется в виде наличия или отсутствия заряда на конденсаторе, образованном в структуре полупроводникового кристалла. Ячейки динамической памяти очень компактны, но со временем конденсатор испытывает утечку заряда, поэтому периодически (приблизительно 1000 раз в сек.) выполняется автоматическое восстановление информации в каждой ячейке. Это снижает скорость работы динамической памяти и является основным ее недостатком.

ОП часто обозначают **RAM (Random Access memory)** – память с произвольным доступом (тип доступа к памяти при котором ячейки памяти пронумерованы, т.е. адресуемы и, следовательно, обращение к ним может производиться в произвольном порядке).

Термин «произвольный доступ» означает, что можно считать (записать) информацию в любой момент времени из любой ячейки.

Заметим, что существует и другая организация памяти, при которой прежде чем считать нужную информацию нужно «вытолкнуть» ранее поступившие операнды.

От объема ОП, установленным на ПК напрямую зависит с каким ПО Вы сможете на нем работать. При недостатке ОП программы не запускаются, выдается сообщение: “Out of memory”, либо работают крайне медленно.

Чем больше ОП в ПК, тем лучше. При необходимости объем ОП можно нарастить (ограничивается параметрами ОП, поддерживаемой конкретной материнской платы, внимательно см.спецификацию к системной плате).

Распределение памяти в ПК (Разделы ОЗУ)

RAM устроена довольно сложно, она иерархична (многоэтажна). ОП разделяют на несколько типов. Деление это обусловлено историческими причинами. Первые компьютеры были выполнены так, что они могли работать максимально с 640Кб памяти. Выделяют 4 вида памяти:

- Стандартная (conventional memory area)
- Верхняя (upper memory blocks(area))
- Дополнительная (expanded memory specification)
- Расширенная (extended memory specification)

Стандартная (conventional memory area) – базовая, первые 640 Кб, также его часто называют lower. В мл. адреса этой памяти загружается ОС и драйверы устройств. Оставшуюся свободную часть памяти занимают пользовательские программы. Резидентные программы так же остаются в этой памяти.

Верхняя (upper memory area) – 640Кб — 1Мб используется для хранения служебной информации: памяти видеоадаптера,BIOS. Спец. драйвер Himem.sys позволяют загружать в свободные участки этой области резидентные программы и драйвера устройств.

High memory – первые 64 Кб после 1Мб. ОС MS DOS позволяет загрузить часть резидентной DOS в эту область, освобождая при этом существенную часть базовой памяти для работы прикладных программ. Особенно это полезно для программ, использующих всю ОП. Используя спец. утилиты (для DOS emm386.exe) в верхние разделы памяти можно загружать также и резидентные программы (команды LH для autoexec.bat и DEVICEHIGT для config.sys).

Вся память свыше 1 Мб может быть рассмотрена как **дополнительная(expanded)** или как **расширенная (extended**). В ОС менеджер памяти позволяет использовать память и как расширенную и как дополнительную, автоматически обеспечивая тот тип взаимодействия с данными, который нужен прикладным программам. Т.е. пользователю новых современных ПК (от Pentium) нет необходимости распределять память «в ручную», менеджер выделить память таким образом, как это требует прикладная программа.

Дополнительная(expanded) память – постраничная, т.е. ОП разбивается на страницы, каждой странице ставится в соответствие определенный адрес в основной памяти. При обращении к такому адресу EMM(expanded memory manager) драйвер расширенной памяти(менеджер памяти) позволяет компьютеру считать информацию с соответствующей страницы памяти.

Расширенная (extended) память построчной организации (Smartdrv — драйвер расширенной памяти) используется для создания временного логического диска (виртуального диска), как буфер обмена с жестким диском.

Распределение ОП в ПК с ОС MS-DOS

1Мб+ 64Кб	High	High Расширенная или дополнительная память
		Резидентные программы и драйверы устройств
		Часть ОС
1Мб	Upper	Верхняя память ПЗУ BIOS
		Видеопамять (текстовый буфер)
		Видеопамять (графический буфер)
640Кб		Свободная часть (command.com) транзитная часть

	Convertional Memory Area (base)Стандартная (базовая память)	Свободная часть для программ пользователя
		Command.com (резидентная часть)
		Программы DOS, драйверы
		Файлы io.sys msdos.sys
		Данные для DOS и BIOS и другая служебная информация

Микросхемы ОП (модули ОП)

Производительность ПК зависит от типа и размера ОП, а это в свою очередь зависит от набора интегральных схем на материнской плате.

Внешний вид микросхем ОП: пластиковая полоска, на ней расположены кремневые «черепашки» – чипы-микросхемы (то есть используется полупроводниковая технология) и имеются «ножевые» контактные разъемы.

Устройства памяти характеризуются следующими основными показателями:

1. временем доступа (быстродействием). Время доступа – промежуток времени, за который может быть записано (прочитано) содержимое ячейки памяти.
2. емкостью (определяет количество ячеек (битов) в устройстве памяти).
3. стоимостью.
4. потребляемой мощностью (электропотреблением).

Существует 2 модуля памяти, отличающиеся формой, внутренней архитектурой, скоростью работы: ***SIMM и DIMM.***
I. SIMM (SINGLE IN-LINE MEMORY MODULES) (SRAM) бывают двух типов (отличающихся количеством контактов).

1. **30-контактные модули SIMM.** Бывают 1 и 4 Мб. Практически сегодня исчезли из продажи для компьютеров 386, 286-процессором. Сегодня им нашлось интересное применение – в качестве ОП, устанавливаемой в некоторые звуковые платы, например, Greafive Sound Blaster 32 (AWE-32) Gravis UltraSound PnP. Однако новая карта AWE-64 уже содержит свои модули ОП, эта память не нужна.

2. **72-контактные SIMM** (на 1, 4, 8, 16, 32, 64 Мб, редко 128 Мб). Внешний вид неизменный, а вот тип устанавливаемой на них памяти меняется (тип памяти указывается на микросхеме).

а) самый старый (редко сейчас встречающийся) – FPM DRAM (или просто DRAM – Dynamic Random Access Memory – динамическая ОП). Работала на 486 и первых Pentium.

b) модифицированный тип EDO DRAM (или EDO – Extended data output).

Микросхемы SIMM выпускаются одинарной и двойной плотности, с контролем четности и без (использование контроля четности позволяет парировать одиночную ошибку памяти). Модули отличаются и по скорости доступа 60 и 70 наносекунд, чем скорость меньше, тем быстрее доступ. 60 наносекунд быстрее 70 наносекунд. Модули SIMM в материнской плате Pentium и Pentium MMX устанавливаются только попарно, образуя так называемый банк.

Пример необходимо 32 Мб => 2 модуля SIMM по 16 Мб. необходимо 64 Мб => 4 модуля SIMM по 16 Мб или 2 модуля SIMM по 32Мб.

В рамках одного банка можно использовать только одинаковые по емкости и скорости доступа модули SIMM. Если на вашей материнской плате 4 слота для модулей памяти SIMM, то можно сформировать два банка различной емкости.

II. DIMM (SDRAM DUAL IN-LINE MEMORY MODULES).

Появился впервые у MMX- компьютеров, стал основой для PII., поэтому у PII редко бывают SIMM-разъемы. DIMM не обязательно должно быть четное число. Модули DIMM бывают емкостью 16, 32, 64, 128, 256, 512 Мб

Виды DIMM.

EDO SD RAM (Synchronous DRAM) – синхронизируемая динамическая ОП)

SD RAM (SINGLE DATA RATE RANDOM ACCESS MEMORY).ЗУПВ с одинарной скоростью передачи данных, которая в зависимости от тактовой частоты называется памятью PC100 и PC133. Микросхема на 168 контактов, является сегодня самой «медленной» из семейства DIMM-модулей памяти, Время доступа = 10-20 наносекунд. Верхний предел ее тактовой частоты 133 МГц. И все же этот тип ОП вполне подходит для большинства офисных и домашних ПК. Пропускная способность 1Гб/с.
Rambus (RD RAM)Двухканальная ОП (микросхема фирмы Intel). Direct Rambus – это новая шина памяти, в которой управление адресацией отделено от работы с данными. Система состоит из контроллера Direct Rambus, подсоединенного к одному или нескольким модулям Direct Rambus DRAM, которые называются RIMM, в отличии от обычных микросхем памяти, соединяемых параллельно, RIMM соединяются последовательно. Канал Direct Rambus включает двунаправленную шину данных и шину адреса, т.е. адреса памяти передаются одновременно с данными. Каждая микросхема RDRAM может содержать до 32 независимых банков, SD RAM – от 2 до 8. Свободно работает на высоких тактовых частотах.
DDR SDRAM (Double Data Rate) – двойная скорость передачи данных – это по

сути модификации обычной SDRAM и отличается от нее тем, что в ней запись и чтение данных происходят и по переднему и по заднему фронту тактового импульса. Поэтому за один такт по шине передается вдвое больше данных, и ее эффективная частота оказывается вдвое больше физической.

Модуль памяти Kingston DDR PC3200

Модуль памяти Kingmax DDR2-667

Пропускная способность

Важнейшей характеристикой модулей оперативной памяти является **пропускная способность.**

Пропускная способность равна произведению разрядности шины данных и частоты операций записи или считывания информации из ячеек памяти:

Пропускная способность = Разрядность шины данных × Частота

Разрядность шины данных = 64 бита.

Максимально возможная в настоящее время (2006 год) частота шины данных совпадает с частотой системной шины и равна 1064 МГц.

Пропускная способность модулей памяти = 64 бита × 1064 МГц = 68 096 Мбит/с = = **8 512 Мбайт/с** ≈ 8 Гбайт/с.

Модули памяти маркируются своей пропускной способностью, выраженной в Мбайт/с: PC3200, PC4200, PC8500 и др.

Физическая и виртуальная память

Объем используемой программами памяти можно увеличить путем добавления к физической памяти (модулям оперативной памяти) **виртуальной памяти.** Виртуальная память выделяется в форме **области жесткого диска**. В ОС Windows это **файл подкачки.** Размер файла подкачки и его размещение в иерархической файловой системе можно изменить.

Быстродействие жесткого диска и, соответственно, виртуальной памяти существенно меньше быстродействия оперативной памяти.

Замедление быстродействия виртуальной памяти может происходить в результате **фрагментации данных** в файле. Для того чтобы этого не происходило, рекомендуется произвести **дефрагментацию диска** и **установить для файла подкачки постоянный размер.**

ПЗУ (постоянное запоминающее устройство)

В ПЗУ информация остаётся неизменной. Запись в ПЗУ обычно осуществляется электрическим или механическим способом, в процессе изготовления материнской карты. Эти данные, как правило, не могут быть изменены, выполняемые не ПК программы могут их только считывать В ПЗУ хранится информация, присутствие которой постоянно необходимо в компьютере.

Часто ее называют ROM (Read Only Memory) – память только для чтения. В постоянной памяти хранятся программы для проверки оборудования компьютера, инициирования загрузки ОС и выполнение базовых функций по обслуживанию устройств ПК. Часто содержимое постоянной памяти называют BIOS(Basic Input Output System) – базовая система ввода/вывода. BIOS – это система контроля и управления устройствами, подключёнными к ПК (жёсткий диск, ОП, часы, календарь). Это часть программного обеспечения ПК, поддерживающая управление адаптерами внешних устройств, экранные операции, тестирование, начальную загрузку и установку OS. BIOS находится на материнской плате (отдельная микросхема с автономным питанием от батарейки в ПК).

На сегодняшних ПК BIOS можно перезаписывать.BIOS сегодня может сам определять новые устройства, подключённые к ПК (стандарт PnP — Plug-And-Play) включи и работай. Управление устройствами осуществляется через механизм прерываний.

Прерывания могут быть:

- аппаратные (инициируются аппаратными средствами),
- логические (инициируются микропроцессором – нестандартные ситуации в работе микропроцессора),
- программные (инициируются каким-либо программным обеспечением).

При включении ПК автоматически загружается и выполняется специальная программа POST(Power-On Self-Test) из состава BIOS.

Эта программа производит самопроверку и тестирование при загрузке:

- проверка переключателей и CMOS-памяти на системной (материнской) плате (определение оборудования, которое подключено к ПК),

- тестирование ОЗУ,
- выполнение действий по загрузке ОС (загрузка в ОЗУ и запуск Блока Начальной Загрузки ОС),
- выполняет другие специфические действия по подготовке ПК и дополнительного оборудования к работе.

ПРАКТИЧЕСКАЯ ЧАСТЬ

Для получения информации о текущем распределении памяти используются программы для просмотра состояния памяти. Некоторые из этих программ достаточно просты и позволяют получить информацию лишь самого общего характера. Для более полного обзора необходимо применять специализированные утилиты, которые часто входят в пакеты различных диагностических программ и менеджеров памяти (например, программу Quarterdeck Manifest из пакета Quarterdeck QEMM).

Утилита MEM, несмотря на свои достаточно ограниченные возможности, позволяет получить подробные сведения о распределении памяти. Ее важнейшим достоинством является доступность, так как она входит в комплект поставки Windows '95/98 и практически всегда находится в каталоге COMMAND системного каталога Windows. Так, например, если Windows установлена в каталог C:\WINDOWS, то MEM находится в каталоге C:\WINDOWS\COMMAND. Таким образом, использование MEM - это простейший способ получить информацию о состоянии памяти системы.

В простейшем случае утилита MEM запускается без параметров и в этом режиме выдает всю основную информацию, сведенную в несколько таблиц. Самая первая таблица показывает информацию о программных модулях, расположенных в стандартной памяти с указанием их размера. Вторая таблица представляет собой карту использования адресного пространства, в которой указаны объемы используемой стандартной и расширенной памяти. В последних строчках вывода MEM показан ряд дополнительных сведений, в частности, указано местонахождение ядра DOS (в нижней памяти или в HMA). Для получения некоторых дополнительных сведений можно использовать различные ключи при запуске MEM. Чаще всего используются ключи /C, /D, /F и /P. Ключ /P включает режим постраничного вывода и может быть использован в сочетании с любыми другими ключами. Прочие ключи являются взаимоисключающими. Использование ключа /C приводит к выводу той же информации, что и в режиме без параметров. Ключ /F показывает свободные участки памяти, а ключ /D используется для получения наиболее полной информации о распределении памяти, которая может быть использована в целях отладки. Примеры запуска:

MEM /P - постраничный вывод информации о распределении памяти

MEM /D /P - вывод детальных сведений о распределении памяти

Задание к работе

Цель работы: закрепить знания о логической организации памяти, получить навыки использования специализированных программ для получения сведений о

распределении памяти, исследовать влияние менеджеров памяти на ее распределение.
1. Работая в Windows '9x, выполнить запуск утилиты MEM, рассмотреть и проанализировать выводимые сведения.
2. Перезагрузиться в режиме MS-DOS, выполнить запуск MEM, проанализировать текущее распределении памяти.
3. Выполнить полную перезагрузку компьютера, по нажатию клавиши F5 в момент загрузки отключить драйвера. Исследовать распределение памяти.

Вопросы по теории
1. Понятие о регионах. Вычисление адресов регионов.
2. Охарактеризовать применение расширенной и дополнительной памяти.
3. Структура верхней памяти.
4. Область HMA и особенности адресации процессоров 8086 и 80286.
5. Особенности адресации в защищенном режиме.

Критерии оценки:

«5» - за разработанную и корректно функционирующую программу и отчет, содержащий все указанные разделы

«4» - за разработанную и корректно функционирующую программу и отчет, не содержащий все указанные разделы

«3» - за разработанную, но не корректно функционирующую программу и отчет, не содержащий все указанные разделы

«2» - за невыполнение разработки программы

Практическая работа №5. Управление устройством вывода

Цель работы: Практическое знакомство с управлением вводом/выводом в операционных системах Windows и кэширования операций ввода/вывода.

Краткие теоретические сведения:

Необходимость обеспечить программам возможность осуществлять обмен данными с внешними устройствами и при этом не включать в каждую двоичную программу соответствующий двоичный код, осуществляющий собственно управление устройствами ввода/вывода, привела разработчиков к созданию системного программного обеспечения и, в частности, самих операционных систем.

Программирование задач управления вводом/выводом является наиболее сложным и трудоемким, требующим очень высокой квалификации. Поэтому код, позволяющий осуществлять операции ввода/вывода, стали оформлять в виде системных библиотечных процедур; потом его стали включать не в системы программирования, а в операционную систему с тем, чтобы в каждую отдельно взятую программу его не вставлять, а только позволить обращаться к такому коду. Системы программирования стали генерировать обращения к этому системному коду ввода/вывода и осуществлять только подготовку к собственно операциям ввода/вывода, то есть автоматизировать преобразование данных к

соответствующему формату, понятному устройствам, избавляя прикладных программистов от этой сложной и трудоемкой работы. Другими словами, системы программирования вставляют в машинный код необходимые библиотечные подпрограммы ввода/вывода и обращения к тем системным программным модулям, которые, собственно, и управляют операциями обмена между оперативной памятью и внешними устройствами.

Таким образом, управление вводом/выводом — это одна из основных функций любой ОС. Одним из средств правления вводом/выводом, а также инструментом управления памятью является диспетчер задач Windows, он отображает приложения, процессы и службы, которые в текущий момент запущены на компьютере. С его помощью можно контролировать производительность компьютера или завершать работу приложений, которые не отвечают.

При наличии подключения к сети можно также просматривать состояние сети и параметры ее работы. Если к компьютеру подключились несколько пользователей, можно увидеть их имена, какие задачи они выполняют, а также отправить им сообщение.

Также управлять процессами можно и «вручную» при помощи командной строки.

Команды Windows для работы с процессами:

- at — запуск программ в заданное время
- Schtasks — настраивает выполнение команд по расписанию
- Start — запускает определенную программу или команду в отдельном окне.
- Taskkill — завершает процесс
- Tasklist — выводит информацию о работающих процессах

Для получения более подробной информации, можно использовать центр справки и поддержки или команду help (например: help at)

- command.com — запуск командной оболочки MS-DOS
- cmd.exe — запуск командной оболочки Windows

Ход работы:

Задание 1. Работа с Диспетчером задач Windows 7.

1. Запустите Windows 7
2. Запуск диспетчера задач можно осуществить двумя способами:

1) Нажатием сочетания клавиш Ctrl+Alt+Del. При использовании данной команды не стоит пренебрегать последовательностью клавиш. Появится меню, в котором курсором следует выбрать пункт «Диспетчер задач».

2) Переведите курсор на область с показаниями системной даты и времени и нажмите правый клик, будет выведено меню, в котором следует выбрать «Диспетчер задач».

3. Будет выведено окно как на рис. 1.

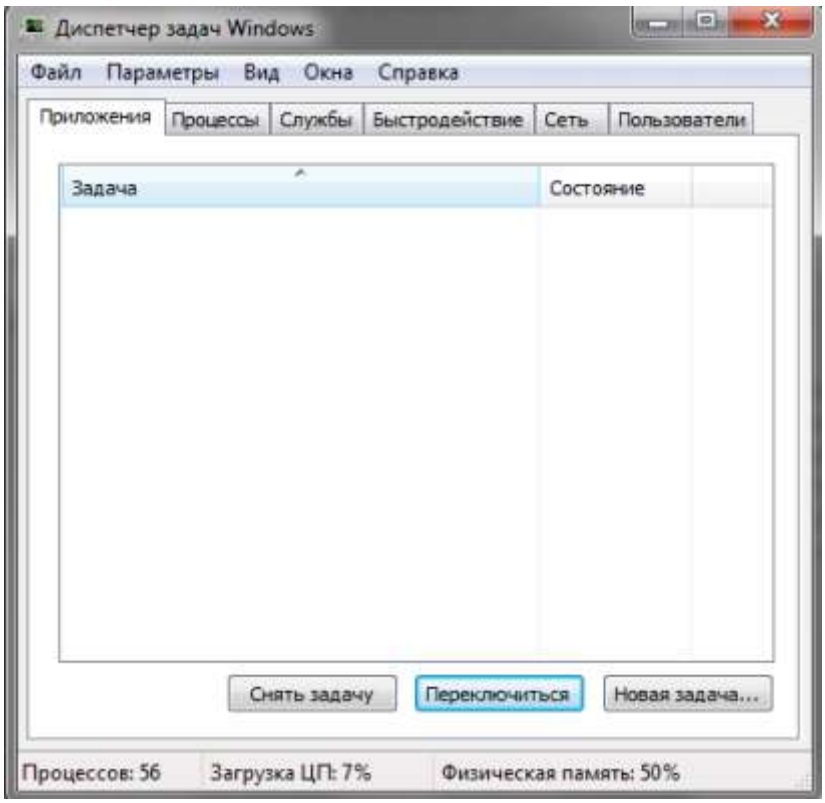


Рис. 1. Диспетчер задач Windows 7.

4. В диспетчере задач есть 6 вкладок:

1) Приложения
2) Процессы
3) Службы
4) Быстродействие
5) Сеть
6) Пользователи

- Вкладка «Приложения» отображает список запущенных задач (программ) выполняющиеся в настоящий момент не в фоновом режиме, а также отображает их состояние. Также в данном окне можно снять задачу переключиться между задачами и запустить новую задачу при помощи соответствующих кнопок.
- Вкладка «Процессы» отображает список запущенных процессов, имя пользователя запустившего процесс, загрузку центрального процессора в процентном соотношении, а также объем памяти используемого для выполнения процесса. Также присутствует возможность отображать процессы всех пользователей, либо принудительного завершения процесса. Процесс — выполнение пассивных инструкций компьютерной программы на процессоре ЭВМ.
- Вкладка «Службы» показывает, какие службы запущены на компьютере. Службы
- — приложения, автоматически запускаемые системой при запуске ОС Windows и выполняющиеся вне зависимости от статуса пользователя.

- Вкладка «Быстродействие» отображает в графическом режиме загрузку процессора, а также хронологию использования физической памяти компьютера. Очень эффективным инструментом наблюдения является «Монитор ресурсов». С его помощью можно наглядно наблюдать за каждой из сторон «жизни» компьютера. Подробное изучение инструмента произвести самостоятельно, интуитивно.
- Вкладка «Сеть» отображает подключенные сетевые адаптеры, а также сетевую активность.
- Вкладка «Пользователи» отображает список подключенных пользователей.
- Потренируйтесь в завершении и повторном запуске процессов.
- Разберите мониторинг загрузки и использование памяти.
- Попытайтесь запустить новые процессы при помощи диспетчера, для этого можно использовать команды: cmd, msconfig.

5. После изучения диспетчера задач:

Задание 2. Командная строка Windows.

1. Для запуска командной строки в режиме Windows следует нажать:

(Пуск) > «Все программы» > «Стандартные» > «Командная строка»

2. Поработайте выполнением основных команд работы с процессами: запуская, отслеживая и завершая процессы.

Основные команды

Schtasks — выводит выполнение команд по расписанию

Start — запускает определенную программу или команду в отдельном окне.
Taskkill — завершает процесс

Tasklist — выводит информацию о работающих процессах

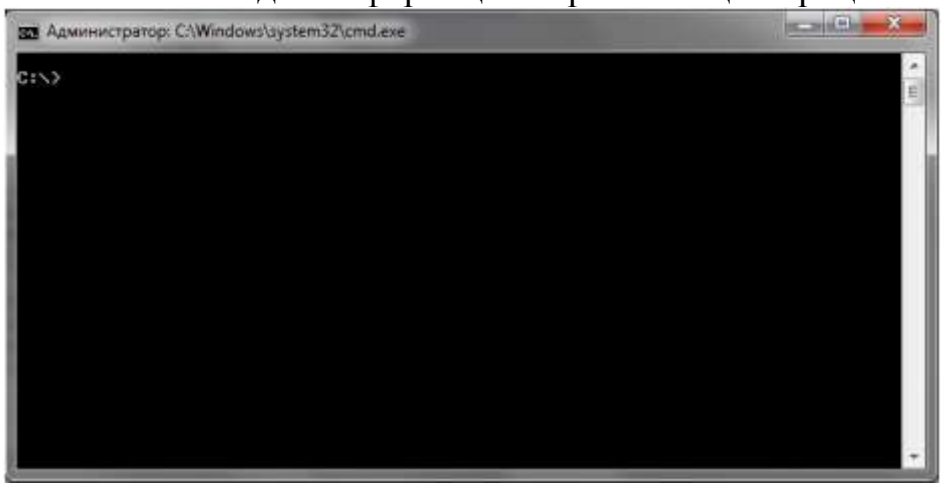


Рис. 2. Командная строка Windows 7.

3. В появившемся окне (рис. 2) наберите:

cd\ — переход в корневой каталог;

cd windows – переход в каталог Windows.

dir — просмотр содержимого каталога.

В данном каталоге мы можем работать с такими программами как «WordPad» и «Блокнот».

4. Запустим программу «Блокнот»:

C:\Windows > start notepad.exe

Отследим выполнение процесса: C:\Windows > tasklist

Затем завершите выполнение процесса: C:\Windows > taskkill /IM notepad.exe

5. Самостоятельно, интуитивно, найдите команду запуска программы WordPad.

Необходимый файл запуска найдите в папке Windows.

6. Выполнение задания включить в отчет по выполнению лабораторной работы.

Задание 3. Самостоятельное задание.

1. Отследите выполнение процесса explorer.exe при помощи диспетчера задач и командной строки.
2. Продемонстрируйте преподавателю завершение и повторный запуск процесса explorer.exe из:
   - Диспетчера задач;
   - Командной строки.
3. Выполнение задания включить в отчет по выполнению лабораторной работы.

Контрольные вопросы:

1. Дайте понятие процессу в операционной системе.
2. Дайте понятие службе в операционной системе.
3. Причислите основные команда работы с процессами при помощи командной строки.

Критерии оценки:

«5» - за разработанную и корректно функционирующую программу и отчет, содержащий все указанные разделы

«4» - за разработанную и корректно функционирующую программу и отчет, не содержащий все указанные разделы

«3» - за разработанную, но не корректно функционирующую программу и отчет, не содержащий все указанные разделы

«2» - за невыполнение разработки программы

Практическая работа №6. Организация выполнение подпрограмм

В программировании часто встречаются ситуации, когда одинаковые действия необходимо выполнять многократно в разных частях программы (например, вычисление функции *sinx*). При этом с целью экономии памяти не следует многократно повторять одну и ту же последовательность команд — достаточно один раз написать так называемую *подпрограмму* (в терминах языков высокого уровня — процедуру) и обеспечить правильный вызов этой подпрограммы и возврат в точку вызова по завершению подпрограммы. Для *вызова* подпрограммы необходимо указать ее начальный адрес в памяти и передать (если необходимо) параметры — те исходные данные, с которыми будут выполняться предусмотренные в подпрограмме действия. Адрес подпрограммы указывается в команде вызова CALL, а параметры могут передаваться через определенные ячейки

памяти, регистры или стек. *Возврат* в точку вызова обеспечивается сохранением адреса текущей команды (содержимого регистра PC) при вызове и использованием в конце подпрограммы команды возврата RET, которая возвращает сохраненное значение адреса возврата в PC.

Для реализации механизма вложенных подпрограмм (возможность вызова подпрограммы из другой подпрограммы и т. д.) адреса возврата целесообразно сохранять в стеке. *Стек* ("магазин") — особым образом организованная безадресная память, доступ к которой осуществляется через единственную ячейку, называемую *верхушкой стека.* При записи слово помещается в верхушку стека, предварительно все находящиеся в нем слова смещаются вниз на одну позицию; при чтении извлекается содержимое верхушки стека (оно при этом из стека исчезает), а все оставшиеся слова смещаются вверх на одну позицию. Такой механизм напоминает действие магазина стрелкового оружия (отсюда и второе название). В программировании называют такую дисциплину обслуживания LIFO (Last In First Out, последним пришел — первым вышел) в отличие от дисциплины типа *очередь* — FIFO (First In First Out, первым пришел — первым вышел).

В обычных ОЗУ нет возможности перемещать слова между ячейками, поэтому при организации стека перемещается не массив слов относительно неподвижной верхушки, а верхушка относительно неподвижного массива. Под стек отводится некоторая область ОЗУ, причем адрес верхушки хранится в специальном регистре процессора — указателе стека SP.

В стек можно поместить содержимое регистра общего назначения по команде PUSH или извлечь содержимое верхушки в регистр общего назначения по команде POP. Кроме того, по команде вызова подпрограммы CALL значение программного счетчика PC (адрес следующей команды) помещается в верхушку стека, а по команде RET содержимое верхушки стека извлекается в PC. При каждом обращении в стек указатель SP автоматически модифицируется.

В большинстве ЭВМ стек "растет" в сторону меньших адресов, поэтому перед каждой записью содержимое SP уменьшается на 1, а после каждого извлечения содержимое SP увеличивается на 1. Таким образом, SP всегда указывает на верхушку стека.

В процессе организации циклов мы будем использовать новые возможности системы команд модели ЭВМ, которые позволяют работать с новым классом памяти — сверхоперативной (регистры общего назначения — РОН). В реальных ЭВМ доступ в РОН занимает значительно меньшее время, чем в ОЗУ; кроме того, команды обращения с регистрами короче команд обращения к памяти. Поэтому в РОН размещаются наиболее часто используемые в программе данные, промежуточные результаты, счетчики циклов, косвенные адреса и т. п.

В системе команд учебной ЭВМ для работы с РОН используются специальные команды, мнемоники которых совпадают с мнемониками соответствующих команд для работы с ОЗУ, но в адресной части содержат символы регистров RO—R9.

Кроме обычных способов адресации (прямой и косвенной) в регистровых командах используются два новых — постинкрементная (@R+) и предекрементная

(-@R). Кроме того, к регистровым относится команда организации цикла JRNZ R,M. По этой команде содержимое указанного в команде регистра уменьшается на 1, и если в результате вычитания содержимого регистра не равно 0, то управление передается на метку M. Эту команду следует ставить в конце тела цикла, метку M — в первой команде тела цикла, а в регистр R помещать число повторений цикла.

Демонстрационный пример

Даны три массива чисел. Требуется вычислить среднее арифметическое их максимальных элементов. Каждый массив задается двумя параметрами: адресом первого элемента и длиной.

Очевидно, в программе трижды необходимо выполнить поиск максимального элемента массива, поэтому следует написать соответствующую подпрограмму.

Параметры в подпрограмму будем передавать через регистры: R1 — начальный адрес массива, R2 — длина массива.

Рассмотрим конкретную реализацию этой задачи. Пусть первый массив начинается с адреса 085 и имеет длину 14 элементов, второй— 100 и 4, третий— 110 и 9. Программа будет состоять из основной части и подпрограммы. Основная программа задает параметры подпрограмме, вызывает ее и сохраняет результаты работы подпрограммы в рабочих ячейках. Затем осуществляет вычисление среднего арифметического и выводит результат на устройство вывода. В качестве рабочих ячеек используются регистры общего назначения R6 и R7 — для хранения максимальных элементов массивов. Подпрограмма получает параметры через регистры R1 (начальный адрес массива) и R2 (длина массива). Эти регистры используются подпрограммой в качестве регистра текущего адреса и счетчика цикла соответственно. Кроме того, R3 используется для хранения текущего максимума, а R4 — для временного хранения текущего элемента. Подпрограмма возвращает результат через аккумулятор. В таблице приведен текст основной программы и подпрограммы. Обратите внимание, цикл в подпрограмме организован с помощью команды JRNZ, а модификация текущего адреса— средствами постинкрементной адресации.

Задание 1.

Набрать, скомпилировать, ознакомиться с работой программы демонстрационного примера.

Команда	Примечания
Основная программа	
RD #85	Загрузка
WR R1	параметров
RD #14	первого
WR R2	массива
CALL M	Вызов подпрограмм
WR R6	Сохранение результата
RD #100	Загрузка

WR R1	параметров
RD #4	второго
WR R2	массива
CALL M	Вызов подпрограммы
WR R7	Сохранение результата
RD #110	Загрузка
WR R1	параметров
RD #9	третьего
WR R2	массива
CALL M	Вызов подпрограммы
ADD R7	Вычисление
ADD R6	среднего
DIV #3	арифметического
OUT	Вывод результата
Подпрограмма MAX	
HLT	Стоп
M: RD @R1	Загрузка
WR R3	первого элемента в R3
L2: RD @R1+	Чтение элемента и модификация адреса
WR R4	сравнение
SUB R3	и замена,
JS L1	если R3<R4
MOV R3,R4	
L1: JRNZ R2, L2	Цикл
RD R3	Чтение результата в Acc
RET	Возврат

Задание 2.

Составить и отладить программу учебной ЭВМ для решения следующей задачи. Три массива в памяти заданы начальными адресами и длинами (первый массив начинается с адреса 085 и имеет длину 14 элементов, второй— 100 и 4, третий— 110 и 9.). Вычислить и вывести на устройство вывода среднее арифметическое суммы элементов этих массивов.

Отчет по практической работе должен содержать следующие разделы:

1. Формулировка варианта задания.
2. Блок-схема алгоритма основной программы.
3. Блок-схема алгоритма подпрограммы.
4. Распределение памяти (размещение в ОЗУ переменных, программы и необходимых констант).
5. Тексты программы и подпрограммы
6. Значения исходных данных и результата выполнения программы. (в виде таблицы)

Критерии оценки:

«5» - за разработанную и корректно функционирующую программу из второго задания и отчет, содержащий все указанные разделы

«4» - за разработанную и корректно функционирующую программу из второго задания и отчет, не содержащий все указанные разделы

«3» - за рассмотренную программу из первого задания, но не составленную программу по вариантам

«2» - за выполнение менее одного задания практической работы

РАЗДЕЛ 4. АРХИТЕКТУРА УЧЕБНОЙ ЭВМ

Практическая работа №7. Программирование разветвляющегося процесса.

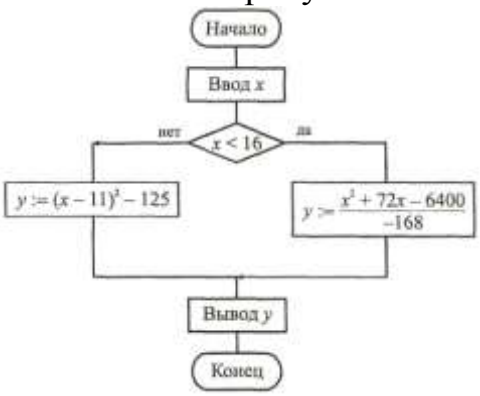
Ход работы.

Задание 1.

Рассмотрите пример реализации алгоритма кусочно-ломаной функции:

$$y=\begin{cases}(x-11)^2-125, & \text{при } x\geq 16,\\ \dfrac{x^2+72x-6400}{-168}, & \text{при } x<16,\end{cases}$$

причем *x* вводится с устройства ввода IR, результат *y* выводится на OR. Блок-схема алгоритма решения задачи показана на рисунке



В данной практической работе используются двухсловные команды с непосредственной адресацией, позволяющие оперировать отрицательными числами и числами по модулю, превышающие 999, в качестве непосредственного операнда. Оценив размер программы примерно в 20—25 команд, отведем для области данных ячейки ОЗУ, начиная с адреса 030. Составленная программа с комментариями представлена в виде таблицы.

Адрес	Команда		Примечание
	Мнемокод	Код	
000	IN	010 000	Ввод x
001	WR 30	220 030	Размещение x в ОЗУ
002	SUB #16	241 016	Сравнение с границей – (x-16)
003	JS 010	130 010	Переход по отрицательной разности

004	RD 30	210 030	Вычисление по первой формуле
005	SUB #11	241 011	
006	WR 31	220 031	
007	MUL 31	250 031	
008	SUB #125	241 125	
009	JMP 020	100 020	Переход на вывод результата
010	RD 30	210 030	Вычисление по второй формуле
011	MUL 30	250 030	
012	WR 31	220 031	
013	RD 30	210 030	
014	MUL #72	251 072	
015	ADD 31	230 031	
016	ADI 106400	430 000	
017		106 400	
018	DIVI 100168	460 000	
019		100 168	
020	OUT	020 000	Вывод результатата
021	HLT	090 000	Стоп

Задание 2

1. Разработать программу вычисления и вывода значения функции:

$$y = \begin{cases} Fi(x), \text{при x} > \text{a} \\ Fj\ (x), \text{при x} < \text{a} \end{cases}$$

для вводимого из IR значения аргумента x. Функции и допустимые пределы изменения аргумента приведены в табл. 1, варианты заданий — в табл. 2

2. Исходя из допустимых пределов изменения аргумента функций (табл. 1) и значения параметра *a* для своего варианта задания (табл. 2) выделить на числовой оси Ox области, в которых функция y вычисляется по представленной в п. 1 формуле, и недопустимые значения аргумента. На недопустимых значениях аргумента программа должна выдавать на OR максимальное отрицательное число: 199 999.

3. Ввести текст программы в окно Текст программы, при этом возможен набор и редактирование текста непосредственно в окне Текст программы или загрузка текста из файла, подготовленного в другом редакторе.

4. Ассемблировать текст программы, при необходимости исправить синтаксические ошибки.

5. Отладить программу. Для этого:

а. записать в IR значение аргумента $x > a$ (в области допустимых значений);

б. записать в PC стартовый адрес программы;

в. проверить правильность выполнения программы (т. е. правильность результата и адреса останова) в автоматическом режиме. В случае наличия ошибки выполнить пп. 5, г и 5, *д;* иначе перейти к п. 5, *е;*

г. записать в PC стартовый адрес программы;

д. наблюдая выполнение программы в режиме Шаг, найти команду, являющуюся причиной ошибки; исправить ее; выполнить пп. *5, а* — 5, *в;*

е. записать в IR значение аргумента $x < a$ (в области допустимых значений); выполнить пп. 5, б и 5, в;

ж. записать в IR недопустимое значение аргумента *x* и выполнить пп. 5, б и 5, *в.*

6. Для выбранного допустимого значения аргумента *x* наблюдать выполнение отлаженной программы в режиме Шаг и записать содержимое регистров ЭВМ перед выполнением каждой команды.

Таблица 1. Функции.

k	$F_k(x)$		k	$F_k(x)$
1	$\frac{x+17}{1-x}$		5	$\frac{(x+2)^2}{15}$
2	$\frac{(x+3)^2}{x}$		6	$\frac{2x^2+7}{x}$
3	$\frac{100}{x+10}$		7	$\frac{x^2+2x}{10}$
4	$(x+3)^3$		8	$\frac{8100}{x^2}$

Таблица 2. Варианты задания

Номер варианта	i	j	a		Номер варианта	i	j	a
1	2	1	12		8	8	6	30
2	4	3	-20		9	2	6	25
3	8	4	15		10	5	7	50
4	6	1	12		11	2	4	18
5	5	2	50		12	8	1	12
6	7	3	15		13	7	6	25
7	6	2	11		14	1	4	5

Отчет по практической работе должен содержать следующие разделы:

1. Формулировка варианта задания.
2. Блок-схема алгоритма решения задачи.
3. Размещение данных в ОЗУ.
4. Программа в форме таблицы
5. Последовательность состояний регистров ЭВМ при выполнении программы в режиме Шаг для одного значения аргумента.
6. Результаты выполнения программы для нескольких значений аргумента, выбранных самостоятельно.

Критерии оценки:

«5» - за разработанную и корректно функционирующую программу из второго задания и отчет, содержащий все указанные разделы

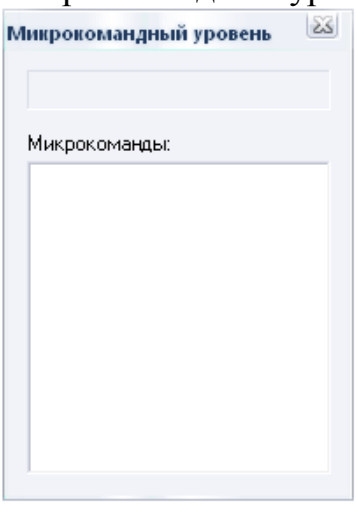
«4» - за разработанную и корректно функционирующую программу из второго задания и отчет, не содержащий все указанные разделы
«3» - за рассмотренную программу из первого задания, но не составленную программу по вариантам
«2» - за выполнение менее одного задания практической работы

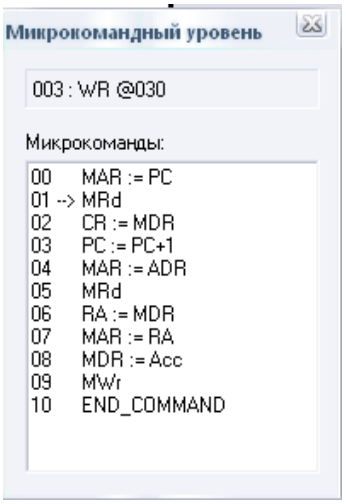
Практическая работа №8. Программирование цикла с переадресацией. Командный цикл процессора

Ход работы:

Реализация программы в ЭВМ сводится к последовательному выполнению команд. Каждая команда, в свою очередь, выполняется как последовательность микрокоманд, реализующих элементарные действия над операционными элементами процессора.

В программной модели учебной ЭВМ предусмотрен **Режим микрокоманд,** в котором действие командного цикла реализуется и отображается на уровне микрокоманд. Список микрокоманд текущей команды выводится в специальном окне Микрокомандный уровень





Окно Микрокомандный уровень используется только в режиме микрокоманд, который устанавливается командой Режим микрокоманд меню Работа. В это окно выводится мнемокод выполняемой команды, список микрокоманд, ее реализующих, и указатель на текущую выполняемую микрокоманду.

Шаговый режим выполнения программы или запуск программы в автоматическом режиме с задержкой командного цикла позволяет наблюдать процесс выполнения программы на уровне микрокоманд.

Если открыть окно Микрокомандный уровень, не установив режим микрокоманд в меню Работа, то после начала выполнения программы в режиме Шаг (или в автоматическом режиме) в строке сообщений окна будет выдано сообщение "Режим микрокоманд неактивен".

Задание 1

Выполнить последовательность команд по своему варианту задания см. табл., но в режиме Шаг.

Таблица Варианты задания 1

№	IR	Команда 1	Команда 2	Команда 3	Команда 4	Команда 5
1	007	IN	MUL @2	WR 10	WR @10	JNS 001
2	X	RD #17	SUB #9	WR 1	WR @16	JNS 001
3	100029	IN	ADD #16	WR 8	WR @8	JS 001
4	X	RD #2	MUL #6	WR 11	WR @11	JNZ 00
5	000016	IN	WR 8	DIV #4	WR @8	JMP 002
6	X	RD #4	WR 11	RD @11	ADD #330	JS 000
7	000000	IN	WR 9	RD @9	SUB #1	JS 001
8	X	RD 4	SUB #8	WR 8	WR@8	JNZ 001
9	100005	IN	ADD #12	WR 10	WR @10	JS 004
10	X	RD 4	ADD #15	WR 13	WR @13	JMP 001
11	000315	IN	SUB #308	WR 11	WR @11	JMP 001
12	X	RD #988	ADD #19	WR 9	WR @9	JNZ 001
13	000017	IN	WR 11	ADD 11	WR @11	JMP 002
14	X	RD #5	MUL #9	WR 10	WR @10	JNZ 001

Зарегистрировать изменения состояния процессора и памяти в форме таблице, в которой приведены состояния ЭВМ при выполнении демонстрационного примера.

Адрес (PC)	Мнемокод	Микрокоманда	ОЗУ		CR			АУ		Ячейки	
			MAR	MDR	COP	TA	ADR	ACC	DR	020	030

Окно Процессор обеспечивает доступ ко всем регистрам и флагам процессора.

□ Программно-доступные регистры и флаги:

- Ace — аккумулятор;
- PC — счетчик адреса команды, содержащий адрес текущей команды;
- SP — указатель стека, содержащий адрес верхушки стека;
- RB — регистр базового адреса, содержащий базовый адрес;
- RA — регистр адреса, содержащий исполнительный адрес при косвенной адресации;
- IR — входной регистр;
- OR — выходной регистр;
- I — флаг разрешения прерываний.

□ Системные регистры и флаги:

- DR — регистр данных АЛУ, содержащий второй операнд;
- MDR — регистр данных ОЗУ;
- MAR — регистр адреса ОЗУ;
- RDR — регистр данных блока РОН;

- RAR — регистр адреса блока РОН;
- CR — регистр команд, содержащий поля:

D COP — код операции;

° TA — тип адресации;

D ADR — адрес или непосредственный операнд;

- Z — флаг нулевого значения Асе;
- S — флаг отрицательного значения Асе;
- OV — флаг переполнения.

Регистры Acc, DR, IR, OR, CR и все ячейки ОЗУ и РОН имеют длину 6 десятичных разрядов, регистры PC, SP, RA и RB — 3 разряда. В окне Процессор отражаются текущие значения регистров и флагов.

Задание 2

Записать последовательность микрокоманд для следующих команд модели учебной ЭВМ:

☐ ADD R3
☐ ADD @R3
☐ ADD @R3+
☐ ADD -@R3
☐ JRNZ R3,M
☐ MOV R4,R2
☐ JMP M
☐ CALL M
☐ RET: PUSH R3
☐ POP R5

Отчет по практической работе должен содержать следующие разделы:

7. Формулировка варианта задания.

8. Распределение памяти (размещение в ОЗУ переменных, программы и необходимых констант).

9. Тексты программы и подпрограммы

10.Значения исходных данных и результата выполнения программы. (в виде таблицы)

Задание 3.

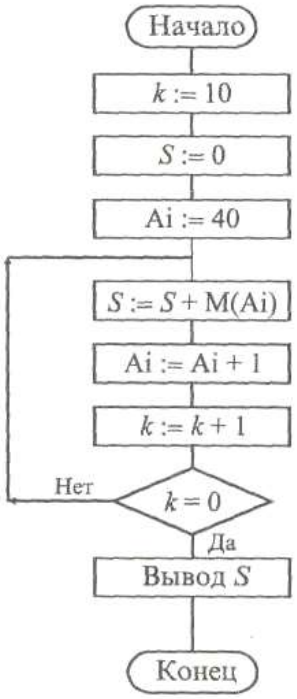
Рассмотрите демонстрационный пример

Разработать программу вычисления суммы элементов массива чисел C_1, $C_2 \ldots C_n$. Исходными данными в этой задаче являются: n — количество суммируемых чисел и $C_1, C_2, \ldots, C_n$ — массив суммируемых чисел. Заметим, что должно выполняться условие $n > 1$, т. к. алгоритм предусматривает, по крайней мере, одно суммирование. Кроме того, предполагается, что суммируемые числа записаны в ОЗУ подряд, т. е. в ячейки памяти с последовательными адресами. Результатом является сумма S.

Составим программу для вычисления суммы со следующими конкретными параметрами: число элементов массива— 10, элементы массива расположены в

ячейках ОЗУ по адресам 040, 041, 042,..., 049. Используемые для решения задачи промежуточные переменные имеют следующий смысл: A_t — адрес числа C_i, i €{l, 2,..., 10}; ОЗУ(A_i) — число по адресу A_i,, S — текущая сумма; $к$ — счетчик цикла, определяющий число повторений тела цикла.

Распределение памяти таково. Программу разместим в ячейках ОЗУ, начиная с адреса 000, примерная оценка объема программы — 20 команд; промежуточные переменные: A_i — в ячейке ОЗУ с адресом 030, $к$ — по адресу 031, S — по адресу 032. Блок-схема алгоритма показана на рисунке, текст программы с комментариями приведен в таблице.



Адрес	Команда	Примечание
000	RD #40	Загрузка начального адреса массива 040
001	WR 30	Запись в ячейку 30
002	RD #10	Загрузка параметра цикла k=10 в ячейку 031
003	WR 31	
004	RD #0	Загрузка начального значения суммы S=0
005	WR 32	Запись в ячейку 032
006	M1: RD 32	Добавление
007	ADD @30	к текущей сумме
008	WR 32	Очередного элемента массива
009	RD 30	Модификация текущего
010	ADD #1	Адреса массива
011	WR 30	(переход к следующему адресу)

012	RD 31	Уменьшение счетчика
013	SUB #1	(параметра цикла)
014	WR 31	На 1
015	JNZ M1	Проверка параметра цикла и переход при k<>0
016	RD 32	Вывод
017	OUT	Результата
018	HLT	Стоп

Задание 4.

Написать программу определения заданной характеристики последовательности чисел Ci $C_2, \ldots C_n$. Варианты заданий приведены в таблице.

1. Записать программу в мнемокодах, введя ее в поле окна **Текст программы.**
2. Сохранить набранную программу в виде текстового файла и произвести ассемблирование мнемокодов.
3. Загрузить в ОЗУ необходимые константы и исходные данные.
4. Отладить программу.

Варианты заданий

Номер варианта	Характеристика последовательности чисел
1	Количество четных чисел
2	Номер минимального числа
3	Произведение всех чисел
4	Номер первого отрицательного числа
5	Количество чисел равных C1
6	Количество отрицательных чисел
7	Максимальное отрицательное число
8	Номер первого положительного числа
9	Минимальное положительное число
10	Номер максимального числа
11	Количество нечетных чисел
12	Количество чисел меньших C1
13	Разность сумм четных и нечетных элементов массива
14	Отношение сумм четных и нечетных элементов массива
15	Минимальное положительное число
16	Номер первого отрицательного числа

Примечание. Под четными (нечетными) элементами массивов понимаются элементы массивов, имеющие четные (нечетные) индексы. Четные числа— элементы массивов, делящиеся без остатка на 2.

Отчет по практической работе должен содержать следующие разделы:

1. Формулировка варианта задания.

2. Блок – схема алгоритма решения задачи.

3. Распределение памяти (размещение в ОЗУ переменных, программы и необходимых констант).

4. Значения исходных данных, программа и результата выполнения программы в виде таблицы

Адрес	**Команда**		**Примечание**	**Состояние аккумулятора ACC**	**Состояние ячеек памяти**
	Мнемокод	Код			

Критерии оценки:

«5» - за разработанную и корректно функционирующую программу из второго задания и отчет, содержащий все указанные разделы

«4» - за разработанную и корректно функционирующую программу из второго задания и отчет, не содержащий все указанные разделы

«3» - за рассмотренную программу из первого задания, но не составленную программу по вариантам

«2» - за выполнение менее одного задания практической работы

Практическая работа № 9. Подпрограммы и стек

Подпрограммы и стек.

Ход работы.

В программировании часто встречаются ситуации, когда одинаковые действия необходимо выполнять многократно в разных частях программы (например, вычисление функции *sinx*). При этом с целью экономии памяти не следует многократно повторять одну и ту же последовательность команд — достаточно один раз написать так называемую *подпрограмму* (в терминах языков высокого уровня — процедуру) и обеспечить правильный вызов этой подпрограммы и возврат в точку вызова по завершению подпрограммы. Для *вызова* подпрограммы необходимо указать ее начальный адрес в памяти и передать (если необходимо) параметры — те исходные данные, с которыми будут выполняться предусмотренные в подпрограмме действия. Адрес подпрограммы указывается в команде вызова CALL, а параметры могут передаваться через определенные ячейки памяти, регистры или стек. *Возврат* в точку вызова обеспечивается сохранением адреса текущей команды (содержимого регистра PC) при вызове и использованием в конце подпрограммы команды возврата RET, которая возвращает сохраненное значение адреса возврата в PC.

Для реализации механизма вложенных подпрограмм (возможность вызова подпрограммы из другой подпрограммы и т. д.) адреса возврата целесообразно сохранять в стеке. *Стек* ("магазин") — особым образом организованная

безадресная память, доступ к которой осуществляется через единственную ячейку, называемую *верхушкой стека.* При записи слово помещается в верхушку стека, предварительно все находящиеся в нем слова смещаются вниз на одну позицию; при чтении извлекается содержимое верхушки стека (оно при этом из стека исчезает), а все оставшиеся слова смещаются вверх на одну позицию. Такой механизм напоминает действие магазина стрелкового оружия (отсюда и второе название). В программировании называют такую дисциплину обслуживания LIFO (Last In First Out, последним пришел — первым вышел) в отличие от дисциплины типа *очередь* — FIFO (First In First Out, первым пришел — первым вышел).

В обычных ОЗУ нет возможности перемещать слова между ячейками, поэтому при организации стека перемещается не массив слов относительно неподвижной верхушки, а верхушка относительно неподвижного массива. Под стек отводится некоторая область ОЗУ, причем адрес верхушки хранится в специальном регистре процессора — указателе стека SP.

В стек можно поместить содержимое регистра общего назначения по команде PUSH или извлечь содержимое верхушки в регистр общего назначения по команде POP. Кроме того, по команде вызова подпрограммы CALL значение программного счетчика PC (адрес следующей команды) помещается в верхушку стека, а по команде RET содержимое верхушки стека извлекается в PC. При каждом обращении в стек указатель SP автоматически модифицируется.

В большинстве ЭВМ стек "растет" в сторону меньших адресов, поэтому перед каждой записью содержимое SP уменьшается на 1, а после каждого извлечения содержимое SP увеличивается на 1. Таким образом, SP всегда указывает на верхушку стека.

В процессе организации циклов мы будем использовать новые возможности системы команд модели ЭВМ, которые позволяют работать с новым классом памяти — сверхоперативной (регистры общего назначения — РОН). В реальных ЭВМ доступ в РОН занимает значительно меньшее время, чем в ОЗУ; кроме того, команды обращения с регистрами короче команд обращения к памяти. Поэтому в РОН размещаются наиболее часто используемые в программе данные, промежуточные результаты, счетчики циклов, косвенные адреса и т. п.

В системе команд учебной ЭВМ для работы с РОН используются специальные команды, мнемоники которых совпадают с мнемониками соответствующих команд для работы с ОЗУ, но в адресной части содержат символы регистров R0—R9.

Кроме обычных способов адресации (прямой и косвенной) в регистровых командах используются два новых — постинкрементная (@R+) и предекрементная (-@R). Кроме того, к регистровым относится команда организации цикла JRNZ R,M. По этой команде содержимое указанного в команде регистра уменьшается на 1, и если в результате вычитания содержимого регистра не равно 0, то управление передается на метку M. Эту команду следует ставить в конце тела цикла, метку M — в первой команде тела цикла, а в регистр R помещать число повторений цикла.

Демонстрационный пример

Даны три массива чисел. Требуется вычислить среднее арифметическое их максимальных элементов. Каждый массив задается двумя параметрами: адресом первого элемента и длиной.

Очевидно, в программе трижды необходимо выполнить поиск максимального элемента массива, поэтому следует написать соответствующую подпрограмму.

Параметры в подпрограмму будем передавать через регистры: R1 — начальный адрес массива, R2 — длина массива.

Рассмотрим конкретную реализацию этой задачи. Пусть первый массив начинается с адреса 085 и имеет длину 14 элементов, второй— 100 и 4, третий— 110 и 9. Программа будет состоять из основной части и подпрограммы. Основная программа задает параметры подпрограмме, вызывает ее и сохраняет результаты работы подпрограммы в рабочих ячейках. Затем осуществляет вычисление среднего арифметического и выводит результат на устройство вывода. В качестве рабочих ячеек используются регистры общего назначения R6 и R7 — для хранения максимальных элементов массивов. Подпрограмма получает параметры через регистры R1 (начальный адрес массива) и R2 (длина массива). Эти регистры используются подпрограммой в качестве регистра текущего адреса и счетчика цикла соответственно. Кроме того, R3 используется для хранения текущего максимума, а R4 — для временного хранения текущего элемента. Подпрограмма возвращает результат через аккумулятор. В таблице приведен текст основной программы и подпрограммы. Обратите внимание, цикл в подпрограмме организован с помощью команды JRNZ, а модификация текущего адреса— средствами постинкрементной адресации.

Задание 1.

Набрать, скомпилировать, ознакомиться с работой программы демонстрационного примера.

Команда	Примечания
Основная программа	
RD #85	Загрузка
WR R1	параметров
RD #14	первого
WR R2	массива
CALL M	Вызов подпрограмм
WR R6	Сохранение результата
RD #100	Загрузка
WR R1	параметров
RD #4	второго
WR R2	массива
CALL M	Вызов подпрограммы
WR R7	Сохранение результата
RD #110	Загрузка

WR R1	параметров
RD #9	третьего
WR R2	массива
CALL M	Вызов подпрограммы
ADD R7	Вычисление
ADD R6	среднего
DIV #3	арифметического
OUT	Вывод результата
Подпрограмма MAX	
HLT	Стоп
M: RD @R1	Загрузка
WR R3	первого элемента в R3
L2: RD @R1+	Чтение элемента и модификация адреса
WR R4	сравнение
SUB R3	и замена,
JS L1	если R3<R4
MOV R3,R4	
L1: JRNZ R2, L2	Цикл
RD R3	Чтение результата в Acc
RET	Возврат

Задание 2.

Составить и отладить программу учебной ЭВМ для решения следующей задачи. Три массива в памяти заданы начальными адресами и длинами (первый массив начинается с адреса 085 и имеет длину 14 элементов, второй— 100 и 4, третий— 110 и 9.). Вычислить и вывести на устройство вывода среднее арифметическое суммы элементов этих массивов.

Отчет по практической работе должен содержать следующие разделы:

11.Формулировка варианта задания.

12.Блок-схема алгоритма основной программы.

13.Блок-схема алгоритма подпрограммы.

14.Распределение памяти (размещение в ОЗУ переменных, программы и необходимых констант).

15.Тексты программы и подпрограммы

16.Значения исходных данных и результата выполнения программы. (в виде таблицы)

Критерии оценки:

«5» - за разработанную и корректно функционирующую программу из второго задания и отчет, содержащий все указанные разделы

«4» - за разработанную и корректно функционирующую программу из второго задания и отчет, не содержащий все указанные разделы

«3» - за рассмотренную программу из первого задания, но не составленную программу по вариантам

«2» - за выполнение менее одного задания практической работы

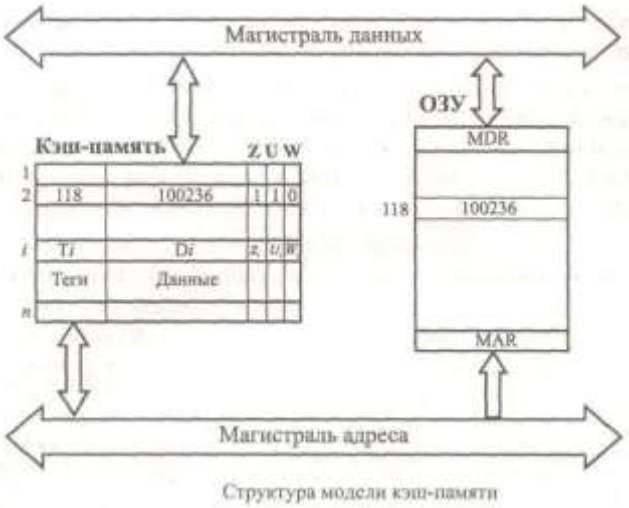
Практическая работа №10. Принципы работы кэш-памяти

Принципы работы кэш-памяти

Ход работы.

Программная модель кэш-памяти

К описанной программной модели учебной ЭВМ может быть подключена программная модель кэш-памяти, структура которой в общем виде отображена на рисунке. Конкретная реализация кэш-памяти в описываемой программной модели показана на рисунке:



Структура модели кэш-памяти

Кэш-память содержит N ячеек (в модели N может выбираться из множества {4, 8, 16, 32}), каждая из которых включает трехразрядное поле тега (адреса ОЗУ), шестиразрядное поле данных и три однобитовых признака (флага):

Z — признак занятости ячейки;

U — признак использования;

W — признак записи в ячейку.

Таким образом, каждая ячейка кэш-памяти может дублировать одну любую ячейку ОЗУ, причем отмечается ее занятость (в начале работы модели все ячейки кэш-памяти свободны, VZi = 0), факт записи информации в ячейку во время пребывания ее в кэш-памяти, а также использование ячейки (т. е. любое обращение к ней).

Текущее состояние кэш-памяти отображается на экране в отдельном окне в форме таблицы, причем количество строк соответствует выбранному числу ячеек кэш. Столбцы таблицы определяют содержимое полей ячеек, например, так, как показано в таблице:

Таблица Пример текущего состояния кэш-памяти

	Теги	Данные	Z	U	W
1	012	220152	1	0	0
2	013	211003	1	1	0
3	050	000025	1	1	1
4	000	000000	0	0	0

Для настройки параметров кэш-памяти можно воспользоваться диалоговым окном Кэш-память, вызываемым командой Вид | Кэш-память, а затем нажать первую кнопку на панели инструментов открытого окна. После этих действий появится диалоговое окно Параметры кэш-памяти, позволяющее выбрать размер кэш-памяти, способ записи в нее информации и алгоритм замещения ячеек.

При сквозной записи при кэш-попадании в процессорных циклах записи осуществляется запись как в ячейку кэш-памяти, так и в ячейку ОЗУ, а при обратной записи — только в ячейку кэш-памяти, причем эта ячейка отмечается битом записи *($W_t:=1$)*. При очистке ячеек, отмеченных битом записи, необходимо переписать измененное значение ноля данных в соответствующую ячейку ОЗУ.

При кэш-промахе следует поместить в кэш-память адресуемую процессором ячейку. При наличии свободных ячеек кэш-памяти требуемое слово помещается в одну из них (в порядке очереди). При отсутствии свободных ячеек следует отыскать ячейку кэш-памяти, содержимое которой можно удалить, записав на его место требуемые данные (команду). Поиск такой ячейки осуществляется с использованием алгоритма замещения строк.

В модели реализованы три различных алгоритма замещения строк:

- случайное замещение, при реализации которого номер ячейки кэш-памяти выбирается случайным образом;
- очередь, при которой выбор замещаемой ячейки определяется временем пребывания ее в кэш-памяти;
- бит использования, случайный выбор осуществляется только из тех ячеек, которые имеют нулевое значение флага использования.

Напомним, что бит использования устанавливается в 1 при любом обращении к ячейке, однако, как только все биты U_i установятся в 1, все они тут же сбрасываются в 0, так что в кэш всегда ячейки разбиты на два непересекающихся подмножества по значению бита U — те, обращение к которым состоялось относительно недавно {после последнего сброса вектора U) имеют значение U = 1, иные — со значением U = 0 являются "кандидатами на удаление" при использовании алгоритма замещения "бит использования".

Если в параметрах кэш-памяти установлен флаг "с учетом бита записи", то все три алгоритма замещения осуществляют поиск "кандидата на удаление" прежде всего среди тех ячеек, признак записи которых не установлен, а при отсутствии таких ячеек (что крайне маловероятно) — среди всех ячеек кэш памяти. При снятом флаге "с учетом бита записи" поиск осуществляется по всем ячейкам кэш-памяти без учета значения W.

Задание 1

В качестве задания предлагается некоторая короткая "программа" (табл. 2), которую необходимо выполнить с подключенной кэш-памятью (размером 4 и 8 ячеек) в шаговом режиме для следующих двух вариантов алгоритмов замещения

Режим записи	Алгоритм замещения
Сквозная	С3, без учета бита записи
Обратная	О, с учетом бита записи

Таблица. Варианты задания

№ варианта	Номера команд программы						
	1	2	3	4	5	6	7
1	RD #12	WR 10	WR @10	ADD 12	WR R0	SUB 10	PUSH R0
2	RD #65	WRR 2	MOV R4,R2	WR 14	PUSH R2	POP R3	CALL 002
3	RD #16	SUB #5	WR 9	WR @9	WR R3	PUSH R3	POP R4
4	RD #99	WR R6	MOV R7,R6	ADD R7	PUSH R7	CALL 006	POP R8
5	RD #11	WR R2	WR - @R2	PUSH R2	CALL 005	POP R3	RET
6	RD #19	SUB #10	WR 9	ADD #3	WR @9	CALL 006	POPR 4
7	RD #6	CALL 006	WR 11	WRR 2	PUSH R2	RET	JMP 002
8	RD #8	WRR 2	WR @R2+	PUSH r2	POP R3	WR - @R3	CALL 003
9	RD #13	WR 14	WR @14	WR @13	ADD 13	CALL 006	RET
10	RD #42	SUB #54	WR 16	WR @16	WRR 1	ADD @R1+	PUSH R1
11	RD #10	WRR 5	ADD R5	WRR 6	CALL 005	PUSH R6	RET
12	JMP 006	RD #76	WR 14	WRR 2	PUSH R2	RET	CALL 001

Не следует рассматривать заданную последовательность команд как фрагмент программы. Некоторые конструкции, например, последовательность команд PUSH R6, RET в общем случае не возвращает программу в точку вызова подпрограммы. Такие группы команд введены в задание для того, чтобы обратить внимание студентов на особенности функционирования стека.

Порядок выполнения работы

1. Ввести в модель учебной ЭВМ текст своего варианта программы (см.табл. 2), ассемблировать его и сохранить на диске в виде txt-файла.

2. Установить параметры кэш-памяти размером 4 ячейки, выбрать режим

записи и алгоритм замещения в соответствии с первой строкой из табл. 1.

3. В шаговом режиме выполнить программу, фиксируя после каждого шага состояние кэш-памяти.

4. Для одной из команд записи (WR) перейти в режим Такт и отметить, в каких микрокомандах происходит изменение кэш-памяти.

5. Для кэш-памяти размером 8 ячеек установить параметры в соответствии со второй строкой своего варианта из табл. 1 и выполнить программу в шаговом режиме еще раз, фиксируя последовательность номеров замещаемых ячеек кэш-памяти.

Отчет по практической работе должен содержать следующие разделы:

1. Вариант задания — текст программы и режимы кэш-памяти.
2. Последовательность состояний кэш-памяти размером 4 ячейки при однократном выполнении программы (команды 1—7).
3. Последовательность микрокоманд при выполнении команды WR С отметкой тех микрокоманд, в которых возможна модификация кэш-памяти.
4. Для варианта кэш-памяти размером 8 ячеек — последовательность номеров замещаемых ячеек кэш-памяти для второго варианта параметров кэш памяти при двукратном выполнении программы (команды 1—7).

Критерии оценки:

«5» - за разработанную и корректно функционирующую программу и отчет, содержащий все указанные разделы

«4» - за разработанную и корректно функционирующую программу и отчет, не содержащий все указанные разделы

«3» - за разработанную, но не корректно функционирующую программу и отчет, не содержащий все указанные разделы

«2» - за невыполнение разработки программы

ЛИТЕРАТУРА

Основные источники:

1. Архитектура компьютерных систем [Электронный ресурс] : учебно-методический комплекс / . — Электрон. текстовые данные. — Алматы: Нур-Принт, 2015. — 179 c. — 9965-894-96-5. — Режим доступа: http://www.iprbookshop.ru/67009.html
2. Учебно-методическое пособие по дисциплине Архитектура вычислительных систем [Электронный ресурс] / . — Электрон. текстовые данные. — М. : Московский технический университет связи и информатики, 2016. — 16 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/61466.html

Дополнительные источники:

1. Архитектура ЭВМ и систем [Электронный ресурс] : учебное пособие / Ю.Ю. Громов [и др.]. — Электрон. текстовые данные. — Тамбов: Тамбовский государственный технический университет, ЭБС АСВ, 2012. — 200 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/64069.html
2. Шаманов А.П. Системы счисления и представление чисел в ЭВМ [Электронный ресурс] : учебное пособие / А.П. Шаманов. — Электрон. текстовые данные. — Екатеринбург: Уральский федеральный университет, 2016. — 52 c. — 978-5-7996-1719-6. — Режим доступа: http://www.iprbookshop.ru/66204.html
3. Архитектура компьютерных систем [Электронный ресурс] : учебно-методический комплекс / . — Электрон. текстовые данные. — Алматы: Нур-Принт, 2015. — 179 c. — 9965-894-96-5. — Режим доступа: http://www.iprbookshop.ru/67009.html
4. Методические рекомендации по организации внеаудиторной самостоятельной работы студентов по дисциплине ОП.01. Основы архитектуры, устройство и функционирование вычислительных систем, 2017г.

Internet – источники:

1. http://www.intuit.ru/ - Интернет-Университет Информационных технологий
2. http://claw.ru/ - Образовательный портал