Оглавление

Контрольная работа №1

Управляющее слово ЦП

Цель контрольной работы

Целью работы является проверка усвоения материалов по архитектуре операционного устройства центрального процессора ВМ и взаимодействия входящих в нее элементов.

Управляющее слово ЦП составляется из разрядов управления устройств, входящих в ОУ ЦП.

Для выполнения контрольной работы необходимо уяснить порядок функционирования составляющих операционное устройство элементов вычислительной техники: мультиплексоров, дешифратора, арифметико-логического устройства (АЛУ), регистров и оперативного запоминающего устройства.

Мультиплексор

Из многих информационных входов I0…I3 мультиплексор подает на выход один, соответствующий коду, установленному на управляющих входах a1, a0. Структурная схема мультиплексора имеет вид:

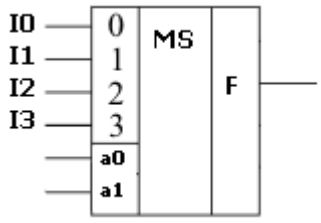

Рис. 1. Схема мультиплексора.

Таблица соответствия

a1	a0	F
0	0	I0
0	1	I1
1	0	I2
1	1	I3

Дешифратор

Это устройство, которое имеет единицу только на одном из выходов, соответствующему коду входных информационных сигналов.

Дешифратор на схемах обозначается следующим образом:

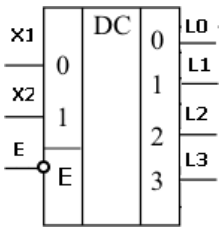

Рис. 2. Схема дешифратора.

Таблица соответствия для двухвходового дешифратора имеет вид:

E	X2	X1	L0	L1	L2	L3
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1
1	X	X	0	0	0	0

Вход E означает разрешение работы дешифратора. Этот вход обычно делают инверсным. Если разрешение есть, то E=0, и один из выходов в соответствии с кодом входных информационных сигналов дешифратора равен 1. Если E=1, то независимо от состояния входных сигналов все выходы дешифратора равны 0.

Регистр

Регистры обычно строятся на D триггерах. Обозначение регистра на схемах может иметь вид:

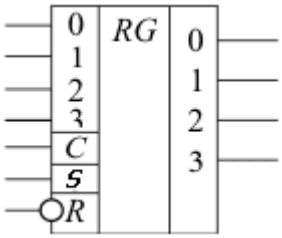

Рис. 3. Схема регистра.

На вход C подается синхросигнал. Регистр может срабатывать по переднему или по заднему фронту синхроимпульса. Вход S – управляющий. При S=0 регистр осуществляет хранение информации. При S=1 регистр переписывает свое состояние по синхросигналу в соответствии с состоянием входных информационных сигналов.

Арифметико-логическое устройство

АЛУ выполнено на основе комбинационных схем, выходные состояния которых полностью определяются входными информационными и управляющими сигналами.

Таблица микроопераций АЛУ

M	S1	S0	C	Микрооперации	Наименование
0	0	0	0	A	Передача A
0	0	0	1	A+1	Инкремент A
0	0	1	0	A+B	Сумма
0	0	1	1	A+B+1	Сумма с инкрементом
0	1	0	0	A-B-1	Разность с декрементом
0	1	0	1	A-B	Разность
0	1	1	0	A-1	Декркмент A
0	1	1	1	A	Передача A
1	0	0	X	A∨B	ИЛИ
1	0	1	X	$A \oplus B$	Искл. ИЛИ
1	1	0	X	A^B	И
1	1	1	X	$\bar{A}$	Не A

Операционное устройство

Операционное устройство (ОУ) осуществляет операции (арифметические и логические) над входными информационными сигналами. Характер операций, входные и выходные сигналы (регистры, вход, выход и оперативное запоминающее устройство) определяются управляющими сигналами устройств, входящих в состав ОУ.

Управляющее слово ВМ составляется из управляющих разрядов MSA, MSB, АЛУ, DS, MSM, WR.

MSA			MSB			АЛУ				DC				MSM	WR
a0	**a1**	**a2**	**a0**	**a1**	**a2**	**M**	**S1**	**S0**	**C0**	**E**	**X3**	**X2**	**X1**	**a0**	**WR**

Задача устройства управления – обеспечить выдачу управляющего слова ЦП. Иначе он называется микрокод, который управляет работой аппаратуры ЦП. Обычно он записывается в постоянное запоминающее устройство (ПЗУ) центрального процессора (прошивается).

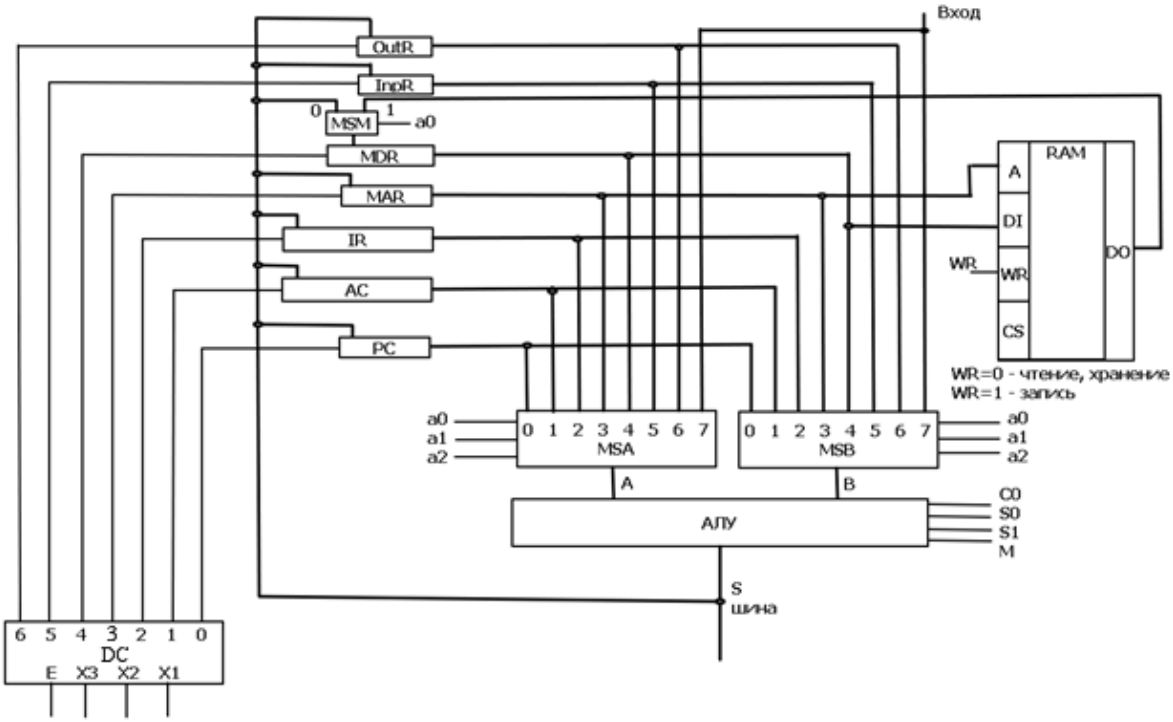

Рис. 4. Схема операционного устройства.

Задача контрольной работы состоит в том, чтобы расписать управляющее слово ЦП для различных микроопераций.

Задание

№ №варианта	**Записать управляющее слово**
1	S← INPR ∧ IR
2	INPR← IR ⊕ MAR
3	IR← MAR ∨ OUTR
4	MAR← OUTR – AC
5	OUTR←AC – MDR – 1
6	AC← MDR + PC + 1
7	MDR← PC + INPR
8	OUTR← INPR
9	S← INPR + 1
10	INPR← IR + MAR
11	IR← MAR + OUTR + 1
12	MAR← OUTR – AC – 1
13	OUTR← AC – MDR
14	AC← IR ⊕ MDR
15	MDR← IR ⊕ AC
16	PC← MAR ∧ PC
17	OUTR← IR – 1
18	IR← MDR ∨ IR
19	PC← INPR ⊕ OUTR
20	S← MDR ∧ IR
21	INPR← MAR + OUTR
22	MDR← OUTR + IR + 1
23	AC← IR – PC – 1

№ №варианта	Записать управляющее слово
24	MAR← INPR – AC
25	PC← MAR ⊕ PC
26	IR← OUTR + PC
27	OUTR← MDR + INPR + 1
28	S← INPR – AC – 1
29	MDR← AC – MAR
30	INPR← IR ∨ MDR
31	AC← INPR ⊕ MDR
32	MAR← AC ∧ IR
33	S← OUTR
34	S← AC+OUTR
35	S← IR+INPR+1
36	S←OUTR – MDR
37	S←INPR – MAR – 1
38	S←IR+1
39	S←OUTR – 1
40	S← AC ⊕ PC

Выводы

Сделать выводы по проделанной работе.

Контрольная работа №2.
Устройство управления ЦП учебной ВМ

Цель контрольной работы

Проверить навыки студентов программировать ПЗУ микропрограмм устройства управления ЦП учебной ВМ.

Последовательность работы устройства управления задается циклами.

Для начала рассмотрим цикл выборки команды. Ведь прежде чем выполнить команду, надо выбрать её из памяти ВМ.

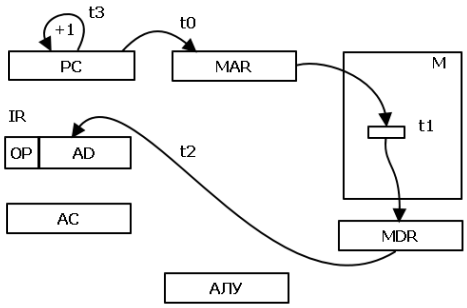

Рис. 5. Цикл выборки команды

Для выборки команды из памяти ВМ надо, во-первых, поместить адрес команды из регистра адреса команды в регистр адреса памяти, во-вторых, прочитать команду из ячейки памяти в регистр данных памяти, в-третьих. Переслать команду в регистр команды, в-четвертых, подготовить счетчик команд для выборки следующей команды – инкременировать PC.

То есть, последовательно надо выполнить 4 действия, обозначенные на рис. 12 t0, t1, t2, t3. Следовательно, цикл выборки команды осуществляется за 4 такта.

Само выполнение команды происходит следующим образом:

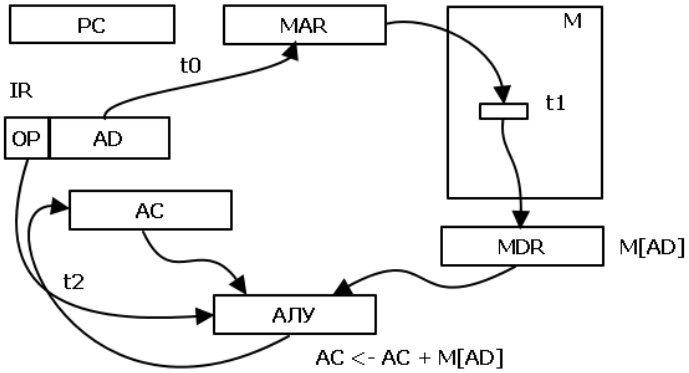

Рис. 5. Цикл выполнения команды

Рассмотрим цикл выполнения команды на примере операции сложения.

Нулевой такт – пересылка адреса операнда из регистра команды в регистр адреса памяти.

Первый такт – выборка из памяти операнда и размещение его в регистре данных памяти.

Второй такт – собственно выполнение сложения в АЛУ с размещением результата в аккумуляторе.

Здесь мы рассматриваем выполнение команды с прямой адресацией, когда в адресном поле команды находится адрес операнда.

Машину Фон Неймана стали совершенствовать, поскольку для выполнения некоторых команд более удобными являются другие методы адресации операндов. Всего способов адресации насчитывается около 20. Из их в каждой конкретной ВМ используется только часть, по выбору конструкторов ВМ и в соответствии с назначением ВМ (ориентированием ВМ на обработку определённых видов информации).

Допустим, что у нас в памяти содержится такая информация:

Адрес	Содержание
30	40
40	50
50	60

Рассмотрим выполнение команды LDA (загрузку аккумулятора) с помощью различных методов адресации.

Прямая адресация: LDA 30. /загрузить в аккумулятор слово из ячейки с адресом 30. В результате выполнения команды AC ← 40.

Непосредственная адресация: LDA # 30. /загрузить в аккумулятор слово «30». В результате выполнения команды AC ← 30. То есть в команде содержится сам операнд, и загрузка его из памяти не требуется.

Косвенная адресация: LDA @ 30. /загрузить в аккумулятор слово из ячейки, адрес которой находится в ячейке 30. В результате выполнения команды AC ← 50. То есть в команде содержится не адрес операнда, а адрес адреса операнда. Соответственно, для подготовки операции требуется двойное обращение к памяти.

Существуют и другие способы адресации, которые здесь мы не будем рассматривать.

В общем случае для указания способа адресации в команде выделяется отдельное поле, содержащее необходимое количество разрядов. Для декодирования способа адресации при этом необходим ещё один цикл – цикл дешифрации команды.

Таким образом, мы получили наличие в УУ ЦП следующих циклов:

1. Цикл выборки команды – С0;
2. Цикл дешифрации команды – С1;
3. Цикл выполнения команды – С2;
4. Цикл обработки прерывания – С3.

Последний цикл необходим для обработки требований прерывания, вырабатываемых внешними устройствами ВМ (например, клавиатурой, устройством ввода/вывода и т.д.) или при появлении нештатных ситуаций при выполнении команд (деление на ноль, переполнение разрядов и др.). Как происходит цикл обработки прерывания мы рассмотрим позднее – при изучении операционных систем.

На основании изложенного можно сформулировать алгоритм работы УУ ЦП:

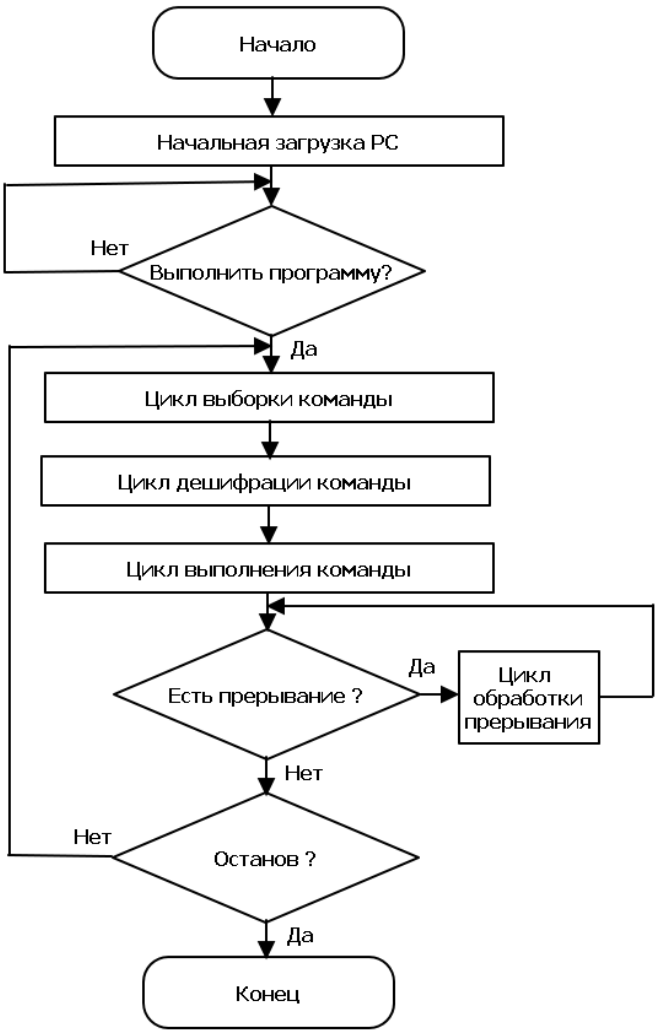

Рис. 7. Алгоритм работы устройства управления ЦП ВМ

Как видите, ВМ реагирует на прерывание только после окончания выполнения очередной команды программы. В противном случае очень трудно за-

помнить текущее состояние ЦП для возврата в программу после обработки прерывания.

Изобразим структурную схему УУ ЦП.

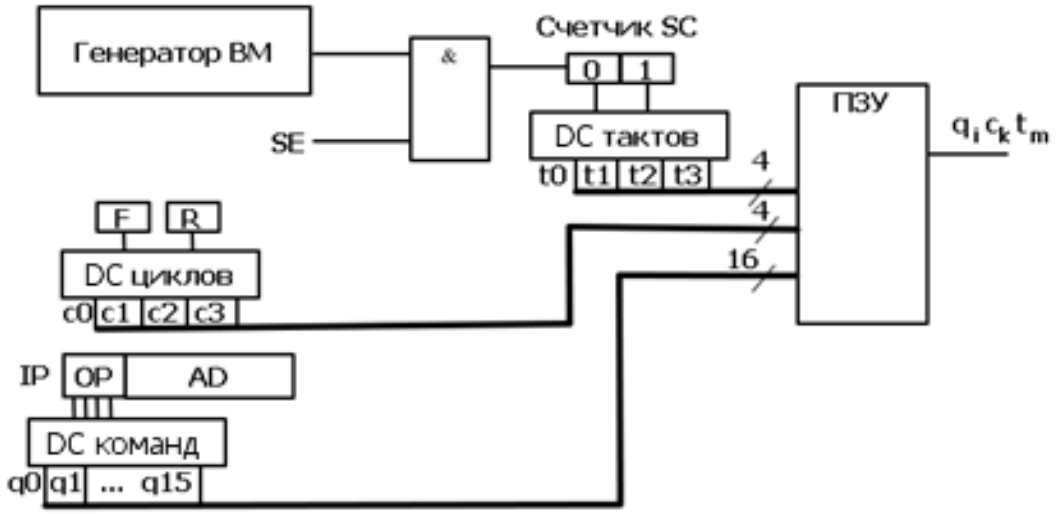

Рис. 8. Структурная схема УУ ЦП

В основе УУ имеется ПЗУ, сигналы, вырабатываемые на его выходе, управляют работой ВМ. Генератор ВМ вырабатывает тактовую частоту. Сигнал SE представляет собой сигнал разрешения работы и останов. При SE=0 сигналы генератора ВМ не проходят через схему совпадения, и схема управления не работает – останов. При SE=1 сигналы генератора ВМ проходят через схему совпадения и поступают на вход двухразрядного счетчика, соединённого по выходу с дешифратором тактов. Логика работы последнего приведена на рис. 15.

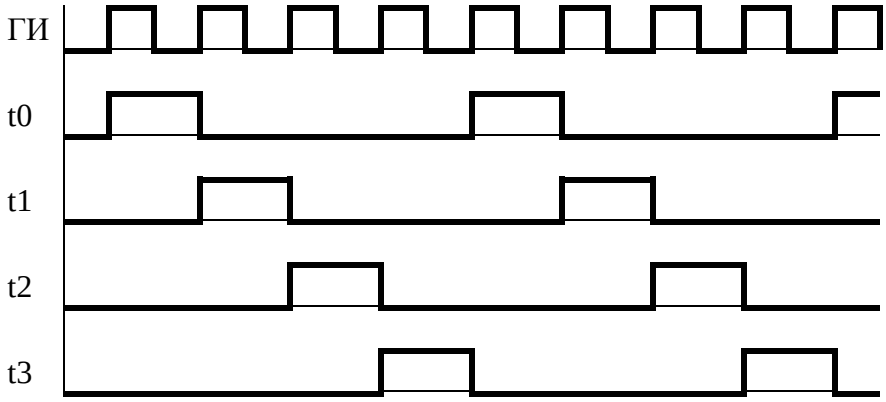

Рис. 9. Временная диаграмма работы DS тактов

В ВМ такты не пересекаются по времени.

DC циклов работает по выходам двух триггеров F и R. В зависимости от состояния этих триггеров формируется сигнал на соответствующем его выходе.

F	R	Цикл	Обозначение цикла
0	0	Выборки команды	C_0
0	1	Дешифрации команды	C_1
1	0	Выполнения команды	C_2
1	1	Прерывания	C_3

DC команды работает по состоянию разрядов регистра команд, соответствующих полю кода операции. В нашем случае код имеет 4 разряда, что соответствует 16 выходам дешифратора команд.

Выходы всех трех дешифраторов составляют адрес ПЗУ, в котором записана (прошита) информация соответствующая содержимому управляющего слова процессора данного такта данного цикла данной команды.

Такова в общем виде структура УУ ЦП.

Таблица команд учебной ВМ:

№ и обозначение команды	Команда	Код команды		Адрес	Расшифровка команды
		2	16		
0 q_0	LDA	0000	0	AD	AC←M[AD]
1 q_1	STA	0001	1	AD	M[AD] ← AC
2 q_2	ADD	0010	2	AD	AC← AC+ M[AD]
3 q_3	SUB	0011	3	AD	AC← AC- M[AD]
4 q_4	AND	0100	4	AD	AC← AC∧M[AD]
5 q_5	OR	0101	5	AD	AC← AC∨M[AD]
6 q_6	XOR	0110	6	AD	AC← AC$\overline{\oplus}$M[AD]
7 q_7	COM	0111	7	AD	AC← $\overline{M[AD]}$ (обратный код)
8 q_8	NEG	1000	8	AD	AC← $\overline{M[AD]}$ + 1(доп. код)
9 q_9	INC	1001	9	AD	AC←M[AD]+1
10 q_{10}	DEC	1010	A	AD	AC←M[AD]-1
11 q_{11}	JMP	1011	B	AD	PC←AD (переход)
12 q_{12}	CLEA	1100	C	AD=FFF*	AC←0
13 q_{13}	INCA	1101	D	AD=FFF*	AC← AC+1
14 q_{14}	INP	1110	E	AD=FFF*	AC← INPR (ввод)
15 q_{15}	OUT	1111	F	AD=FFF*	OUTR ← AC (вывод)

Примечание *: AD=FFF означает заполнение единицами адресного поля.

Обычно ВМ содержит команды различной длины – безадресные, одноадресные, двухадресные… В нашей учебной ВМ мы рассматриваем все команды единой длины равной 16 двоичным разрядам. Поэтому безадресные команды (с номерами от 12 до 15) содержат фиктивный адрес, равный FFF.

Рассмотрим для примера команду сложения ADD. Для неё в поле кода операции будет иметься значение 2 16-ричного кода. В адресном поле находит-

ся 12-разрядный адрес AD. Напишем микрокоманды циклов выборки и выполнения команды.

F=0, R=0, C0=1

C0t0 : MAR← PC / подготовить выборку команды из памяти.

C0t1 : MDR← M[MAR] / прочесть команду из памяти по адресу MAR.

C0t2 : IR← MDR / поместить команду в регистр команды.

C0t3 : PC← PC +1, F← 1/ подготовить адрес след. команды, перейти к циклу выполнения команды.

F=1, R=0, C2=1

C2t0 : MAR← IR[AD] / передать адрес операнда в регистр адреса памяти MAR.

C2t1 : MDR← M[MAR] / прочесть операнд из памяти по адресу MAR.

C2t2 : AC← AC+MDR / выполнить сложение.

C2t3 : F← 0/ перейти к циклу выборки команды.

Так выполняются микрокоманды ВМ в циклах выборки и выполнения команды сложения. Аналогичным образом можно описать порядок выполнения любой команды из перечня команд ВМ.

ВМ фон Неймана первоначально была рассчитана, как и наша учебная ВМ, только для одного типа адресации – прямой адресации. Для других типов адресации, что задаётся обычно в специальном поле команды, требуется наличие цикла дешифрации команды и осуществление выборки операндов соответствующим образом. Мы цикл дешифрации команды не рассматриваем, поскольку считаем, что используется только прямая адресация операндов.

Задание

Составить микрооперации каждого такта циклов выборки и выполнения команды, соответствующей номеру студента по списку группы. Если номер больше 15, то из номера по списку группы надо вычесть 16 и выбрать команду с соответствующим номером. Определить вариант прошивки ПЗУ УУ ЦП для данной команды. Результаты оформить в виде таблицы по следующему примеру:

Команда, цикл, такт	Микрокоманда	MSA	MSB	АЛУ	DC	MSM	WR	FR
C_0t_0 :	MAR← PC	000	xxx	0000	0011	x	0	00
C_0t_1 :	MDR← M[MAR]	xxx	xxx	xxxx	0100	1	0	00
C_0t_2 :	IR← MDR	100	xxx	0000	0010	x	0	00
C_0t_3:	PC←PC+1, F←1	000	xxx	0001	0000	x	0	10
$q_2C_2t_0$:	MAR← IR[AD]	010	xxx	0000	0011	x	0	10
$q_2C_2t_1$:	MDR← M[MAR]	xxx	xxx	xxxx	0100	1	0	10

$q_2C_2t_2$:	AC← AC+MDR	001	100	0010	0001	x	0	10
$q_2C_2t_3$:	F← 0	xxx	xxx	xxxx	1xxx	x	0	00

Выводы

Сделать выводы по проделанной работе.

Контрольная работа №3.
Команды учебной ЭВМ

Цель контрольной работы

Целью работы является проверка усвоения архитектуры устройства управления и операционного устройства центрального процессора ВМ и взаимодействия входящих в неё элементов.

Таблица команд учебной ВМ:

№ и обозначение команды	Команда	Код команды		Адрес	Расшифровка команды
		2	16		
0 q_0	LDA	0000	0	AD	AC←M[AD]
1 q_1	STA	0001	1	AD	M[AD] ← AC
2 q_2	ADD	0010	2	AD	AC← AC+ M[AD]
3 q_3	SUB	0011	3	AD	AC← AC- M[AD]
4 q_4	AND	0100	4	AD	AC← AC∧M[AD]
5 q_5	OR	0101	5	AD	AC← AC∨M[AD]
6 q_6	XOR	0110	6	AD	AC← AC$\oplus$M[AD]
7 q_7	COM	0111	7	AD	AC← $\overline{M[AD]}$ (обратный код)
8 q_8	NEG	1000	8	AD	AC← $\overline{M[AD]} + 1$(доп. код)
9 q_9	INC	1001	9	AD	AC←M[AD]+1
10 q_{10}	DEC	1010	A	AD	AC←M[AD]-1
11 q_{11}	JMP	1011	B	AD	PC←AD (переход)
12 q_{12}	CLEA	1100	C	AD=FFF*	AC←0
13 q_{13}	INCA	1101	D	AD=FFF*	AC← AC+1
14 q_{14}	INP	1110	E	AD=FFF*	AC← INPR (ввод)
15 q_{15}	OUT	1111	F	AD=FFF*	OUTR ← AC (вывод)

Примечание *: AD=FFF означает заполнение единицами адресного поля.

Обычно ВМ содержит команды различной длины – безадресные, одноадресные, двухадресные… В учебной ВМ мы рассматриваем все команды единой длины равной 16 двоичным разрядам. Поэтому безадресные команды (с номерами от 12 до 15) содержат фиктивный адрес, равный FFF.

Рассмотрим для примера команду сложения ADD. Для неё в поле кода операции будет иметься значение 2 16-ричного кода. В адресном поле находится 12-разрядный адрес AD. Напишем микрокоманды циклов выборки и выполнения команды.

F=0, R=0, C_0=1

C0t0 : MAR← PC / подготовить выборку команды из памяти.
C0t1 : MDR← M[MAR] / прочесть команду из памяти по адресу MAR.
C0t2 : IR← MDR / поместить команду в регистр команды.
C0t3 : PC← PC +1, F← 1/ подготовить адрес след. команды, перейти к циклу

выполнения команды.

F=1, R=0, C_2=1

C2t0 : MAR← IR[AD] / передать адрес операнда в регистр адреса памяти MAR.
C2t1 : MDR← M[MAR] / прочесть операнд из памяти по адресу MAR.
C2t2 : AC← AC+MDR / выполнить сложение.
C2t3 : F← 0/ перейти к циклу выборки команды.

Так выполняются микрокоманды ВМ в циклах выборки и выполнения команды сложения. Аналогичным образом можно описать порядок выполнения любой команды из перечня команд ВМ.

Контрольная работа состоит в том, чтобы вы научились строить команды из микрокоманд.

Рассмотрим типовую задачу и приведём пример её решения.

Задача. По адресу 5A8 записана команда сложения ADD с адресом F12. По этому адресу записан операнд 731F. В регистре AC находится операнд 8721. Определить информацию, которая будет иметься в регистрах ВМ PC, MAR, MDR, IR, AC после выполнения данной команды.

Решение.

1. Определяем исходное состояние регистров.
2. Выписываем микрокоманды циклов выборки команды и выполнения команды.
3. Определяем значения регистров после выполнения каждой микрокоманды.
4. Результаты оформляем в виде таблицы.

Регистры	PC	MAR	MDR	IR	AC
Исх. состояние рег.	5A8				8721
C_0t_0 : MAR← PC		5A8			
C_0t_1 : MDR← M[MAR]			2F12		
C_0t_2 : IR← MDR				2F12	
C_0t_3: PC←PC+1, F←1	5A9				
$q_2C_2t_0$: MAR← IR[AD]		F12			
$q_2C_2t_1$: MDR← M[MAR]			731F		
$q_2C_2t_2$: AC← AC+MDR					FA40
$q_2C_2t_3$: F← 0					
Результат:	5A9	F12	731F	2F12	FA40

Варианты заданий:

Вариант №1

1.Записать микрооперации цикла выборки и выполнения команды LDA.

2.По адресу 13A записана команда LDA с адресом C05. По этому адресу команды записан операнд B72E. В регистре-аккумуляторе находится операнд 3C91.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 2

1.Записать микрооперации цикла выборки и выполнения команды STA.

2.По адресу 658 записана команда STA с адресом 4C2. По этому адресу команды записан операнд F0F5. В регистре-аккумуляторе находится операнд BD02.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 3

1.Записать микрооперации цикла выборки и выполнения команды ADD.

2.По адресу 25C записана команда ADD с адресом 50A. По этому адресу команды записан операнд 631F. В регистре-аккумуляторе находится операнд 53F3.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 4

1.Записать микрооперации цикла выборки и выполнения команды SUB.

2.По адресу 3B9 записана команда SUB с адресом 465. По этому адресу команды записан операнд 1AB3. В регистре-аккумуляторе находится операнд FF32.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 5

1.Записать микрооперации цикла выборки и выполнения команды AND.

2.По адресу 3D1 записана команда AND с адресом 5C2. По этому адресу команды записан операнд FBF1. В регистре-аккумуляторе находится операнд 3028.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 6

1.Записать микрооперации цикла выборки и выполнения команды OR.

2.По адресу 6D1 записана команда OR с адресом 3AD. По этому адресу команды записан операнд C15A. В регистре-аккумуляторе находится операнд 2F74.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 7

1.Записать микрооперации цикла выборки и выполнения команды XOR.

2.По адресу 37A записана команда XOR с адресом 40B. По этому адресу команды записан операнд BA12. В регистре-аккумуляторе находится операнд CD3A.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 8

1.Записать микрооперации цикла выборки и выполнения команды COM.

2.По адресу 74C записана команда COM с адресом 508. По этому адресу команды записан операнд 52D3. В регистре-аккумуляторе находится операнд 2A7E.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 9

1.Записать микрооперации цикла выборки и выполнения команды JMP.

2.По адресу E73 записана команда JMP с адресом 8BF. По этому адресу команды записан операнд E257. В регистре-аккумуляторе находится операнд B2A3.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 10

1.Записать микрооперации цикла выборки и выполнения команды DEC.

2.По адресу 2E5 записана команда DEC с адресом 379. По этому адресу команды записан операнд 6A4B. В регистре-аккумуляторе находится операнд 56DC.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 11

1.Записать микрооперации цикла выборки и выполнения команды INC.

2.По адресу 785 записана команда INC с адресом D19. По этому адресу команды записан операнд 5246. В регистре-аккумуляторе находится операнд A7E6.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 12

1.Записать микрооперации цикла выборки и выполнения команды NEG.

2.По адресу 38E записана команда NEG с адресом 407. По этому адресу команды записан операнд AD42. В регистре-аккумуляторе находится операнд 2B0C.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 13

1.Записать микрооперации цикла выборки и выполнения команды CLEA.

2.По адресу 1F3 записана команда CLEA с адресом FFF. По этому адресу команды записан операнд FFFF. В регистре-аккумуляторе находится операнд F658.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 14

1.Записать микрооперации цикла выборки и выполнения команды INCA.

2.По адресу 2A7 записана команда INCA с адресом FFF. По этому адресу команды записан операнд FFFF. В регистре-аккумуляторе находится операнд F23E.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 15

1.Записать микрооперации цикла выборки и выполнения команды INP.

2.По адресу 83C записана команда INP с адресом FFF. По этому адресу команды записан операнд FFFF. В регистре-аккумуляторе - операнд 5A4B.Регистры INPR=AB, OUTR=DE.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC, INPR, OUTR после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 16

1.Записать микрооперации цикла выборки и выполнения команды OUT.

2.По адресу DF1 записана команда OUT с адресом FFF. По этому адресу команды записан операнд FFFF. В регистре-аккумуляторе - операнд AB12. Регистры INPR=BC, OUTR=AE.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC, INPR, OUTR после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант №17

1.Записать микрооперации цикла выборки и выполнения команды LDA.

2.По адресу 27A записана команда LDA с адресом 505. По этому адресу команды записан операнд C34A. В регистре-аккумуляторе находится операнд 5541.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 18

1.Записать микрооперации цикла выборки и выполнения команды STA.

2.По адресу 342 записана команда STA с адресом 15E. По этому адресу команды записан операнд 1F15. В регистре-аккумуляторе находится операнд CE13.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 19

1.Записать микрооперации цикла выборки и выполнения команды AND.

2.По адресу 34A записана команда ADD с адресом 14B. По этому адресу команды записан операнд 477A. В регистре-аккумуляторе находится операнд 226F.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 20

1.Записать микрооперации цикла выборки и выполнения команды SUB.

2.По адресу 12F записана команда SUB с адресом 234. По этому адресу команды записан операнд 2AB5. В регистре-аккумуляторе находится операнд FF45.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 21

1.Записать микрооперации цикла выборки и выполнения команды AND.

2.По адресу 25C записана команда AND с адресом 67B. По этому адресу команды записан операнд 377D. В регистре-аккумуляторе находится операнд 557A.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 22

1.Записать микрооперации цикла выборки и выполнения команды OR.

2.По адресу 28F записана команда OR с адресом 2CC. По этому адресу команды записан операнд CCFF. В регистре-аккумуляторе находится операнд FAC1.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 23

1.Записать микрооперации цикла выборки и выполнения команды XOR.

2.По адресу 45F записана команда XOR с адресом 50C. По этому адресу команды записан операнд AB45. В регистре-аккумуляторе находится операнд FF34.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 24

1.Записать микрооперации цикла выборки и выполнения команды COM.

2.По адресу 88A записана команда COM с адресом 609. По этому адресу команды записан операнд 54FF. В регистре-аккумуляторе находится операнд 3CCF.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 25

1.Записать микрооперации цикла выборки и выполнения команды JMP.

2.По адресу A17 записана команда JMP с адресом 99B. По этому адресу команды записан операнд CF12. В регистре-аккумуляторе находится операнд AB55.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 26

1.Записать микрооперации цикла выборки и выполнения команды DEC.

2.По адресу 33D записана команда DEC с адресом 457. По этому адресу команды записан операнд 7B11. В регистре-аккумуляторе находится операнд 543F.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 27

1.Записать микрооперации цикла выборки и выполнения команды INC.

2.По адресу 567 записана команда INC с адресом A12. По этому адресу команды записан операнд B456. В регистре-аккумуляторе находится операнд A113.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 28

1.Записать микрооперации цикла выборки и выполнения команды NEG.

2.По адресу 41F записана команда NEG с адресом 609. По этому адресу команды записан операнд CB64. В регистре-аккумуляторе находится операнд CCA9.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 29

1.Записать микрооперации цикла выборки и выполнения команды CLEA.

2.По адресу 123 записана команда CLEA с адресом FFF. По этому адресу команды записан операнд FFFF. В регистре-аккумуляторе находится операнд 458C.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 30

1.Записать микрооперации цикла выборки и выполнения команды INCA.

2.По адресу 30F записана команда INCA с адресом FFF. По этому адресу команды записан операнд FFFF. В регистре-аккумуляторе находится операнд AB1F.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 31

1.Записать микрооперации цикла выборки и выполнения команды INP.

2.По адресу 39F записана команда INP с адресом FFF. По этому адресу команды записан операнд FFFF. В регистре-аккумуляторе - операнд 172F. Регистры INPR=FE, OUTR=F6.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC, INPR, OUTR после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Вариант № 32

1.Записать микрооперации цикла выборки и выполнения команды OUT.

2.По адресу F33 записана команда OUT с адресом FFF. По этому адресу команды записан операнд FFFF. В регистре-аккумуляторе - операнд CD1E. Регистры INPR=D1, OUTR=7A.

Определить информацию, которая будет записана в регистрах PC, MAR, MDR, IP, AC, INPR, OUTR после выполнения этой команды.

Показать процесс выполнения команды. Записать ответ в строчку.

Выводы

Сделать выводы по проделанной работе.

Контрольная работа №4
Управление памятью

Цель контрольной работы

Проверить и закрепить знание студентами порядка пересчета виртуального адреса в физический адрес при страничном управлении памятью с использованием дискового пространства.

По характеру использования разделяют память на виртуальную, физическую, страничную и др. Дело в том, что обычно оперативной памяти (ОП) не хватает для исполняемых на ВМ процессов.

В качестве первого выхода из этого положения предложили так называемые оверлеи. Программист сам разбивал программу на части – оверлеи, очередной оверлей загружался в ОП после выполнения предыдущего оверлея. Такой подход частично решал проблему с нехваткой оперативной памяти, но создавал трудности для программистов.

Вторым выходом из положения стало использование виртуальной памяти с возложением на операционную систему задач, связанных с загрузкой в ОП необходимых для работы процессов частей виртуальной памяти с диска. При этом решалась задача обеспечения для программистов прозрачности работы с памятью. Вся работа с загрузкой в ОП частей процесса с диска и выгрузкой обратно выполнялась диспетчером памяти операционной системы (ОС) и становилась для программистов невидимой.

Виртуальная память – это совокупность программно-аппаратных средств, позволяющих пользователю писать и выполнять на ВМ программы, объём которых превосходит имеющуюся в ВМ оперативную (физическую) память. Для этого делают следующее:

- в исходном состоянии все данные и программный код размещают на диске;
- ОС при выполнении процесса перемещает данные и программный код между диском и ОП частями таким образом, чтобы пользователь этого не замечал;
- пользователь все время работает как бы в виртуальной памяти большого объема.

Как это осуществляется на практике?

Для этого существует три метода управления памятью: страничное распределение, сегментное распределение и сегментно-страничное.

Страничное распределение

На рис.1 приведена схема взаимодействия виртуальной и физической памяти. В диске данные хранятся на дорожках секторами, размер которых равен

512 байт. Поэтому размер страницы выбирают кратным этому значению: 1К, 2К, 4К, 8К.

Пример.

Пусть внешняя память (диск) имеет объем 1Мбайт = 2^{20} байт. Физическая память имеет объем 16 Кбайт = 2^{14} байт. Страница имеет объем 4 Кбайт = 2^{12} байт. Соответственно, для адресации виртуальной памяти требуется 20 разрядов, физической памяти - 14 разрядов, смещения внутри страницы - 12 разрядов. Кроме того, отсюда следует, что физическая память содержит 2^2 = 4 страницы, а виртуальная – 2^8 = 256 страниц.

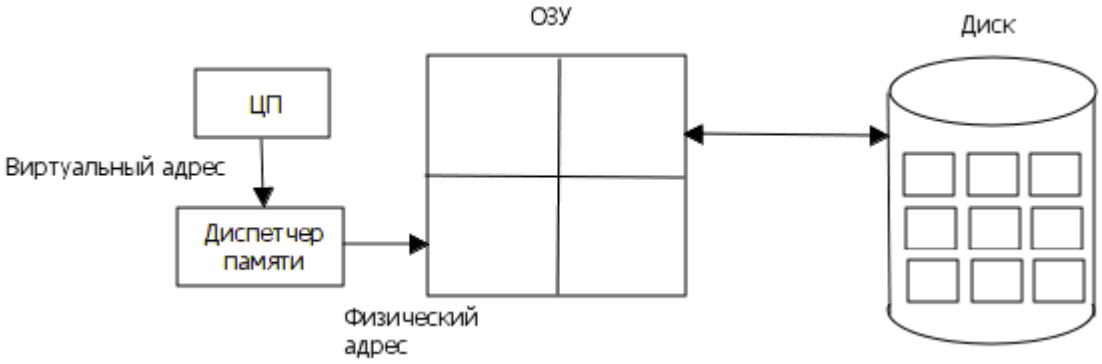

Рис.1. Взаимодействие виртуальной и физической памяти.

Виртуальный адрес состоит из двух частей: 8 старших разрядов – номер страницы, 12 младших разрядов – смещение в странице. Физический адрес тоже состоит из двух частей: 2 старших разряда – номер страницы, 12 младших разрядов – смещение в странице.

Пусть процессору потребовалось обратиться к байту на 9-й странице со смещением 2049. В физической памяти свободна страница номер 2, куда и будет переписана страница из виртуальной памяти. Надо определить виртуальный и физический адреса требуемого байта.

Для решения задачи надо построить шаблоны двух частей виртуального и физического адресов, преобразовать номера страниц виртуальной и физической памяти в двоичный код и записать их в свои места шаблонов, заполнив недостающие старшие разряды нулями. Затем преобразовать смещение в двоичный код и записать его в соответствующую часть шаблонов адреса, тоже при необходимости заполнив недостающие старшие разряды нулями.

В результате получим:

- виртуальный адрес 00001001 100000000001.
- физический адрес 10 100000000001.

Для преобразования виртуального адреса в физический диспетчер памяти ОС строит страничную таблицу:

№ вирт. стр.					№ физ. стр.	Карта диска
0						
1						
…						
$1001=9_{10}$	1				$10=2_{10}$	3 дор.2 сект.
…						
$2^8-1=255_{10}$						

В таблице обозначено:

V – признак присутствия виртуальной страницы в ОП;

R – признак использования страницы. Обычно – несколько разрядов. При каждом обращении к странице значение инкрементируется. Чем больше R, тем активнее идет обращение к странице и тем важнее оставить ее в ОП при необходимости вытеснения страницы на диск;

M – признак модификации страницы. Равен 1, если страница модифицировалась, т.е. в нее производилась запись данных. При вытеснении такой страницы её необходимо сохранить на диск прежде записи на это место другой страницы;

A – признак прав доступа к странице. Обычно составляет 2 разряда.

Правила преобразования виртуального адреса в физический при страничном распределении памяти поясняется рис.2.

Обращение к таблице страниц может быть достигнуто быстрее при использовании буфера быстрого преобразования адреса TLB (Translation Lookaside Buffer). Если страничную таблицу хранить в памяти, то при каждом обращении к памяти надо обратиться прежде всего к таблице, т.е. требуется дополнительное обращение к памяти. Поэтому наиболее используемую часть страничной таблицы размещают в ассоциативной памяти, дорогой, но малой емкости. При этом поиск в таблице производится по определенным признакам и осуществляется до 10 раз быстрее, чем в обычной памяти.

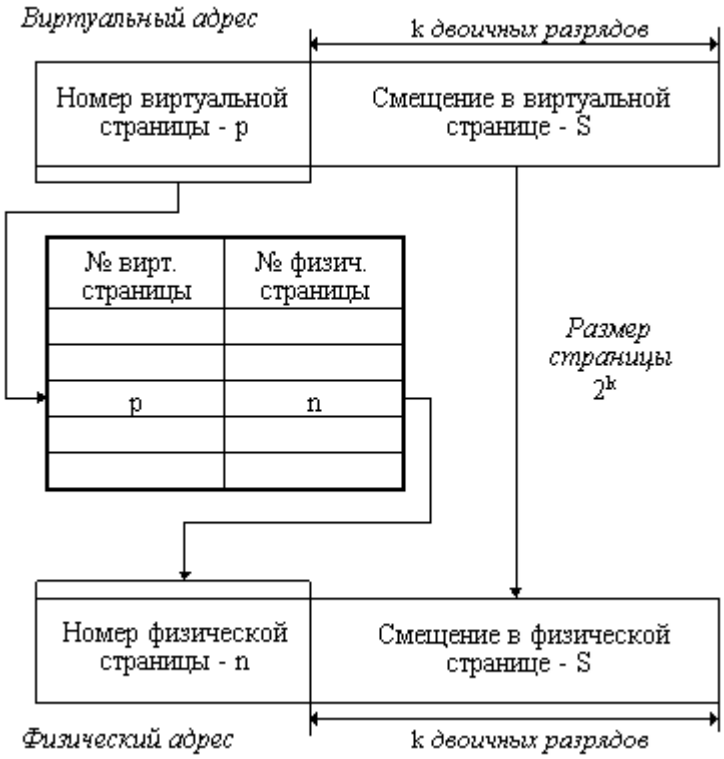

Рис.2. Преобразование виртуального адреса в физический при страничной организации памяти.

Положительными качествами рассмотренного страничного распределения памяти являются его простота и быстродействие. Недостатками являются недостаточный учет прав доступа к информации, которая передается из диска в ОП и обратно одинаковыми страницами.

Для устранения этого недостатка применяют сегментное распределение памяти.

Сегментное распределение памяти

Программиста интересует, в первую очередь, назначение информации, представленной в памяти сегментами с различными правами доступа. Поэтому сегментное распределение предусматривает передачу информации между диском и ОП целыми сегментами с одинаковыми правами доступа. При этом возможны ситуации, когда один и тот же сегмент (например, подпрограмма) используется одновременно двумя и более процессами. Поэтому не обязательно иметь много копий соответствующего кода в памяти каждого процесса, а достаточно иметь один такой сегмент в ОП и обращаться к нему из разных процессов по мере необходимости.

В случае сегментного распределения памяти для каждого процесса создается таблица сегментов, которая имеет сведения о размере сегмента, нахожде-

нии его в ОП и управляющую информацию, позволяющую определять, какой сегмент следует выгрузить из ОП при необходимости загрузки нового сегмента.

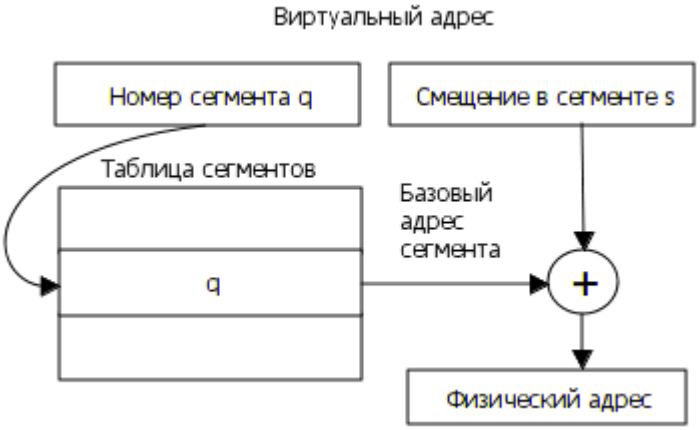

Рис.3. Вычисление виртуального адреса при сегментной организации памяти.

Фактически таблица сегментов аналогична таблице страниц при страничном распределении памяти. Отличие заключается в том, что сегменты имеют различный размер и находятся в виртуальной памяти не обязательно с начала страницы. Поэтому при вычислении физического адреса необходимо производить арифметические вычисления, учитывая базовый адрес сегмента в ОП и смещение внутри сегмента. Эти вычисления требуют больше времени по сравнению с вычислением физического адреса при страничном распределении, которое осуществляется посредством конкатенации номера страницы и смещения внутри страницы.

Кроме того, недостатком сегментного распределения является эффект фрагментации ОП ввиду неодинаковости размера сегментов.

Способом устранения этих недостатков является переход к сегментно-страничному распределению памяти.

Сегментно-страничное распределение памяти

Сегментно-страничное распределение памяти совмещает достоинства обоих вышерассмотренных способов распределения памяти. При этом виртуальный адрес состоит из трех составляющих: номера сегмента q, номера страницы p и смещения внутри страницы s. Перемещение информации между диском и ОП осуществляется страницами одинакового размера. Для вычисления физического адреса используются две таблицы: таблица сегментов и таблица страниц сегмента.

Схема определения физического адреса показана на рис. 4.

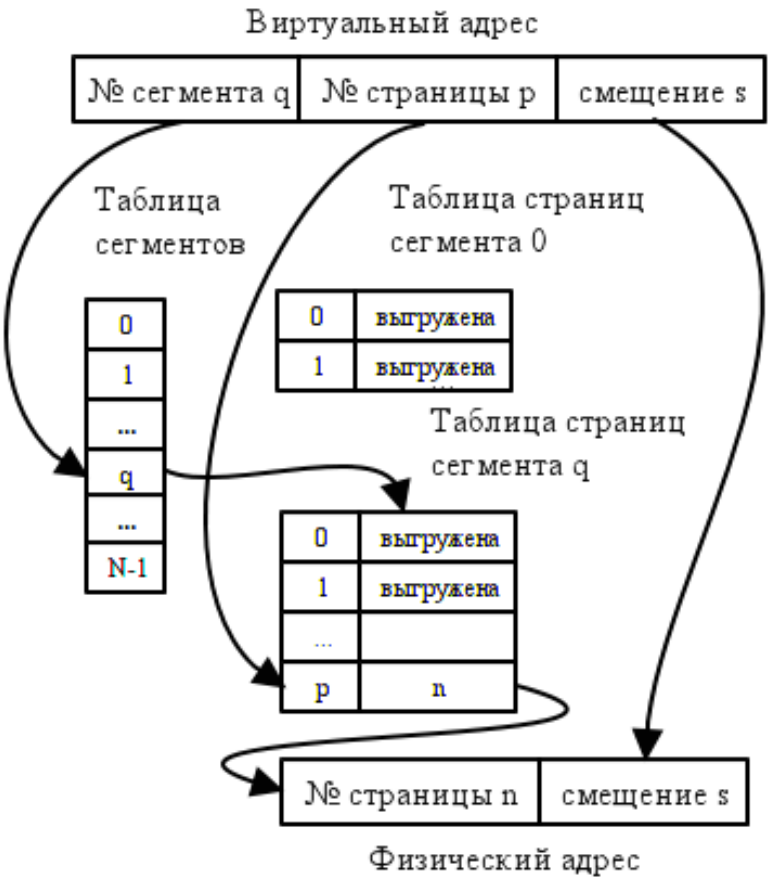

Рис.4. Вычисление адресов при сегментно-страничном распределении памяти.

Фрагментация памяти при таком распределении существенно уменьшается, поскольку передача информации осуществляется страницами одинакового размера, и неполной может быть только последняя страница сегмента. Кроме того, упрощается вычисление адресов, которое аналогично страничному распределению, но требует двойного обращения к памяти (используются две таблицы). По номеру сегмента выбирается таблица страниц сегмента, по номеру страницы в этой таблице выбирается номер страницы в физической памяти. Смещение внутри страницы виртуальной и физической памяти совпадают, поэтому используется операция конкатенации.

Задание

Виртуальная память имеет объем 16 М байт, физическая – 1 М байт, страница - 64 К байт. Слово из виртуальной памяти со страницы № 17 со смещением 224 переписывается на страницу № 11 физической памяти. Записать виртуальный и физический адреса слова в двоичном коде, разделив страницу и смещение. (В условии задачи номера страниц и величина смещения даны в десятичной системе).

Каждому студенту выдается задание (по номеру студента по списку группы) со своими значениями объемов виртуальной и физической памяти, размера страницы, номеров страниц (виртуальной и физической) и смещения в странице в соответствии с таблицей.

Варианты заданий:

№ варианта	Объем виртуальной памяти, байт	Объем физической памяти, байт	Размер страницы, байт	№ виртуальной страницы	Смещение, байт	№ физической страницы
1	8 М	512 к	2 к	35	123	33
2	16 М	1 М	4 к	48	259	17
3	32 М	4 М	8 к	46	306	23
4	1 М	256 к	16 к	99	462	7
5	512 к	128 к	1 К	77	239	32
6	4 М	512 к	16 К	12	145	21
7	2 М	512 к	32 К	24	289	8
8	8 М	256 к	64 К	23	177	2
9	16 М	1 М	2 к	45	153	22
10	32 М	32 к	4 к	24	99	6
11	1 М	1 М	8 к	7	53	19
12	512 к	128 к	16 к	32	49	5
13	4 М	256 к	2 к	15	69	31
14	2 М	16 к	2 к	13	121	5
15	8 М	512 к	4 к	14	257	13
16	16 М	64 к	8 к	25	215	3
17	32 М	512 к	16 к	27	217	23
18	1 М	256 к	1 К	29	39	19
19	512 к	128 к	16 К	24	119	7
20	4 М	512 к	32 К	23	187	4
21	2 М	128 к	8 К	12	305	13

Выводы

Сделать выводы по проделанной работе.

Контрольная работа №5
Обработка прерываний от внешних устройств

Цель контрольной работы

Целью работы является проверка усвоения последовательности выполнения операций процессором при обработке прерываний от внешних устройств.

Общая организация прерываний

Механизм прерывания обеспечивается соответствующими аппаратно-программными средствами компьютера.

Задачей аппаратных средств обработки прерывания в процессоре ЭВМ является приостановка выполнения одной программы (иногда называемой основной) и передача управления подпрограмме обработки прерывания.

Поскольку для выполнения подпрограммы обработки прерывания используются различные регистры процессора (РОНы, счетчик команд, регистр флагов и т.д.), то информацию, содержащуюся в них в момент прерывания, необходимо сохранить для последующего возврата в прерванную программу.

Обычно задача сохранения содержимого счетчика команд и регистра флагов, содержащего вектор состояния процессора, возлагается на аппаратные средства обработки прерывания. Сохранение содержимого других регистров процессора, используемых в подпрограмме обработки прерывания, производится непосредственно в подпрограмме.

Организация системы прерываний с использованием векторов прерываний

Действия, выполняемые при обработке внешних прерываний процессором, как правило, те же, что и при обращении к обычной подпрограмме; различие в том, что при обращении к подпрограмме эти действия инициируются командой, а при обработке прерывания - управляющим сигналом от контроллера внешнего устройства, называемым Запрос прерывания.

Эта важная особенность обработки прерывания программы позволяет организовать обмен данными с внешними устройствами в произвольные моменты времени, не зависящие от программы, выполняемой в ЭВМ. Таким образом, появляется возможность обмена данными с внешними устройствами в реальном масштабе времени, определяемом внешней по отношению к ЭВМ средой (например, с датчиками, следящими за состоянием технологического процесса).

Прерывание программы по требованию внешнего устройства не должно оказывать на прерванную программу никакого влияния, кроме увеличения времени ее выполнения за счет приостановки на время выполнения подпрограммы обработки прерывания.

Формирование сигналов прерываний – запросов внешних устройств на обслуживание, происходит в их контроллерах – электронных схемах, обеспечивающих связь с процессором. В серийных ЭВМ обычно используется одноуровневая система прерываний – сигналы Запрос прерывания от всех внешних устройств поступают на один вход процессора. Поэтому возникает проблема идентификации внешнего устройства, запросившего обслуживание. Основным способом, решающим проблему идентификации в большинстве современных ЭВМ в настоящее время, является использование векторов прерываний.

Внешнее устройство, запросившее обслуживание, само идентифицирует себя с помощью адреса своего вектора прерывания – ячейки основной памяти, в которой хранится адрес начала программы обработки прерывания данного типа.

Векторы всех обработчиков прерываний собраны в единую таблицу векторов прерываний, располагающуюся в самых младших адресах оперативной памяти, имеющую объем 1 Кбайт и содержащую 4-х байтные элементы (векторы прерываний) для 256 обработчиков прерываний. Так как таблица всегда имеет нулевой начальный адрес и длину вектора в 4 байта, чтобы определить адрес вектора для прерывания типа i, достаточно просто умножить это значение на 4.

Вектор прерывания выдается контроллером не одновременно с запросом на прерывание, а только по разрешению процессора (рис.1).

Регистр прерываний составлен триггерами внешних устройств, устанавливаемыми в единичное состояние требованиями прерывания соответствующих контроллеров. Выходы триггеров поступают на входы приоритетного шифратора через элементы совпадения, вторые входы которых соединены с выходами регистра маски. Разряды этого регистра устанавливаются ОС в 0 при запрете прерывания соответствующего устройства и в 1, если прерывание разрешено.

При поступлении хотя бы одного требования прерывания на входы приоритетного шифратора он устанавливает в единичное состояние свой выход E и устанавливает код приоритета прерывания на других своих выходах в соответствии с таблицей соответствия

I0	I1	I2	I3	Y	X	IST
1	X	X	X	0	0	1
0	1	X	X	0	1	1
0	0	1	X	1	0	1
0	0	0	1	1	1	1
0	0	0	0	0	0	0

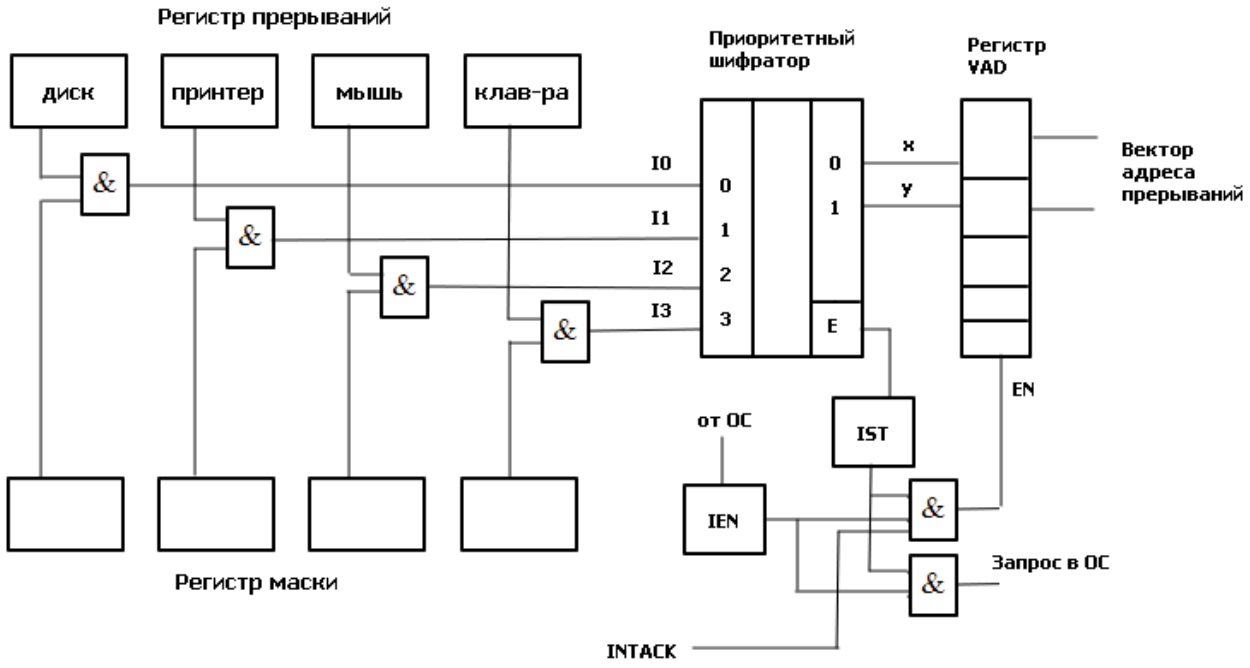

Рис.1. Схема обработки требования прерывания от внешнего устройства. IEN – Interrupt Enable – прерывание разрешено; IST – Interrupt Status – статус прерывания; INTACK – подтверждение прерывания; VAD – вектор адреса прерывания.

В регистре прерываний устройства подключены в соответствии с их приоритетами. Чем выше приоритет, тем на меньший по значению вход приоритетного шифратора он поступает. Поэтому при одновременном поступлении требований прерывания от нескольких внешних устройств будет обрабатываться требование от устройства с высшим приоритетом. В начале оперативной памяти расположена таблица векторов прерываний. В каждой ячейке этой таблицы расположена команда безусловного перехода на начальный адрес соответствующей устройству программы обслуживания прерывания (ПОП).

Рассмотрим пример вычислительного процесса при обработке прерываний от внешних устройств. Пусть программы обслуживания прерываний находятся по адресам 101-200 для диска (нулевой приоритет), 201-300 для принтера (1-й приоритет), 301-400 для мыши (2-й приоритет) и 401-500 для клавиатуры (3-й приоритет). Исполняемый код программы находится в области памяти 701-1500. Для сохранения адресов возврата предусмотрен стек.

Пусть при выполнении команды основной программы по адресу 809 пришло прерывание от клавиатуры. Затем, при выполнении команды по адресу 442 ПОП клавиатуры, пришло прерывание от принтера.

Ход вычислительного процесса в этих условиях может быть изображен в виде схемы (рис. 2).

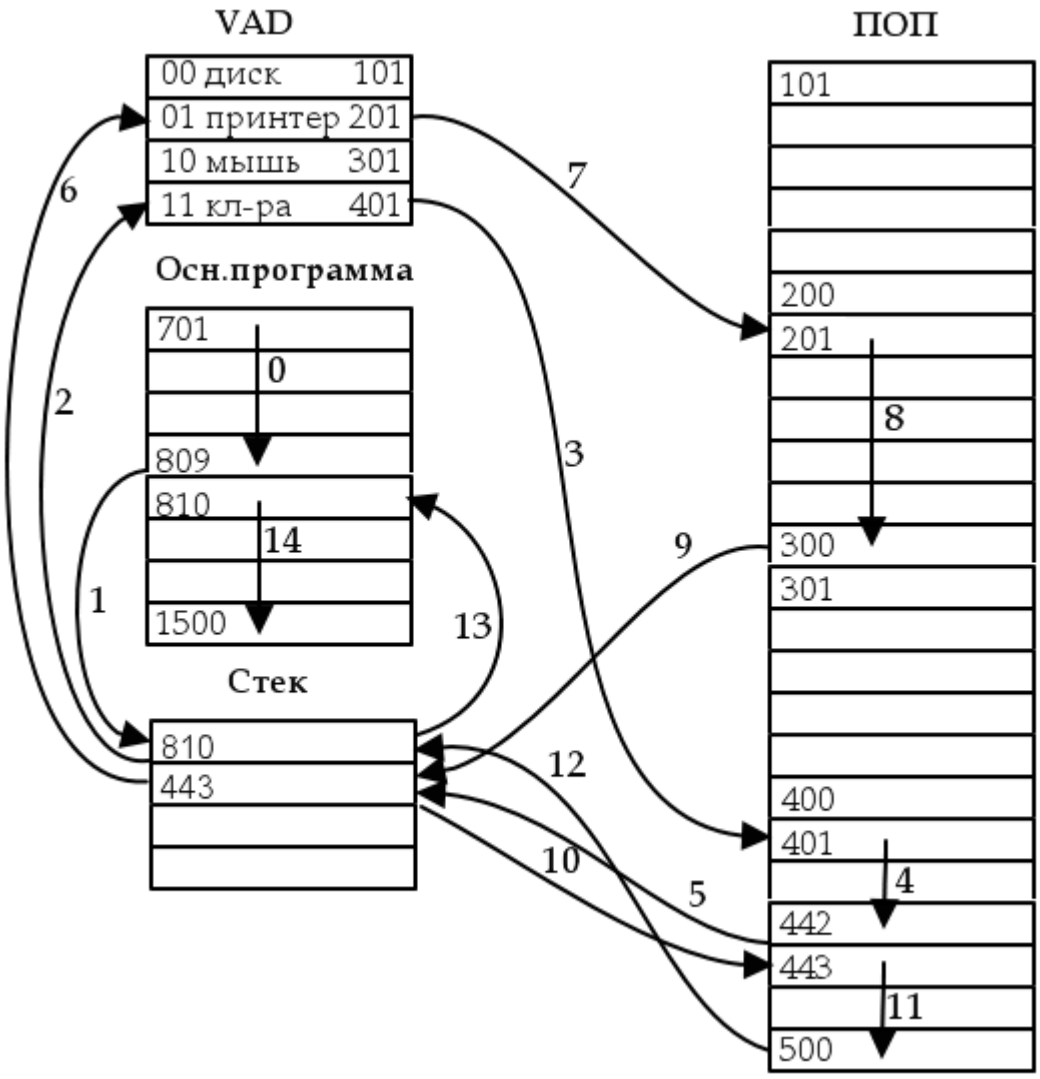

Рис.2. Последовательность передачи управления вычислительным процессом при обработке прерываний от внешних устройств. VAD – область памяти с векторами обработки прерываний.

Задание

К компьютеру подключены 4 периферийных устройства (ПУ) с номерами 0, 1, 2 и 3 (в порядке убывание их приоритета). Вектор-адреса (VAD) этих устройств располагаются в оперативной памяти (ОП) по адресам 00, 01, 10, 11, а соответствующие драйвера, т.е. программы обслуживания прерываний (ПОП), находятся по адресам 101-200, 201-300, 301-400 и 401-500.

Объектный код прикладной программы, исполнение которой прерывается при обращениях к ПУ, хранятся в ОП в области адресов 701-1500.

Для сохранения адресов возврата при прерываниях предусмотрен стек.

В соответствии с заданием, где конкретизированы название ПУ и приведен порядок прерываний прикладной программы от 2-х из перечисленных ПУ, при-

вести поясняющий рисунок, где следует схематически изобразить стек, оперативную память с перечисленными адресами для VAD, для программ обслуживания прерываний и прикладной программы, а также указать с помощью стрелок порядок выполнения вычислительного процесса в ходе прерываний. Чтобы порядок выполнения был более ясным, возле стрелок следует поставить последовательные номера: 0,1,2,3 и т.д.

Варианты заданий:

Вариант №1

ПУ: 0. Диск 1. Принтер 2. Манипулятор 3. Клавиатура.

При выполнении команды 742 прикладной программы пришло прерывание от манипулятора. Далее, при выполнении команды 335 ПОП манипулятора пришло прерывание от диска.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №2

ПУ: 0. Диск 1. Принтер 2. Манипулятор 3. Клавиатура.

При выполнении команды 625 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 336 ПОП клавиатуры пришло прерывание от манипулятора.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №3

ПУ: 0. Диск 1. Принтер 2. Манипулятор 3. Клавиатура.

При выполнении команды 882 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 456 ПОП клавиатуры пришло прерывание от принтера.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №4

ПУ: 0. Диск 1. Принтер 2. Манипулятор 3. Тачпад.

При выполнении команды 805 прикладной программы пришло прерывание от манипулятора. Далее, при выполнении команды 380 ПОП манипулятора пришло прерывание от диска.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №5

ПУ: 0. Диск 1. Плоттер 2. Манипулятор 3. Клавиатура.

При выполнении команды 909 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 445 ПОП клавиатуры пришло прерывание от манипулятора.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №6

ПУ: 0. Диск 1. Принтер 2. Тачпад 3. Клавиатура.

При выполнении команды 1201 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 423 ПОП клавиатуры пришло прерывание от принтера.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №7

ПУ: 0. Диск 1. Плоттер 2. Манипулятор 3. Клавиатура.

При выполнении команды 1102 прикладной программы пришло прерывание от манипулятора. Далее, при выполнении команды 309 ПОП манипулятора пришло прерывание от диска.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №8

ПУ: 0. Диск 1. Плоттер 2. Манипулятор 3. Клавиатура.

При выполнении команды 990 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 442 ПОП клавиатуры пришло прерывание от манипулятора.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №9

ПУ: 0. Диск 1. Принтер 2. Тачпад 3. Клавиатура.

При выполнении команды 998 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 467 ПОП клавиатуры пришло прерывание от принтера.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №10

ПУ: 0. Диск 1. Принтер 2. Манипулятор 3. Тачпад.

При выполнении команды 876 прикладной программы пришло прерывание от манипулятора. Далее, при выполнении команды 335 ПОП манипулятора пришло прерывание от диска.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №11

ПУ: 0. Диск 1. Плоттер 2. Манипулятор 3. Клавиатура.

При выполнении команды 754 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 487 ПОП клавиатуры пришло прерывание от манипулятора.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №12

ПУ: 0. Диск 1. Принтер 2. Плоттер 3. Клавиатура.

При выполнении команды 1200 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 432 ПОП клавиатуры пришло прерывание от принтера.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №13

ПУ: 0. Диск 1. Принтер 2. Манипулятор 3. Тачпад.

При выполнении команды 845 прикладной программы пришло прерывание от манипулятора. Далее, при выполнении команды 323 ПОП манипулятора пришло прерывание от диска.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №14

ПУ: 0. Диск 1. Плоттер 2. Манипулятор 3. Клавиатура.

При выполнении команды 987 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 432 ПОП клавиатуры пришло прерывание от манипулятора.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №15

ПУ: 0. Диск 1. Принтер 2. Плоттер 3. Клавиатура.

При выполнении команды 945 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 462 ПОП клавиатуры пришло прерывание от принтера.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №16

ПУ: 0. Диск 1. Принтер 2. Манипулятор 3. Клавиатура.

При выполнении команды 888 прикладной программы пришло прерывание от манипулятора. Далее, при выполнении команды 359 ПОП манипулятора пришло прерывание от диска.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №17

ПУ: 0. Диск 1. Принтер 2. Манипулятор 3. Клавиатура.

При выполнении команды 966 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 412 ПОП клавиатуры пришло прерывание от манипулятора.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №18

ПУ: 0. Диск 1. Принтер 2. Манипулятор 3. Клавиатура.

При выполнении команды 1309 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 488 ПОП клавиатуры пришло прерывание от принтера.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №19

ПУ: 0. Диск 1. Принтер 2. Манипулятор 3. Клавиатура.

При выполнении команды 789 прикладной программы пришло прерывание от манипулятора. Далее, при выполнении команды 376 ПОП манипулятора пришло прерывание от диска.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №20

ПУ: 0. Диск 1. Принтер 2. Манипулятор 3. Клавиатура.

При выполнении команды 952 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 466 ПОП клавиатуры пришло прерывание от манипулятора.

Изобразить на рисунке ход выполнения вычислительного процесса.

Вариант №21

ПУ: 0. Диск 1. Принтер 2. Плоттер 3. Клавиатура.

При выполнении команды 977 прикладной программы пришло прерывание от клавиатуры. Далее, при выполнении команды 489 ПОП клавиатуры пришло прерывание от принтера.

Изобразить на рисунке ход выполнения вычислительного процесса.

Выводы

Сделать выводы по проделанной работе.

Контрольная работа №6
Структура и характеристики памяти

Цель контрольной работы

Целью занятия является проверка усвоения материалов лекции по устройству и характеристикам памяти ВС.

Стековая память

Структура стековой памяти (стека) и логика управления ею представлены на рис. 3.5. Стековая память состоит из ячеек, причем обмен информацией между остальными устройствами ЭВМ и стеком всегда выполняется только через верхнюю ячейку – вершину стека. При записи нового слова (команды, числа, символа и т.п.) все ранее записанные слова сдвигаются на одну ячейку вниз, а новое слово помещается в вершину стека. Считывание возможно только с вершины стека и производится с удалением или без удаления считываемого слова. Такую память часто называют памятью типа LIFO (Last – In First – Out – последним вошел, первым вышел).

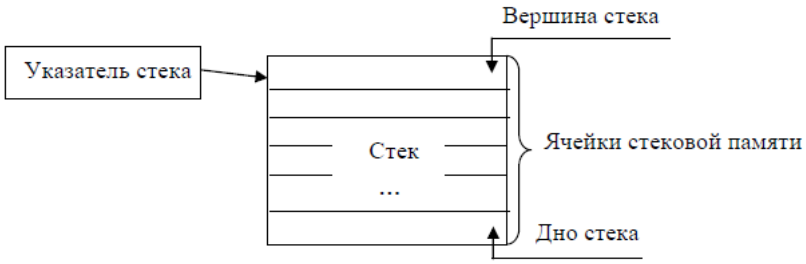

Рис.3.5. Структура стека (указатель стека - регистр для указания адреса вершины стека).

Принципы функционирования стека проиллюстрированы на рис. 3.6.

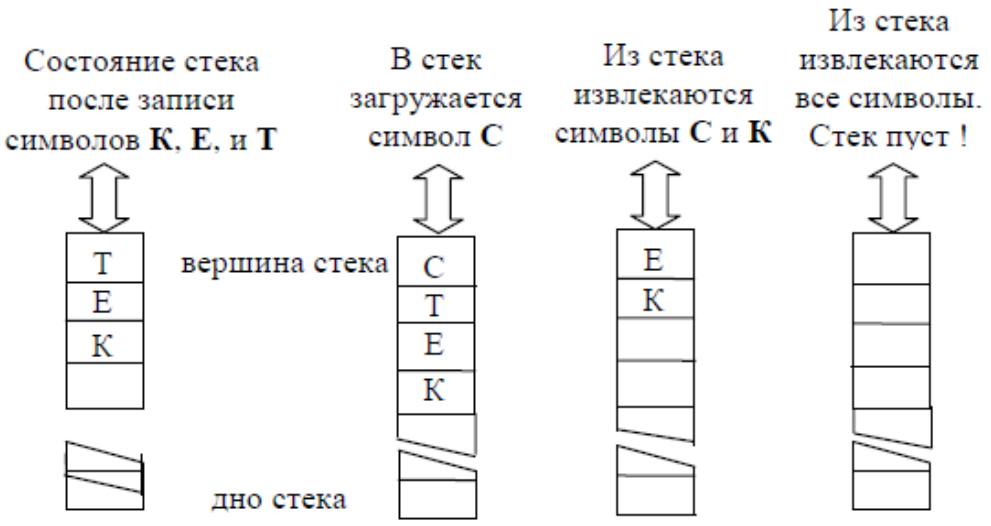

Рис.3.6. Пример работы со стеком

Очевидно, что аппаратная реализация стека, как правило, не целесообразна, поскольку при выполнении каждой операции содержимое стека перемещается целиком, поэтому в ЭВМ стек моделируется, и в качестве стековой памяти используется просто часть адресной памяти, что позволяет легко менять содержимое стека (рис. 3.7).

Несмотря на то, что внешне модель стека отличается от аппаратного стека, она функционирует по тому же алгоритму: последним вошел - первым вышел. Это обеспечивается с помощью указателя стека, который содержит адрес плавающей вершины стека. Заметим, что моделируемый стек растет не сверху вниз (ко дну стека), а снизу вверх (от дна стека). Однако это не нарушает алгоритма функционирования такой памяти. На рис. 3.7. показан стек, организованный в памяти с 32-разрядными словами. Предполагается, что дно стека располагается в ячейке с 8-ричным адресом 1000.

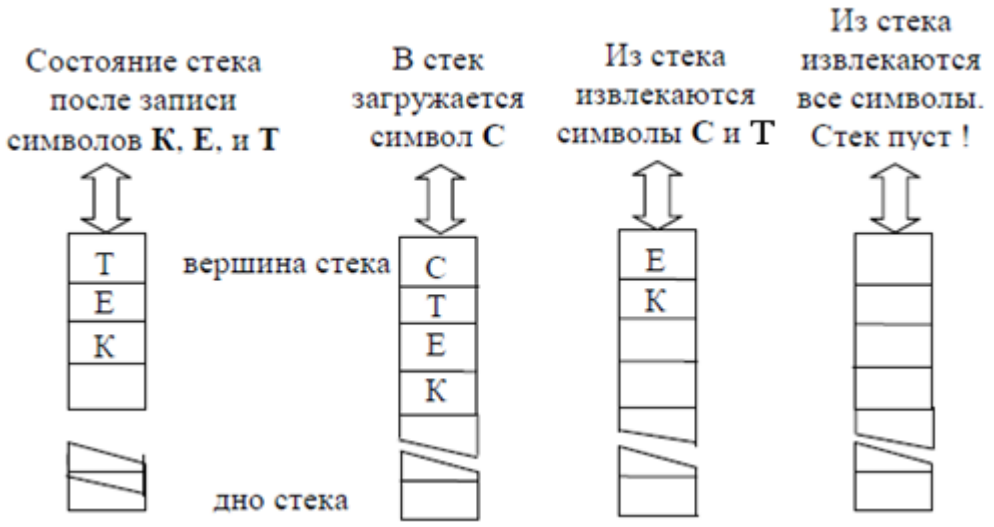

Рис. 3.7. Модель стека, реализованная в оперативной памяти

При работе со стеком, представленным на рис. 3.7, следует учесть, что при занесении данного в стек сначала уменьшается значение указателя стека, то есть происходит его настройка на новую вершину, а затем уже производится загрузка в стек; при считывании из стека после извлечения данного из вершины указатель стека увеличивает свое значение. Именно поэтому, когда стек пуст, значение указателя устанавливается на ячейке, следующей за дном стека.

В качестве демонстрации приведем пример использования стека для вычисления арифметических выражений с использованием Польской Инверсной (бесскобочной) Записи (ПОЛИЗ). В ПОЛИЗе операция записывается не между операндами (Х+У), а после них (ХУ+).

Подобные выражения можно вычислять по следующему алгоритму:

1) проанализировать каждый символ бесскобочной записи формулы, начиная с крайнего левого символа, до тех пор, пока не встретится знак операции;

2) взять ближайших два операнда (если операция двуместная) или один операнд (при унарной операции), расположенные слева от обнаруженного знака операции, выполнить операцию и результат поместить в формулу на место выделенных операндов и знака операции;

3) если после выполнения пункта 2) формула состоит из одного значения, это значение и есть результат, то есть алгоритм завершен, в противном случае перейти к пункту 1).

Пример использования такого алгоритма для вычисления по формуле

Y = (A+B*C)/(E−D),

преобразованной в формулу

Y=ABC*+ED−/

приведен в нижеследующей таблице.

Для определенности в ней выбраны следующие числовые значения операндов: A=8, B=2, C=5, E=6 и D=4.

Шаг	Формула, подлежащая расчету	Левый знак операции	Операнды	Результат	Новая формула после выполнения операции
1	8 2 5 * + 6 4 − /	*	2 *и* 5	10	8 10 + 6 4 − /
2	8 10 + 6 4 − /	+	8 *и* 10	18	18 6 4 − /
3	18 6 4 − /	−	6 *и* 4	2	18 2 /
4	18 2 /	/	18 *и* 2	9	9

На рис. 3.8. изображен расчет выражения (*) с использованием стека.

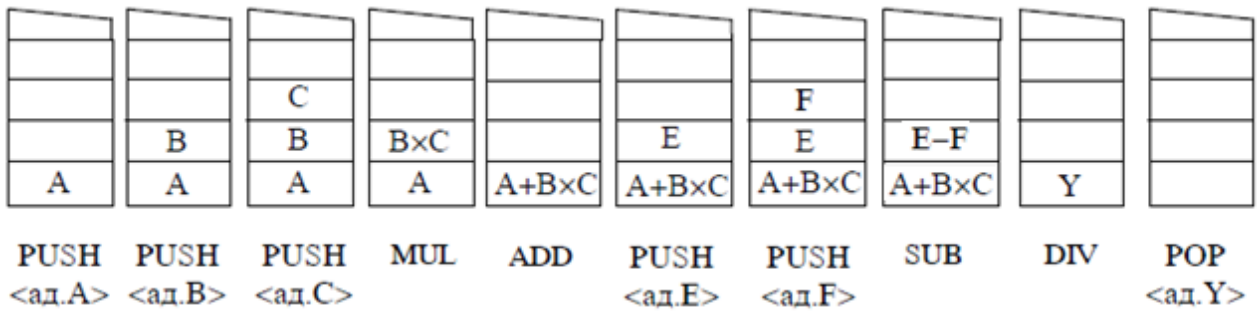

Рис. 3.8. Пример использования стека для вычисления выражений.

Предполагается, что при работе со стеком на рис. 3.8. используются сле-

дующие команды:

PUSH <ад.> - загрузка в стек содержимого ячейки с адресом <ад.>, POP <ад.> - выгрузка из стека в память по адресу <ад.>, MUL - умножение содержимого двух верхних ячеек стека, ADD - сложение содержимого двух верхних ячеек стека, SUB - вычитание содержимого двух верхних ячеек стека, DIV - деление содержимого двух верхних ячеек стека.

Контрольные вопросы.

1.Что такое байт?

2.Какова длина байта для разных типов ЭВМ?

3.Что такое длина машинного слова ЭВМ? Чем она определяется?

4. Чем определяется размер физической памяти ЭВМ?

5. Какие операции допустимы при взаимодействии с памятью?

6. Какую функцию выполняет запоминающее устройство?

7. Чем отличаются цикл записи и цикл считывания информации?

8. Куда девается старое содержимое при операции записи в ЗУ?

9. Сформулируйте понятие цикл памяти.

10. Какие типы ЗУ различают по способу выборки информации?

11. Как классифицируют основную память по методу размещения и поиска информации?

12. Как осуществляется доступ к информации в ассоциативной памяти?

13. Как осуществляется доступ к информации в стековой памяти?

14. Как осуществляется доступ к информации в адресной памяти?

15. Чем отличаются ЗУ ROM и RAM?

16. Чем отличаются энергозависимые и энергонезависимые ЗУ?

17. Как классифицируются ЗУ по своему функциональному назначению?

18. Какие ЗУ обычно невелики по объему?

19. Из чего состоит основная память ЭВМ?

20. Что такое сверхоперативная память?

21. Что относится к классу внешних ЗУ?

22. Назовите несколько типов ЗУ, входящих в состав основной памяти

ЭВМ.

23. Из каких элементов состоит оперативная память?

23. Объясните последовательность функционирования ОЗУ в цикле считывания информации.

24. Объясните последовательность функционирования ОЗУ в цикле записи информации.

25. Что такое логическая структура оперативной памяти?

26. Из каких элементов состоит постоянная память?

27. Каков порядок доступа к информации в стековой памяти?

28. Что такое ПОЛИЗ?

29. Каков алгоритм формирования арифметического выражения в ПОЛИЗе?

30. В чем состоит проблема согласования пропускной способности процессора и памяти.

31. Что такое кэш-память?

32. Сколько и какие регистры входят в состав устройства управления ассоциативной памяти, каков размер этих регистров?

33. Что называется маской в АЗУ?

34. Что такое контекст в АЗУ?

35. Как осуществляется ассоциативный поиск информации?

36. Важнейшие свойства АЗУ?

37. Почему объем АЗУ обычно невелик?

38. На чем основан эффект от использования кэш-памяти?

39. Для чего используется кэш-память?

40. Какова структура и принцип функционирования кэш-памяти?

41. Что такое пространственная локальность данных?

42. Что такое временная локальность данных?

43. Какие архитектуры кэш-памяти различают по способу определения соответствия строки кэш-памяти и области ОП?

44. Чем характеризуется кэш-память прямого отображения?

45. Что такое полностью ассоциативный кэш?

46. Что такое частично ассоциативный кэш?

47. Что такое политика замещения строк в кэш-памяти?

48. Что такое эффективность кэша?

49. Что означает эффективность кэш равная 1?

Задание

1. Ответить письменно на 3 контрольных вопроса с номерами k, k+10, k+20, где k – номер студента по списку группы.
2. Решить задачу с номером k, где k – номер студента по списку группы.

Варианты задач:

Вариант №1

Преобразовать в ПОЛИЗ арифметическое выражение Q=(A-B)*C/(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=2, B=4, C=1, D=2, E=3.

Вариант №2

Преобразовать в ПОЛИЗ арифметическое выражение Q=(A-B)*C/(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=3, B=5, C=3, D=4, E=1.

Вариант №3

Преобразовать в ПОЛИЗ арифметическое выражение Q=(A+B)*C/(D+E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=1, B=4, C=4, D=1, E=4.

Вариант №4

Преобразовать в ПОЛИЗ арифметическое выражение Q=A*(B+C)/(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=5, B=6, C=2, D=5, E=1.

Вариант №5

Преобразовать в ПОЛИЗ арифметическое выражение Q=A*(B+C)/(D+E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=6, B=5, C=3, D=1, E=3.

Вариант №6

Преобразовать в ПОЛИЗ арифметическое выражение Q=A*(B-C)/(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=7, B=4, C=2, D=5, E=3.

Вариант №7

Преобразовать в ПОЛИЗ арифметическое выражение Q=A*(B+C)*(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=4, B=2, C=3, D=8, E=4.

Вариант №8

Преобразовать в ПОЛИЗ арифметическое выражение Q=A/(B+C)*(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=8, B=2, C=2, D=5, E=1.

Вариант №9

Преобразовать в ПОЛИЗ арифметическое выражение Q=A*(B-C)/(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=9, B=1, C=5, D=5, E=4.

Вариант №10

Преобразовать в ПОЛИЗ арифметическое выражение Q=A/(B+C)/(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=12, B=1, C=3, D=5, E=2.

Вариант №11

Преобразовать в ПОЛИЗ арифметическое выражение Q=A*(B+C)*(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=2, B=3, C=5, D=1, E=4.

Вариант №12

Преобразовать в ПОЛИЗ арифметическое выражение Q=A*(B-C)/(D+E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=5, B=4, C=2, D=2, E=3.

Вариант №13

Преобразовать в ПОЛИЗ арифметическое выражение Q=A/(B-C)*(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=8, B=1, C=3, D=3, E=4.

Вариант №14

Преобразовать в ПОЛИЗ арифметическое выражение Q=(A+B)*C/(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=4, B=4, C=2, D=5, E=4.

Вариант №15

Преобразовать в ПОЛИЗ арифметическое выражение Q=A/(B-C)*(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=5, B=4, C=1, D=5, E=2.

Вариант №16

Преобразовать в ПОЛИЗ арифметическое выражение Q=A/(B+C)*(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=9, B=2, C=1, D=7, E=4.

Вариант №17

Преобразовать в ПОЛИЗ арифметическое выражение Q=A/(B-C)*(D+E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=2, B=7, C=3, D=5, E=4.

Вариант №18

Преобразовать в ПОЛИЗ арифметическое выражение Q=A*(B-C)*(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=2, B=4, C=3, D=7, E=4.

Вариант №19

Преобразовать в ПОЛИЗ арифметическое выражение Q=A/(B*C)+(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=12, B=4, C=3, D=5, E=4.

Вариант №20

Преобразовать в ПОЛИЗ арифметическое выражение Q=A-(B*C)*(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=2, B=7, C=2, D=5, E=4.

Вариант №21

Преобразовать в ПОЛИЗ арифметическое выражение Q=A/(B/C)+(D-E).

Изобразить на рисунке ход вычислительного процесса в стековой ВМ для данного примера, а также в виде таблицы последовательность выполнения арифметических операций при значениях A=4, B=4, C=2, D=5, E=2.

Выводы

Сделать выводы по проделанной работе.

Контрольная работа №7
Конвейеры

Цель контрольной работы

Целью занятия является проверка усвоения материалов лекции по конвейерной обработке в ВС.

Показатели эффективности конвейеров

Для расчета эффективности за счет конвейеризации вычислений обычно используют показатели: ускорение, эффективность и производительность.

Под ускорением понимается отношение времени обработки без конвейера и времени обработки при наличии конвейера. Теоретически наилучшее время обработки массива значений T_{NK} на конвейере с ступенями и тактовым периодом T_K можно определить выражением

$$\boldsymbol{T_{NK} = \big(K + (N - 1)\big) * T_K}.$$

Здесь учтено, что первый результат будет получен за K тактов, а последующие будут получены в каждом такте по одному. В процессоре без конвейера время обработки составит $N*K*T_K$. Следовательно, ускорение обработки составит

$$\boldsymbol{S = \frac{N\,K\,T_K}{(K+(N-1))\,T_K} = \frac{NK}{K+(N-1)}}.$$

При $N\to\infty$ ускорение стремится к величине K - числу ступеней конвейера. Следующий параметр – эффективность E – доля ускорения, приходящаяся на одну ступень конвейера.

$$\boldsymbol{E = \frac{S}{K} = \frac{N}{K+(N-1)}}.$$

При $N\to\infty$ E стремится к 1.

Третьим параметром выступает пропускная способность или производительность P – эффективность, деленная на длительность тактового периода:

$$\boldsymbol{P = \frac{N}{T_K\,((K+(N-1))}}.$$

При $N\to\infty$ P стремится к частоте тактирования конвейера $F = 1/ T_K$.

Существуют три класса конфликтов:

- Структурные конфликты, которые возникают из-за конфликтов по ресурсам, когда аппаратные средства не могут поддерживать все возможные комбинации команд в режиме одновременного выполнения с совмещением.
- Конфликты по данным, возникающие в случае, когда выполнение одной команды зависит от результата выполнения предыдущей команды.

- Конфликты по управлению, которые возникают при конвейеризации команд переходов и других команд, которые изменяют значение счетчика команд.

Задание.

Решить 2 задачи: с номером, соответствующим последней цифре номера студента по списку группы (0 – решает №10) и с номером на 10 больше. То есть 1 и 11, 2 и 12, …, 10 и 20.

Варианты задач:

Вариант №1

Конвейерная обработка в процессоре составляет 5 этапов, каждый длительностью в 1 такт. Определите среднюю длительность выполнения одной команды при количестве команд 8 и отсутствии конфликтов в конвейере.

Вариант №2

Конвейерная обработка в процессоре составляет 5 этапов, каждый длительностью в 1 такт. Определите среднюю длительность выполнения одной команды при количестве команд 12 и отсутствии конфликтов в конвейере.

Вариант №3

Конвейерная обработка в процессоре составляет 6 этапов, каждый длительностью в 1 такт. Определите среднюю длительность выполнения одной команды при количестве команд 10 и отсутствии конфликтов в конвейере.

Вариант №4

Конвейерная обработка в процессоре составляет 6 этапов, каждый длительностью в 1 такт. Определите среднюю длительность выполнения одной команды при количестве команд 12 и отсутствии конфликтов в конвейере.

Вариант №5

Конвейерная обработка в процессоре составляет 7 этапов, каждый длительностью в 1 такт. Определите среднюю длительность выполнения одной команды при количестве команд 12 и отсутствии конфликтов в конвейере.

Вариант №6

Конвейерная обработка в процессоре составляет 7 этапов, каждый длительностью в 1 такт. Определите среднюю длительность выполнения одной команды при количестве команд 16 и отсутствии конфликтов в конвейере.

Вариант №7

Конвейерная обработка в процессоре составляет 8 этапов, каждый длительностью в 1 такт. Определите среднюю длительность выполнения одной команды при количестве команд 10 и отсутствии конфликтов в конвейере.

Вариант №8

Конвейерная обработка в процессоре составляет 8 этапов, каждый длительностью в 1 такт. Определите среднюю длительность выполнения одной команды при количестве команд 12 и отсутствии конфликтов в конвейере.

Вариант №9

Конвейерная обработка в процессоре составляет 9 этапов, каждый длительностью в 1 такт. Определите среднюю длительность выполнения одной команды при количестве команд 10 и отсутствии конфликтов в конвейере.

Вариант №10

Конвейерная обработка в процессоре составляет 9 этапов, каждый длительностью в 1 такт. Определите среднюю длительность выполнения одной команды при количестве команд 12 и отсутствии конфликтов в конвейере.

Вариант №11

Конвейерная обработка в процессоре составляет 9 этапов, каждый длительностью в 1 такт, кроме 5-го этапа, который составляет 2 такта. Определите среднюю длительность выполнения одной команды при количестве команд 10 и отсутствии конфликтов в конвейере.

Вариант №12

Конвейерная обработка в процессоре составляет 9 этапов, каждый длительностью в 1 такт, кроме 8-го этапа, который составляет 2 такта. Определите среднюю длительность выполнения одной команды при количестве команд 12 и отсутствии конфликтов в конвейере.

Вариант №13

Конвейерная обработка в процессоре составляет 8 этапов, каждый длительностью в 1 такт, кроме 4-го этапа, который составляет 2 такта. Определите среднюю длительность выполнения одной команды при количестве команд 10 и отсутствии конфликтов в конвейере

Вариант №14

Конвейерная обработка в процессоре составляет 8 этапов, каждый длительностью в 1 такт, кроме 2-го этапа, который составляет 2 такта. Определите среднюю длительность выполнения одной команды при количестве команд 6 и отсутствии конфликтов в конвейере

Вариант №15

Конвейерная обработка в процессоре составляет 7 этапов, каждый длительностью в 1 такт, кроме 4-го этапа, который составляет 2 такта. Определите среднюю длительность выполнения одной команды при количестве команд 6 и отсутствии конфликтов в конвейере

Вариант №16

Конвейерная обработка в процессоре составляет 7 этапов, каждый длительностью в 1 такт, кроме 5-го этапа, который составляет 2 такта. Определите среднюю длительность выполнения одной команды при количестве команд 8 и отсутствии конфликтов в конвейере

Вариант №17

Конвейерная обработка в процессоре составляет 6 этапов, каждый длительностью в 1 такт, кроме 4-го этапа, который составляет 2 такта. Определите среднюю длительность выполнения одной команды при количестве команд 10 и отсутствии конфликтов в конвейере

Вариант №18

Конвейерная обработка в процессоре составляет 6 этапов, каждый длительностью в 1 такт, кроме 3-го этапа, который составляет 2 такта. Определите среднюю длительность выполнения одной команды при количестве команд 8 и отсутствии конфликтов в конвейере

Вариант №19

Конвейерная обработка в процессоре составляет 5 этапов, каждый длительностью в 1 такт, кроме 4-го этапа, который составляет 2 такта. Определите среднюю длительность выполнения одной команды при количестве команд 12 и отсутствии конфликтов в конвейере

Вариант №20

Конвейерная обработка в процессоре составляет 5 этапов, каждый длительностью в 1 такт, кроме 3-го этапа, который составляет 2 такта. Определите среднюю длительность выполнения одной команды при количестве команд 10 и отсутствии конфликтов в конвейере.

Выводы

Сделать выводы по проделанной работе.

Контрольная работа №8
Параллельные вычисления

Цель контрольной работы

Закрепить знания учащихся по вопросам оценивания эффективности параллельных вычислений. Выработать навыки оценки метрик эффективности для заданного набора параметров системы.

Метрики параллельных вычислений

Современные требования увеличения производительности ВМ и систем обусловливают применение параллельных вычислений как в рамках одной ЭВМ, так и путем создания многопроцессорных систем и сетей, объединяющих большое количество отдельных процессоров или вычислительных машин. Для такого подхода следует применять термин вычислительная система (ВС) вместо термина ВМ. Общей отличительной особенностью ВС является наличие средств, реализующих параллельную обработку, за счет введения параллельных ветвей в вычислениях, что не предусматривалось классической архитектурой ЭВМ.

В силу особенностей параллельных вычислений для оценки их эффективности используют специфическую систему метрик. Наиболее распространенными из таких метрик являются следующие:

- профиль параллелизма программы;
- индекс параллелизма;
- ускорение;
- эффективность;
- утилизация;
- избыточность;
- сжатие;
- качество.

Для вывода соотношений, определяющих значение метрик, будем использовать следующие характеристики:

- n – количество процессоров в ВС;
- O(n) – общее число операций (команд), выполненных на n-процессорной системе;
- T(n) – время выполнения O(n) операций на n-процессорной системе в виде числа квантов времени.

Примем допущения, что время изменяется дискретно, за 1 квант времени процессор выполняет любую операцию. Поэтому справедливы следующие со-

отношения для времени и объема вычислений: T(1)=O(1), T(n)≤O(n). Последнее соотношение формулирует утверждение: время вычислений можно сократить за счет распределения объема вычислений по нескольким процессорам.

Профиль параллелизма программы

Число процессоров многопроцессорной системы p, параллельно участвующих в выполнении программы в каждый момент времени t, определяют понятием степень параллелизма p(t). Графическое представление параметра p(t) как функции времени называют профилем параллелизма программы. Изменения в уровне загрузки процессоров за время наблюдения зависят от многих факторов (алгоритма, доступных ресурсов, степени оптимизации, обеспечиваемой компилятором и т. д.). Типичный профиль параллелизма для алгоритма декомпозиции показан на рис. 7.1.

В дальнейшем будем исходить из следующих предположений: система состоит из п гомогенных процессоров; максимальный параллелизм в профиле равен т и, в идеальном случае, т стремится к п. Производительность Δ одиночного процессора системы выражается в единицах скорости вычислений (количество операций в единицу времени) и не учитывает издержек, связанных с обращением к памяти и пересылкой данных. Если за наблюдаемый период загружены i процессоров, то p=i.

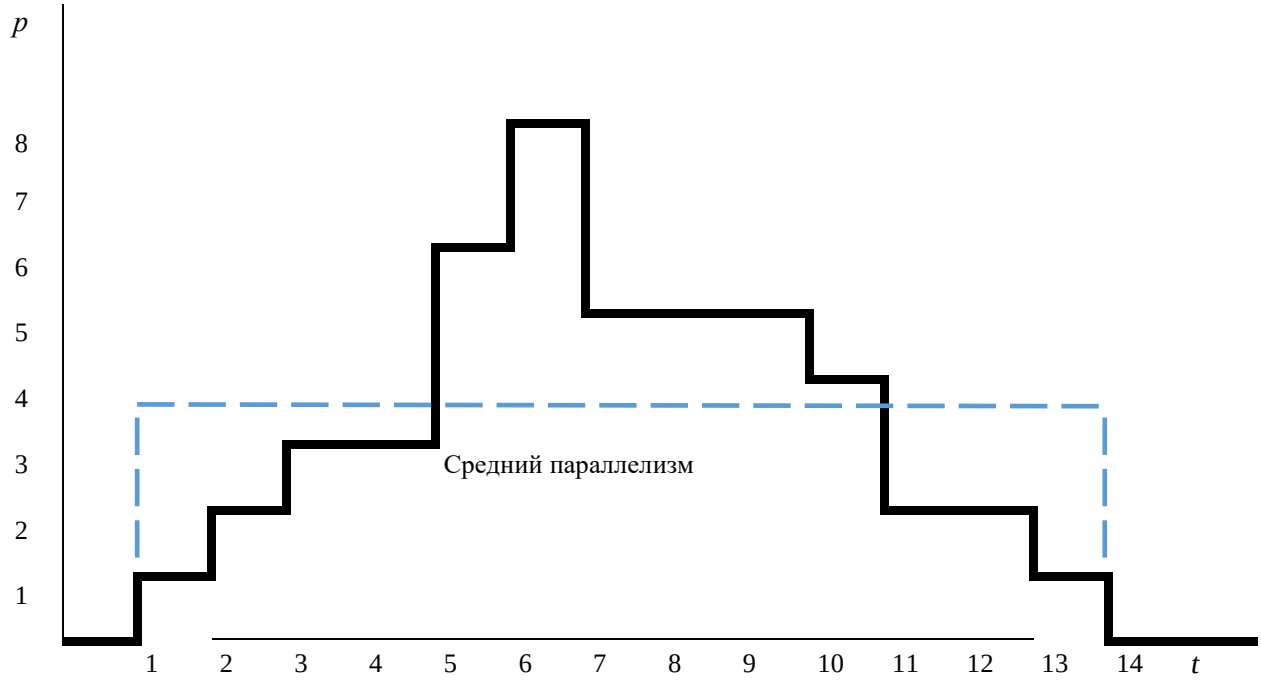

Рис. 7.1. Профиль параллелизма.

Общий объем вычислений O(n) за период от стартового времени tн до момента завершения $t_к$ пропорционален площади под кривой профиля паралле-

лизма:

$$O(n) \quad = \Delta \sum_{i=1}^{n} \mathrm{it}_i$$

Где ti – интервал времени (общее количество квантов времени) в течение которого p=i, а $T = \sum_{i=1}^{n} \mathrm{t}_i = t_{\text{к}} - t_{\text{н}}$ – общее время вычислений. Средний параллелизм определяется как O(n)/T.

Метрики, характеризующие скорость вычислений

К таким метрикам относятся индекс параллелизма и ускорение.

Индекс параллелизма характеризует среднюю скорость параллельных вычислений через количество выполненных операций:

PI(n) = O(n)/T(n).

Ускорение за счет параллельного выполнения программы служит показателем эффективной скорости вычислений. Ускорение определяется как отношение времени, затрачиваемого на проведение вычислений на однопроцессорной машине (в варианте наилучшего последовательного алгоритма), к времени решения той же задачи на параллельной n-процессорной ВС (при использовании наилучшего параллельного алгоритма):

S(n) = T(1) / T(n).

Примечания по поводу алгоритмов отражают тот факт, что время выполнения вычислений существенно зависит от алгоритмов, реализуемых программой, и ускорение вычисляется при условии реализации наилучших алгоритмов.

Метрики, характеризующие эффективность вычислений во времени

Эти метрики позволяют судить об эффективности привлечения к решению задачи дополнительных процессоров. К ним относятся метрики эффективность и утилизация;

Эффективность характеризует целесообразность наращивания числа процессоров через ту долю ускорения, достигнутого за счет параллельных вычислений, которая приходится на один процессор:

E(N)=S(n) / n = T(1) / nT(n)

Утилизация учитывает вклад каждого процессора при параллельных вычислениях, но в виде количества операций, приходящихся на один процессор в единицу времени.

$$U(n) = O(n) / N\,T(n)$$

Метрики, характеризующие эффективность вычислений по объему

Эти метрики сравнивают объемы вычислений, выполненные при параллельном и последовательном выполнении задач. К ним относятся избыточность и сжатие.

Избыточность – это отношение объема параллельных вычислений к объему эквивалентных последовательных вычислений:

$$R(n) = O(n) / O(1)$$

Данная метрика основана на абсолютных показателях, базирующихся на объеме вычислительной работы. Избыточность характеризует степень соответствия между программным и аппаратным параллелизмом.

Кстати, поскольку T(1)=O(1), показатель утилизации может быть определен через метрики избыточности и эффективности:

$$U(n) = O(n) / N\,T(n) = R(n)\,E(n).$$

Сжатие вычисляется как величина, обратная избыточности:

$$C(n) = O(1) / O(n)$$

Метрика качества. Эта метрика увязывает между собой метрики ускорения, эффективности и сжатия, является более объективным показателем улучшения производительности за счет параллельных вычислений.

$$Q(n) = \frac{T^3(1)}{nT^2(n)O(n)} = S(n)E(n)C(n).$$

Поскольку как эффективность, так и величина, обратная избыточности, представляют собой дроби, то Q(n) < =S(n). То есть качество Q(n) при любых условиях ограничено сверху величиной ускорения S(n).

Закономерности параллельных вычислений

В идеальном случае система из n процессоров могла бы ускорить вычисления в n раз. В реальности достичь такого показателя по ряду причин не удается. Главная из этих причин заключается в невозможности полного распараллеливания ни одной из задач. Как правило, в каждой программе имеется фрагмент кода, который принципиально должен выполняться последовательно и только одним из процессоров. Это может быть часть программы, отвечающая

за запуск задачи и распределение распараллеленного кода по процессорам, либо фрагмент программы, обеспечивающий операции ввода/вывода. Можно привести и другие примеры, но главное состоит в том, что о полном распараллеливании задачи говорить не приходится.

Известные проблемы возникают и с той частью задачи, которая поддается распараллеливанию. Здесь идеальным был бы вариант, когда параллельные ветви программы постоянно загружали бы все процессоры системы, причем так, чтобы нагрузка на каждый процессор была одинакова. К сожалению, оба этих условия на практике трудно реализуемы. Таким образом, ориентируясь на параллельную ВС, необходимо четко сознавать, что добиться прямо пропорционального числу процессоров увеличения производительности не удастся, и, естественно, встает вопрос о том, на какое реальное ускорение можно рассчитывать.

Ответ зависит от того, каким образом пользователь собирается использовать вычислительные мощности ВС, возросшие в результате увеличения числа процессоров. Наиболее характерны три варианта:

- Объем вычислений не изменяется, главная цель – сократить время вычислений. Достижимое в этом случае ускорение определяется законом Амдала.
- Время вычислений с расширением системы не меняется, но при этом увеличивается объем решаемой задачи. Цель такого подхода – за выделенное время выполнить максимальный объем вычислений. В этой ситуации применим закон Густафсона.
- Этот вариант похож на предыдущий с условием, что объем решаемой задачи ограничен емкостью доступной памяти. Ускорение в этом случае может быть определено по закону Сана и Ная.

Закон Амдала

Джин Амдал (Gene Amdahl) -- один из разработчиков всемирно известной системы IBM 360, в своей работе, опубликованной в 1967 году, предложил формулу, отражающую зависимость ускорения вычислений, достигаемого на многопроцессорной ВС, от числа процессоров и соотношения между последовательной и параллельной частями программы. Показателем сокращения времени вычислений служит такая метрика, как «ускорение». Напомним, что ускорение S-- это отношение времени Ts, затрачиваемого на проведение вычислений на однопроцессорной ВС (в варианте наилучшего последовательного алгоритма), ко времени Tp решения той же задачи на параллельной системе (при

использовании наилучшего параллельного алгоритма):

$$S(n) = T(1) / T(n).$$

Оговорки относительно алгоритмов решения задачи сделаны, чтобы подчеркнуть тот факт, что для последовательного и параллельного решения лучшими могут оказаться разные реализации, а при оценке ускорения необходимо исходить именно из наилучших алгоритмов.

Проблема рассматривалась Амдалом в следующей постановке. Прежде всего, объем решаемой задачи с изменением числа процессоров, участвующих в ее решении, остается неизменным. Программный код решаемой задачи состоит из двух частей: последовательной и распараллеливаемой. Обозначим долю операций, которые должны выполняться последовательно одним из процессоров, через f, где $0<f<1$ (здесь доля понимается не по числу строк кода, а по числу реально выполняемых операций). Отсюда доля, приходящаяся на распараллеливаемую часть программы, составит 1-f. Крайние случаи в значениях f соответствуют полностью параллельным (f=0) и полностью последовательным (f=1) программам. Распараллеливаемая часть программы равномерно распределяется по всем процессорам.

С учетом приведенной формулировки имеем:

$$T(n) = fT(1) + \frac{(1-f)T(1)}{n}$$

В результате получаем формулу Амдала, выражающую ускорение, которое может быть достигнуто на системе из n процессоров:

$$S(n) = T(1) / T(n) = \frac{T(1)}{fT(1)+\frac{(1-f)T(1)}{n}} = \frac{1}{f+\frac{1-f}{n}} = \frac{n}{1+(n-1)f}.$$

Формула выражает простую и обладающую большой общностью зависимость. Если устремить число процессоров к бесконечности, то в пределе получаем:

$$\lim_{n\to\infty} S = \frac{1}{f}.$$

Это означает, что если в программе 10% последовательных операций (то есть f=0,1), то, сколько бы процессоров ни использовалось, убыстрения работы программы более чем в десять раз никак ни получить, да и то, 10 -- это теоретическая верхняя оценка самого лучшего случая, когда никаких других отрицательных факторов нет.

Следует отметить, что распараллеливание ведет к определенным издержкам, которых нет при последовательном выполнении программы. К ним, например, относятся:

Программные издержки. Даже если последовательные и параллельные алгоритмы выполняют одни и те же вычисления, параллельным алгоритмам присущи добавочные программные издержки — дополнительные индексные вычисления, неизбежно возникающие из-за декомпозиции данных и распределения их по процессорам; различные виды учетных операций, требуемые в параллельных алгоритмах, но отсутствующие в последовательных алгоритмах.

Издержки из-за дисбаланса загрузки процессоров. Между точками синхронизации каждый из процессоров должен быть загружен одинаковым объемом работы, иначе часть процессоров будет ожидать, пока остальные завершат свои операции. Эта ситуация известна как дисбаланс загрузки. Таким образом, ускорение ограничивается наиболее медленным из процессоров.

Коммуникационные издержки. Если принять, что обмен информацией и вычисления могут перекрываться, то любые коммуникации между процессорами снижают ускорение. В плане коммуникационных затрат важен уровень гранулярности, определяющий объем вычислительной работы, выполняемой между коммуникационными фазами алгоритма. Для уменьшения коммуникационных издержек выгоднее, чтобы вычислительные гранулы были достаточно крупными, и доля коммуникаций была меньше.

Закон Густафсона

Решая на вычислительной системе из 1024 процессоров три больших задачи, для которых доля последовательного кода f лежала в пределах от 0,4 до 0,8%, Джон Густафсон из NASA Ames Research получил значения ускорения по сравнению с однопроцессорным вариантом, равные соответственно 1021,1020 и 1016. Согласно закону Амдала для данного числа процессоров и диапазона f, ускорение не должно было превысить величины порядка 201.

Пытаясь объяснить это явление, Густафсон пришел к выводу, что причина кроется в исходной предпосылке, лежащей в основе закона Амдала: увеличение числа процессоров не сопровождается увеличением объема решаемой задачи. Реальное же поведение пользователей существенно отличается от такого представления.

Обычно, получая в свое распоряжение более мощную систему, пользователь не стремится сократить время вычислений, а, сохраняя его практически

неизменным, старается пропорционально мощности ВС увеличить объем решаемой задачи. И тут оказывается, что наращивание общего объема программы касается главным образом распараллеливаемой части программы. Это ведет к сокращению значения f. Примером может служить решение дифференциального уравнения в частных производных. Если доля последовательного кода составляет 10% для 1000 узловых точек, то для 100 000 точек доля последовательного кода снизится до 0,1%. Фактически, оставаясь практически неизменной, последовательная часть в общем объеме увеличенной программы имеет уже меньший удельный вес.

Было отмечено, что в первом приближении объем работы, которая может быть произведена параллельно, возрастает линейно с ростом числа процессоров в системе. Для того чтобы оценить возможность ускорения вычислений, когда объем последних увеличивается с ростом количества процессоров в системе (при постоянстве общего времени вычислений), Густафсон рекомендует использовать выражение, предложенное Е. Барсисом (E. Barsis):

$$S = \frac{T_s}{T_p} = \frac{f \times T_s + n \times (1-f) \times T_s}{f \times T_s + (1-f) \times T_s} = n + (1-n) \times f.$$

Данное выражение известно как закон масштабируемого ускорения или закон Густафсона (иногда его называют также законом Густафсона-Барсиса). В заключение отметим, что закон Густафсона не противоречит закону Амдала. Различие состоит лишь в форме утилизации дополнительной мощности ВС, возникающей при увеличении числа процессоров.

Закон Сана – Ная

В многопроцессорной параллельной ВС каждый процессор обычно имеет независимую локальную память сравнительно небольшой емкости. Общая память ВС образуется объединением локальной памяти каждого процессора ВС. При решении задача разделяется на подзадачи и распределяется на подзадачи и распределяется по множеству процессоров. Подзадача размещается в локальной памяти процессора. Как и в постановке густафсона, увеличение числа процессоров сопровождается возрастанием размера решаемой задачи, но до предела, обусловленного объемом доступной памяти. Иными словами, объем задачи увеличивается так, чтобы каждая подзадача полностью занимала локальную память процессора. Такая постановка задачи лежит в основе закона, сформулированного Ксиан-Х- Саном и Лайонелом Наем и носит название закона ускорения, ограниченного памятью.

Если M – емкость локальной памяти одного процессора, то суммарная память n процессоров будет составлять nM. В формулировке проблемы с ограничением, обусловленным доступной памятью, предполагается, что память каждого процессора задействуется полностью, а рабочая нагрузка на один процессор равна O(1), где O(1) = fO(1)+(1-f)O(1)/ Положим, что при использовании всех n процессоров распараллеливаемая часть задачи может масштабироваться в G(n) раз. В этом случае масштабируемая рабочая нагрузка может быть описана выражением O* = fO+(1-f)G(n)O. Здесь параметр G(n) отражает возрастание рабочей нагрузки с увеличением числа процессоров, а значит, и емкости памяти в n раз. В указанных рамках ускорение описывается выражением:

$$S(n) = \frac{f+(1-f)G(n)}{f+(1-f)\frac{G(n)}{n}}.$$

Нетрудно показать, что полученное выражение представляет собой обобщение законов Амдала и Густафсона.

При G(n) = 1 размер задачи фиксирован, что соответствует постановке Амдала:

$$S(\mathrm{n}) = \frac{\mathrm{f}+(1-\mathrm{f})1}{\mathrm{f}+(1-\mathrm{f})\frac{1}{\mathrm{n}}} = \frac{1}{\mathrm{f}+\frac{1-\mathrm{f}}{\mathrm{n}}}$$

Вариант G(n) = n соответствует случаю, когда с увеличением емкости памяти в n раз рабочая нагрузка возрастает тоже в n раз. Это идентично постановке Густафсона:

$$S(\mathrm{n}) = \frac{\mathrm{f}+(1-\mathrm{f})\mathrm{n}}{\mathrm{f}+(1-\mathrm{f})} = \mathrm{f}+(1-\mathrm{f})\mathrm{n}.$$

В случае, когда вычислительная нагрузка возрастает быстрее, чем требования к памяти (G(n) > n) модель с ограничением памяти дает более оптимистическую оценку ускорения.

Задание (пример)

1) Использование процессоров в 16-процессорной ВС при решении задачи характеризуется таблицей:

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P(t)	1	5	6	8	10	12	14	16	16	16	14	16	14	10	9	8	6	4	3	1

   Определите профиль параллелизма, средний параллелизм, ускорение, эффективность, утилизацию, сжатие и качество параллельных вычислений, если избыточность равна 1,2.

2) Рассчитайте ускорение вычислений S(n) на n-процессорной ВС (n=2, 8, 32, 128, 512) по сравнению с однопроцессорной ВС в соответствии с за-

кономерностями Амдала, Густафсона и Сана-Ная при доле последовательной части программы f=0,1 и G(n)=0,5n. Постройте графики зависимости S(n) от n.

Варианты задач:

Конкретные задания по вариантам (номер варианта соответствует номеру студента в группе по модулю 20) отличаются числовыми значениями величин, используемых в заданиях, и приведены в таблице

Вариант	Избыточность R(n)	Доля последовательной части программы f	Коэффициент G(n)
1	1,1	0,2	0,1n
2	1,25	0,2	0,2n
3	1,3	0,2	0,5n
4	1,4	0,2	0,8n
5	1,5	0,1	0,1n
6	1,6	0,1	0,2n
7	1,15	0,1	0,5n
8	1,2	0,1	0,8n
9	1,35	0,05	0,1n
10	1,45	0,05	0,2n
11	1,55	0,05	0,5n
12	1,65	0,05	0,8n
13	1,7	0,01	0,1n
14	1,75	0,01	0,2n
15	1,8	0,01	0,5n
16	1,85	0,01	0,8n
17	1,9	0,005	0,1n
18	1,95	0,005	0,2n
19	1.05	0,005	0,5n
20	1.15	0,005	0,8n

Контрольная работа №9
IP-адресация в локальных и глобальных компьютерных сетях

Цель контрольной работы

Уяснить порядок адресации в сети интернет, получить практику расчета адресов в глобальной и локальных сетях.

Стек протоколов TCP/IP предназначен для соединения отдельных подсетей, построенных по разным технологиям канального и физического уровней (Ethernet, Token Ring, FDDI, ATM, X.25 и т. д.) в единую составную сеть. Каждая из технологий нижнего уровня предполагает свою схему адресации. Поэтому на межсетевом уровне требуется единый способ адресации, позволяющий уникально идентифицировать каждый узел, входящий в составную сеть. Таким способом в TCP/IP-сетях является IP-адресация.

Узел составной сети, имеющий IP-адрес, называется хост (host). Хороший пример, иллюстрирующий составную сеть, – международная почтовая система адресации. Информация сетевого уровня – это индекс страны, добавленный к адресу письма, написанному на одном из тысяч языков земного шара, например на китайском. И даже если это письмо должно пройти через множество стран, почтовые работники которых не знают китайского, понятный им индекс страны-адресата подскажет, через какие промежуточные страны лучше передать письмо, чтобы оно кратчайшим путем попало в Китай. А уже там работники местных почтовых отделений смогут прочитать точный адрес, указывающий город, улицу, дом и человека, и доставить письмо адресату, так как адрес написан на языке и в форме, принятой в данной стране.

Типы адресов стека TCP/IP

В стеке TCP/IP используются три типа адресов:

– локальные (другое название – аппаратные);

– IP-адреса (сетевые адреса);

– символьные доменные имена.

Локальный адрес – это адрес, присвоенный узлу в соответствии с технологией подсети, входящей в составную сеть. Если подсетью является локальная сеть Ethernet, Token Ring или FDDI, то локальный адрес – это MAC-адрес (MAC address – Media Access Control address).

MAC-адреса назначаются сетевым адаптерам и портам маршрутизаторов производителями оборудования и являются уникальными, так как распределя-

ются централизованно. MAC-адрес имеет размер 6 байт и записывается в шестнадцатеричном виде, например, 00-08-A0-12-5F-72.

IP-адреса (IP address) представляют собой основной тип адресов, на основании которых сетевой уровень передает сообщения, называемые IP- пакетами. Эти адреса состоят из 4 байт, записанных в десятичном виде и разделенных точками, например, 117.52.9.44.

Номер узла в протоколе IP назначается независимо от локального адреса узла. Маршрутизатор по определению входит сразу в несколько сетей. Поэтому каждый порт маршрутизатора имеет собственный IP-адрес. Конечный узел также может входить в несколько IP-сетей. В этом случае компьютер должен иметь несколько IP-адресов, по числу сетевых адаптеров. Таким образом, IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение.

Символьные доменные имена (domain name) служат для удобства представления IP-адресов. Человеку неудобно запоминать числовые IP-адреса, поэтому была разработана специальная служба, DNS (Domain Name System), устанавливающая соответствие между IP-адресами и символьными доменными именами, например, www.rambler.ru.

Структура IP-адреса. IP-адрес представляет собой 32-разрядное двоичное число, разделенное на группы по 8 бит, называемых октетами, например: 00010001 11101111 00101111 01011110.

Обычно IP-адреса записываются в виде четырех десятичных октетов и разделяются точками. Таким образом, приведенный выше IP-адрес можно записать в следующей форме: 17.239.47.94.

Следует заметить, что максимальное значение октета равно 11111111_2 (двоичная система счисления), что соответствует в десятичной системе 255_{10}. Поэтому IP-адреса, в которых хотя бы один октет превышает это число, являются недействительными. Пример: 172.16.123.1 – действительный адрес, 172.16.123.256 – несуществующий адрес, поскольку 256 выходит за пределы допустимого диапазона.

IP-адрес состоит из двух логических частей – номера подсети (ID1 подсети) и номера узла (ID хоста) в этой подсети. При передаче пакета из одной подсети в другую используется ID подсети. Когда пакет попал в подсеть назначения, ID хоста указывает на конкретный узел в рамках этой подсети.

Чтобы записать ID подсети, в поле номера узла в IP-адресе ставят нули. Чтобы записать ID хоста, в поле номера подсети ставят нули. Например, если в IP-адресе 172.16.123.1 первые два байта отводятся под номер подсети, осталь-

ные два байта – под номер узла, то номера записываются следующим образом: ID подсети: 172.16.0.0. ID хоста: 0.0.123.1.

По числу разрядов, отводимых для представления номера узла (или номера подсети), можно определить общее количество узлов (или подсетей) по простому правилу: если число разрядов для представления номера узла равно N, то общее количество узлов равно $2^N – 2$.

Два узла вычитаются вследствие того, что адреса со всеми разрядами, равными нулям или единицам, являются особыми и используются в специальных целях. Например, если под номер узла в некоторой подсети отводится два байта (16 бит), то общее количество узлов в такой подсети равно $2^{16} – 2$ =65534 узла.

Для определения того, какая часть IP-адреса отвечает за ID подсети, а какая за ID хоста, применяются два способа: с помощью классов и с помощью масок. Общее правило: под ID подсети отводятся первые несколько бит IP-адреса, оставшиеся биты обозначают ID хоста.

Классы IP-адресов

Существует пять классов IP-адресов: A, B, C, D и E.

За принадлежность к тому или иному классу отвечают первые биты IP-адреса. Деление сетей на классы описано в RFC 791 (документ описания протокола IP). Целью такого деления являлось создание малого числа больших сетей (класса A), умеренного числа средних сетей (класс B) и большого числа малых сетей (класс C).

Если адрес начинается с 0, то сеть относят к классу A и номер сети занимает один байт, остальные 3 байта интерпретируются как номер узла в сети.

Сети класса A имеют номера в диапазоне от 1 до 126. Сетей класса A немного, зато количество узлов в них может достигать $2^{24} – 2$, то есть 16 777 214 узлов.

Если первые два бита адреса равны 10, то сеть относится к классу B. В сетях класса B под номер сети и под номер узла отводится по 16 бит, то есть по 2 байта. Таким образом, сеть класса B является сетью средних размеров с максимальным числом узлов $2^{16} – 2$, что составляет 65 534 узлов.

Если адрес начинается с последовательности 110, то это сеть класса C. В этом случае под номер сети отводится 24 бита, а под номер узла – 8 бит. Сети этого класса наиболее распространены, число узлов в них ограничено $2^8 – 2$, то есть 254 узлами.

Адрес, начинающийся с 1110, обозначает особый, групповой адрес (multicast). Пакет с таким адресом направляется всем узлам, которым присвоен данный адрес.

Адреса класса E в настоящее время не используются (зарезервированы для будущих применений).

Применение классов удовлетворительно решало задачу деления на подсети в начале развития Интернета. В 90-е годы с увеличением числа подсетей стал ощущаться дефицит IP-адресов. Это связано с неэффективностью распределения при классовой схеме адресации. Например, если организации требуется тысяча IP-адресов, ей выделяется сеть класса B, при этом 64534 адреса не будут использоваться.

Существует два основных способа решения этой проблемы:

1) более эффективная схема деления на подсети с использованием масок (RFC 950);

2) применение протокола IP версии 6 (IPv6).

Использование масок

Маска подсети (subnet mask) – это число, которое используется в паре с IP-адресом; двоичная запись маски содержит единицы в тех разрядах, которые должны в IP-адресе интерпретироваться как номер сети. Для стандартных классов сетей маски имеют следующие значения:

• класс A – 11111111. 00000000. 00000000. 00000000 (255.0.0.0);

• класс B – 11111111. 11111111. 00000000. 00000000 (255.255.0.0);

• класс C – 11111111. 11111111. 11111111. 00000000 (255.255.255.0).

Маска подсети записывается либо в виде, аналогичном записи IP-адреса, например 255.255.255.0, либо совместно с IP-адресом с помощью указания числа единичных разрядов в записи маски, например 192.168.1.1/24, т. е. в маске содержится 24 единицы (255.255.255.0).

При использовании масок можно вообще отказаться от понятия классов.

Пример. Пусть задан IP-адрес 17.239.47.94, маска подсети 255.255.0.0 (другая форма записи: 17.239.47.94/16). Требуется определить ID подсети и ID хоста в обеих схемах адресации.

1) Адресация с использованием классов. Двоичная запись IP-адреса имеет вид: 00010001. 11101111. 00101111. 01011110 Так как первый бит равен нулю, адрес относится к классу A. Следовательно, первый байт отвечает за ID подсети, остальные три байта – за ID хоста: ID подсети: 17.0.0.0. ID хоста: 0.239.47.94.

2) Адресация с использованием масок. Запишем IP-адрес и маску подсети в двоичном виде: IP-address: 17.239.47.94 = 00010001. 11101111. 00101111. 01011110 Subnet mask: 255.255.0.0 = 11111111. 11111111. 00000000. 00000000

Вспомним определение маски подсети: интерпретируем как номер подсети те биты, которые в маске равны 1, т. е. первые два байта. Оставшаяся часть IP-адреса будет номером узла в данной подсети. ID подсети: 17.239.0.0. ID хоста: 0.0.47.94.

Номер подсети можно получить другим способом, применив к IP-адресу и маске операцию логического умножения AND: В масках количество единиц в последовательности, определяющей границу номера сети, не обязательно должно быть кратным 8.

Пример

Задан IP-адрес 192.168.89.16, маска подсети 255.255.192.0 (другая форма записи: 192.168.89.16/18). Требуется определить ID подсети и ID хоста. Воспользуемся операцией AND: Чтобы получить номер узла, нужно в битах, отвечающих за номер подсети, поставить нули: Host ID: Для масок существует важное правило: разрывы в последовательности единиц или нулей недопустимы. Например, не существует маски подсети, имеющей следующий вид: 11111111. 11110111. 00000000. 00001000 (255.247.0.8), так как последовательности единиц и нулей не являются непрерывными.

Особые IP-адреса. Некоторые IP-адреса являются особыми, они не должны применяться для идентификации обычных сетей.

• Если первый октет ID сети начинается со 127, такой адрес считается адресом машины-источника пакета. В этом случае пакет не выходит в сеть, а возвращается на компьютер-отправитель. Такие адреса называются loopback («петля», «замыкание на себя») и используются для проверки функционирования стека TCP/IP.

• Если все биты IP-адреса равны нулю, адрес обозначает узел-отправитель и используется в некоторых сообщениях ICMP.

• Если все биты ID сети равны 1, адрес называется ограниченным широковещательным (limited broadcast), пакеты, направленные по такому адресу рассылаются всем узлам той подсети, в которой находится отправитель пакета.

• Если все биты ID хоста равны 1, адрес называется широковещательным (broadcast), пакеты, имеющие широковещательный адрес, доставляются всем узлам подсети назначения.

• Если все биты ID хоста равны 0, адрес считается идентификатором подсети (subnet ID). Наличие особых IP-адресов объясняет, почему из диапазона

доступных адресов исключаются два адреса – это случаи, когда все биты ID хоста равны 1 или 0. Например, в сети класса C не 256 (2^8), а 254 узлов.

С помощью масок администратор (провайдер сети) может структурировать свою сеть путем выделения нескольких подсетей с меньшим количеством узлов. Например, в сети 160.240.128.0/17 требуется выделить 32 подсети меньшего объема. Для этого администратор сети выделяет требуемое количество бит ($32=2^5$, то есть 5 разрядов) в старших разрядах выделенного ему адресного пространства для обозначения номера подсети (от 0 до 31). Тогда для нумерации узла в подсети остается 32-17-5=10 двоичных разрядов, что соответствует максимальному количеству узлов в каждой подсети 2^{10}-2=1022. В этом случае IP-адрес 160.240.192.177/17 будет принадлежать 177-му узлу в 16-й подсети данного провайдера (сети 160.240.128.0/17).

Маршрутизаторы во внешней сети (Интернете) ничего «не знают» о делении сети 160.240.128.0/17 на подсети, все пакеты направляются на маршрутизатор организации, который переправляет их в требуемую внутреннюю подсеть.

Протокол IPv6

Использование масок является временным решением проблемы дефицита IP-адресов, так как адресное пространство протокола IP не увеличивается, а количество хостов в Интернете растет с каждым днем. Для принципиального решения проблемы требуется существенное увеличение количества IP-адресов. Используемый в настоящее время и рассматриваемый в данном курсе протокол IP называется IPv4 – протокол IP 4-й версии. Для преодоления ограничений IPv4 был разработан протокол IP 6-й версии – IPv6 (RFC 2373, 2460).

Протокол IPv6 имеет следующие основные особенности:

• длина адреса 128 бит – такая длина обеспечивает адресное пространство 2^{128}, или примерно $3.4\cdot10^{38}$ адресов. Такое количество адресов позволит присваивать в обозримом будущем уникальные IP-адреса любым устройствам;

• автоматическая конфигурация – протокол IPv6 предоставляет средства автоматической настройки IP-адреса и других сетевых параметров даже при отсутствии таких служб, как DHCP;

• встроенная безопасность – для передачи данных является обязательным использование протокола защищенной передачи IPsec. Протокол IPv4 также может использовать IPsec, но не обязан этого делать.

В настоящее время многие производители сетевого оборудования включают поддержку протокола IPv6 в свои продукты, однако преобладающим остается протокол IPv4. Связано это с тем, что IPv6 обратно несовместим с IPv4 и процесс перехода сопряжен с определенными трудностями.

Протокол ARP

Протокол IP действует на сетевом уровне модели OSI, поэтому IP-адреса называются сетевыми. Они предназначены для передачи сообщений в составных сетях, связывающих подсети, построенные на различных локальных или глобальных сетевых технологиях, например, Ethernet или ATM. Однако для непосредственной передачи сообщения в рамках одной подсети вместо IP-адреса нужно использовать локальный (аппаратный) адрес технологии канального уровня, чаще всего MAC-адрес. При этом к IP-пакету добавляются заголовок и концевик кадра канального уровня, в заголовке указываются MAC-адреса источника и приемника кадра.

При формировании кадра канального уровня возникает проблема: каким образом по известному IP-адресу определить соответствующий MAC-адрес? Указанная проблема решается при помощи протокола ARP (Address Resolution Protocol – протокол разрешения адресов).

Протокол ARP определяет MAC-адреса следующим образом. Осуществляется рассылка всем узлам сети специального кадра, который называется ARP-запрос (ARP Request). В этом кадре содержится IP-адрес компьютера, у которого требуется узнать MAC-адрес. Каждый узел сети принимает ARP-запрос и сравнивает IP-адрес из запроса со своим IP-адресом. Если адреса совпадают, узел высылает ARP-ответ (ARP Reply), содержащий требуемый MAC-адрес. Результаты своей работы протокол ARP сохраняет в специальной таблице, хранящейся в оперативной памяти, которая называется ARP-кэш. При необходимости разрешения IP-адреса, протокол ARP сначала ищет IP-адрес в ARP-кэше и только в случае отсутствия нужной записи производит рассылку ARP-запроса.

ARP-кэш имеет следующий вид: Записи в ARP-кэше могут быть двух типов: статические и динамические. Статические записи заносятся в кэш администратором при помощи утилиты arp с ключом /s. Динамические записи помещаются в кэш после полученного ARP- ответа и по истечении двух минут удаляются.

Удаление происходит для того, чтобы при перемещении в другую подсеть компьютера с MAC-адресом, занесенным в таблицу, кадры не отправлялись бесполезно в сеть. Иногда требуется по известному MAC-адресу найти IP-адрес (например, при начале работы компьютеров без жесткого диска, у которых есть MAC-адрес сетевого адаптер и им нужно определить свой IP-адрес). В этом случае используется реверсивный протокол RARP (Reverse ARP).

Задание

Интернет-провайдер назначил компьютеру IP-адрес: 110.210.116.69/22 (после косой черты за IP-адресом указано число разрядов маски сети провайдера). Количество подсетей провайдера – 4.

Записать IP-адрес в двоичном коде, подчеркнуть биты блока IP-адреса сети провайдера (одной чертой) и биты адресов подсетей (двумя чертами);

Записать в двоичном виде блок адресов сети провайдера;

Записать в дот-нотации блок адресов сети провайдера;

Записать в двоичном виде маску сети;

Записать в дот-нотации маску сети;

Определить номер подсети, в которой находится компьютер (в десятичной системе);

Определить номер компьютера в подсети (в десятичной системе);

Определить максимальное количество компьютеров в подсети.

(Для каждого студента цифровые величины в задании индивидуальны).

Варианты задач:

№ варианта	IP адрес	Кол-во подсетей
1	214.107.30.99/17	16
2	235.86.29.103/24	8
3	211.110.227.46/22	32
4	114.207.45.95/22	4
5	93.228.44.112/17	128
6	208.113.237.18/23	8
7	217.104.60.91/18	64
8	238.83.58.111/22	16
9	196.125.169.21/17	32
10	139.210.75.87/25	4
11	90.231.73.115/23	16
12	120.201.219.39/17	8
13	220.101.131.82/19	16
14	241.80.97.119/21	64
15	205.116.173.34/19	32
16	108.213.106.79/21	8
17	100.234.105.23/20	4
18	72.249.124.88/25	2
19	223.98.121.77/19	4
20	244.77.118.54/20	64
21	123.198.157.26/18	32
22	109.216.136.70/22	8
23	84.236.134.43/19	128

№ варианта	IP адрес	Кол-во подсетей
24	202.127.143.14/20	16
25	244.77.118.54/20	64
26	123.198.157.26/18	32
27	109.216.136.70/22	8
28	84.236.134.43/19	128
29	81.240.165.51/18	16
30	139.210.75.87/25	4
31	214.107.30.99/17	16

Выводы

Сделать выводы по проделанной работе.

Сведения об авторах

Мусихин Александр Григорьевич, кандидат технических наук, доцент кафедры вычислительной техники Института информационных технологий, РТУ МИРЭА.

Смирнов Николай Алексеевич, кандидат технических наук, доцент, про-фессор кафедры вычислительной техники Иститута информационных тех-но-логий, РТУ МИРЭА