МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«Санкт-Петербургский государственный университет аэрокосмического приборостроения»

ОСНОВЫ ОРГАНИЗАЦИИ ЭВМ

(Архитектура ЭВМ**)**

Методические указания
к выполнению лабораторных работ

Санкт-Петербург
2010

ВВЕДЕНИЕ

Под архитектурой ЭВМ, как правило, понимают организацию машины с точки зрения программиста. Данное понятие - довольно ёмкое и включает в себя все, что находит отражение в машинном языке: организация системы памяти, механизмы адресации, организация внутренней памяти процессора, состав системы команд, форматы команд, средства управления периферийными устройствами, система прерывания, иногда функциональная схема процессора и т.п.

При этом к архитектуре не относятся особенности физической (технической) реализации машины и такие характеристики, как быстродействие, невидимое программисту совмещение операций во времени и другие особенности, не отражающиеся в языке машины.

ЭВМ семейства VAX относятся к классу супермини-ЭВМ. Эти 32-разрядные машины, разработанные американской фирмой Digital Equipment Corporation (DEC), обеспечивают поддержку виртуального адресного пространства - 4Гбайта, что и подчеркнуто в самом названии VAX – Virtual Address eXtension. Вычислительные системы VAX могут быть использованы для работы в режиме разделения времени, обслуживая одновременно 100 и более пользователей, а также для решения широкого круга задач, решение которых занимает много времени даже на ЭВМ с очень высоким быстродействием. Они обеспечивают наибольшую эффективность при решении задач обработки больших обьемов информации, структурно организованных в виде стеков, таблиц и очередей; сбора, обработки информации и управления в режиме реального времени, побайтной обработки и передачи данных.

Как предмет изучения, архитектура VAX интересна тем, что она может быть рассмотрена как базовая при изучении организации вычислительных машин и систем, а также при обучении программированию на языках ассемблера. ЭВМ семейства VAX имеют одну из самых мощных и универсальных систем адресации. Форматы команд имеют четкую, строгую организацию, что позволяет легко анализировать машинные программы "в ручную", без использования вспомогательных средств.

1. ПРОЦЕССОР И СТРУКТУРА ПАМЯТИ

1.1 ОСНОВНОЙ ПРОЦЕССОР

Процессор ЭВМ семейства VAX подключается к интерфейсу "Общая шина" и управляет распределением общей шины между памятью и контроллерами внешних устройств (рис.1.1). Процессор выполняет следующие действия: формирование адреса для выборки команды из памяти, дешифрирование команды, определение исполнительных адресов всех операндов, участвующих в операции, выполнение операции, формирование признаков результата выполняемой команды. При выполнении большинства команд устанавливаются следующие признаки:

C=1 , если в результате выполнения арифметической операции произошел перенос или заимствование из старшего разряда результата;

V=1, если произошло арифметическое переполнение;

Z=1, если результат выполнения операции равен нулю;

N=1, если результат выполнения операции отрицательный.

Указанные признаки вместе с информацией о разрешении прерываний сохраняются в слове состояния процессора.

Процессор содержит шестнадцать 32-разрядных регистров общего назначения (РОН) R0 - R15.

Эти регистры могут быть использованы как регистры данных, индексные регистры, указатели адресов, таблиц, списков и т.п., указатели стека и выполнять другие функции. Конкретное использование регистров зависит от выбранного режима адресации. Среди этих регистров следует особо выделить четыре регистра (R12 - R15).

Регистр R15 используется в качестве счетчика команд и называется программным счетчиком (Program Counter) или просто PC. Он всегда содержит адрес следующего обрабатываемого байта команды, что позволяет получить несколько дополнительных режимов адресации.

Регистр R14 употребляется как указатель стека SP (Stack Pointer).

Регистр R13 является указателем кадра вызова, который создается для хранения отдельных разрядов слова состояния процессора и содержимого некоторых регистров при вызове процедуры. Регистр R12 при вызове процедуры используется как указатель списка аргументов.

Специальное назначение регистров R12 - R14 не исключает возможность их использования для других целей.

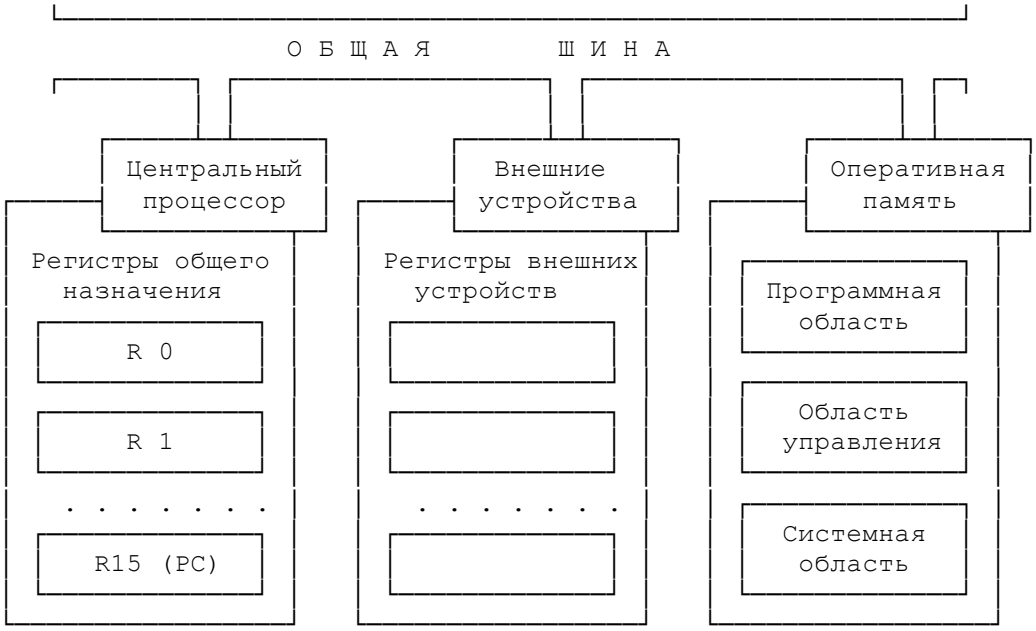


Рис.1.1 Общая структурная схема ЭВМ семейства VAX

1.2 ПРЕДСТАВЛЕНИЕ ЧИСЕЛ В ЭВМ СЕМЕЙСТВА VAX

1.2.1 Представление целых чисел

Большинство целочисленных операций в ЭВМ семейства VAX выполняются над группами из 8, 16 или 32 двоичных разрядов (или битов). Группа из 8 битов называется байтом (byte). Информация, содержащаяся в байте, в ЭВМ представляется в двоичном виде. Однако для человека более удобным является шестнадцатеричное представление, когда информация в байте представляется в виде двух шестнадцатеричных цифр. Отметим, что в ЭВМ семейства VAX шестнадцатеричная система применяется для записи данных, адресов и команд. При этом может возникнуть неоднозначность в интерпретации чисел. Например, число 1734 может интерпретироваться и как десятичное, и как шеснадцатеричное. Чтобы избежать недоразумений, шестнадцатеричные числа, где необходимо, будем сопровождать префиксом ^X. Числа без префикса ^X будем рассматривать как десятичные. Таким образом, если предыдущее число обозначить шестнадцатеричным, то его следовало бы записать как ^X 1734.

В восьми битах можно составить только 2**8 , или 256, двоичных комбинаций. Чтобы избежать подобных ограничений, в ЭВМ семейства VAX для хранения большого количества информации используется 2, 4, 8 или 16 байтов. Информация, содержащаяся в двух байтах, называется словом (word). Более крупными единицами информации являются длинное слово (longword), квадрослово (quadword) и октаслово (octaword). Единицы информации, используемые в ЭВМ семейства VAX приведены на рис.1.2.

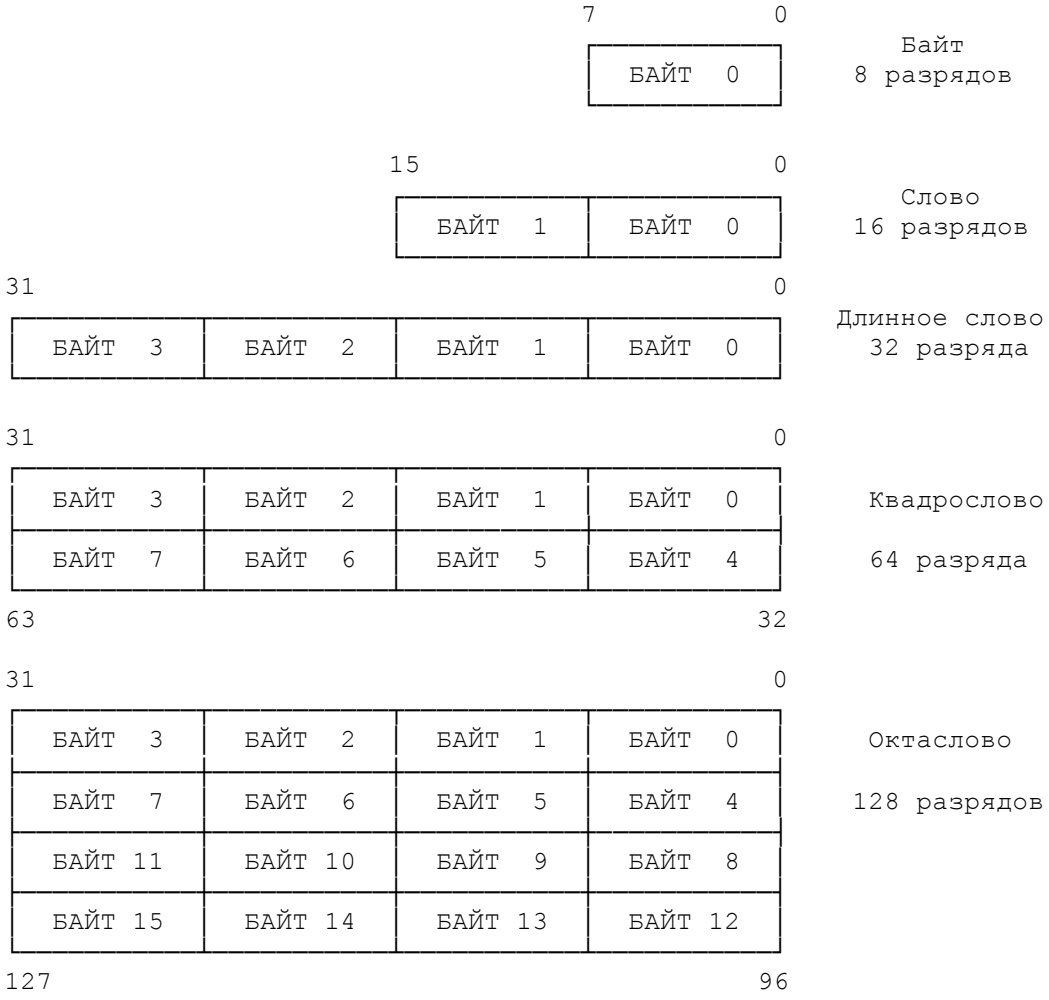


Рис. 1.2 Единицы информации в ЭВМ семейства VAX

Представленные на рис.1.2 форматы используются для хранения положительных и отрицательных целых чисел. Наибольшее применение находят форматы байта, слова и длинного слова. Квадрослова и октаслова лишь частично используются для представления целых чисел и поддерживаются ограниченным числом инструкций. Максимальное значение целого числа зависит от используемого формата (табл.1.1). Отрицательные числа представляются в дополнительном коде. Знак числа всегда содержится в последнем (старшем) разряде (7 - для байта, 15 - для слова и т.д.). Если значение знакового разряда 0, то число положительное, если 1, - число отрицательное.

Диапазон представления целых чисел Таблица 1.1

Формат числа	Диапазон представления целых чисел	
	со знаком	без знака
Байт	от -128 до 127	от 0 до 255
Слово	от -32768 до 32767	от 0 до 65535
Длинное слово	от -(2**31) до (2**31)-1	от 0 до (2**32)-1
Квадрослово	от -(2**63) до (2**63)-1	от 0 до (2**64)-1
Октаслово	от -(2**127) до (2**127)-1	от 0 до (2**128)-1

1.2.2 Представление чисел с плавающей запятой

Числа с плавающей запятой используют для расширения диапазона представления чисел и повышения точности вычислений при решении сложных научно-технических задач, в которых диапазон представления целых чисел оказывается недостаточным для представления данных.

Число с плавающей запятой является одним из вариантов нормальной формы представления чисел. При этом мантисса хранится в памяти ЭВМ всегда в нормализованном виде (первый разряд после запятой отличен от нуля), а порядок заменяется так называемой характеристикой.

Характеристика - это специфичная машинная форма представления порядка и в отличие от истинного порядка, который есть число со знаком, всегда является положительным числом. Значение характеристики (E) определяется как сумма порядка (P) и постоянного смещения (CM)

$$E = P + CM \ .$$

Смещение, с учетом того, что ноль является положительным числом, равно максимальному порядку плюс 1 (или абсолютной величине самого большого по модулю отрицательного порядка). Тогда характеристика самого маленького числа (с порядком P = -P[max]-1) будет равна нулю (E = 0), характеристика самого большого числа (с порядком P = P[max]) – E = 2P[max]+1, а характеристика числа с нулевым порядком – E = CM = P[max]+1. Характеристика числа является смещенным порядком, поэтому ее часто называют просто порядком.

В машинах семейства VAX для выполнения вычислений над числами с плавающей запятой используется двоичная система счисления. При этом старший разряд нормализованной мантиссы всегда равен 1. Так как нет необходимости хранить явную константу, старший разряд мантиссы при записи числа в память опускается, а при выборке числа из памяти - восстанавливается. Такая форма хранения чисел с плавающей запятой называется записью со скрытым битом. Отметим, что запись со скрытым битом позволяет хранить дополнительный разряд мантиссы, что несколько повышает относительную точность вычислений.

Характеристика числа (смещенный порядок) есть целое положительной число (без знака). Тогда при записи числа с плавающей запятой необходимо указывать только один знак - знак числа, который совпадает со знаком мантиссы. При этом знак числа указывается отдельно и мантисса храниться в прямом коде (т.е. записывается модуль мантиссы). Таким образом, число с плавающей запятой представляется в следующем виде

Знак числа	Характеристика	Модуль мантиссы

В ЭВМ семейства VAX применяются четыре формата для записи чисел с плавающей запятой: формат типа F, формат типа D, формат типа G и формат типа H. Требуемый для них объем памяти изменяется от 32-битового длинного слова (формат F) до 128-битового октаслова (формат H). Кроме того, форматы типов D и G, каждый занимающий по 64 бита, различаются числом битов, отведенных для порядка и мантиссы.

1.2.2.1 Числа с плавающей запятой в формате F

В формате F для записи числа с плавающей запятой отводится 32 разряда (одно длинное слово). При этом, знак числа занимает один бит, под характеристику отводится 8 разрядов, и 24-разрядная мантисса (с учетом скрытого бита) хранится в 23 разрядах.

В формате F на ЭВМ VAX-11 могут быть представлены числа в диапазоне от 2**(-129) до (2**127)-1. Смещение порядка равно 128 (10000000 в двоичном виде). Следовательно, имеется следующая таблица порядков :

Десятичный порядок	Двоичное представление	Шестнадцатеричное представление
+127	11111111	FF
+1	10000001	81
0	10000000	80
-1	01111111	7F
-128	00000000	00

В качестве примера рассмотрим представление десятичного числа 0.75 или 3/4 * (2**0) в виде двоичного числа с плавающей запятой.Число 0.75 в нормальной форме в двоичной системе счисления:

Знак	Порядок	Мантисса
0	00000000	11000000000000000000000

Число 0.75 в форме числа с плавающей запятой в двоичной системе счисления:

Знак	Характеристика	Мантисса (со скрытым битом)
0	10000000	10000000000000000000000

ЭВМ семейства VAX разрабатывались так, чтобы обеспечивалась некоторая совместимость с 16-разрядными ЭВМ (в том числе с ЭВМ семейства PDP-11). 32-битовые и 64-битовые числа с плавающей запятой в памяти этих ЭВМ размещаются как последовательности 16-битовых слов, причем первой располагается старшая значащая часть числа. Аналогичная схема записи чисел с плавающей запятой принята и для ЭВМ семейства VAX. При записи чисел с плавающей запятой в формате 16-битовых слов это не вызывает затруднений, так как соответствует естественному для человека порядку записи. Однако при записи чисел с плавающей запятой в формате 32-битовых слов внутри длинного слова младшие 16 разрядов числа меняются местами со старшими разрядами числа. Соответственно, такой порядок записи чисел с плавающей запятой распространяется и на другие форматы. Формат F представлен на рис.1.3.

В мантиссе разряды увеличивают свое значение от 16-го до 31 (младшая часть) и от 0-го до 6-го (старшая часть). Число 0.75 в формате F будет представлено следующим образом:

Двоичное представление	Шестнадцатеричное представление
0000 0000 0000 0000 0100 0000 0100 0000	^ X 00004040

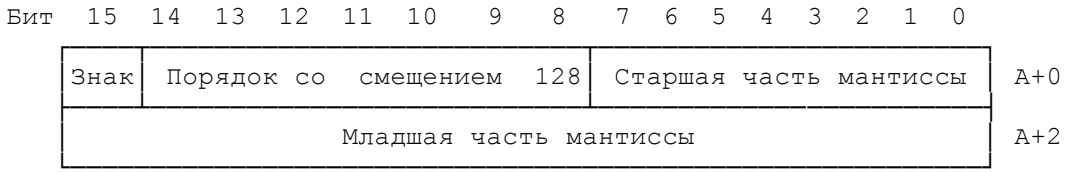


Рис.1.3 Формат F

1.2.2.2 Числа с плавающей запятой в форматах типа D, G, H

Числа с плавающей запятой в формате F иногда называют числами с плавающей запятой одинарной точности (точность представления составляет примерно 7 значащих десятичных цифр).

Числа в формате D, которые иногда называют числами с плавающей запятой двойной точности, имеют 64-битовую длину и поэтому занимают квадрослово. Формат D идентичен формату F, за исключением того, что поле мантиссы расширяется от 23 до 55 битов (добавляются 32 бита; скрытый бит не считается). Формат D приведен на рис.1.4.

Дополнительные биты поля мантиссы допускают представление чисел с точностью примерно 16 значащих десятичных цифр. Число .75 в формате D будет выглядеть следующим образом :

Содержимое	Адрес
4040	начальный адрес
0000	начальный адрес + 2
0000	начальный адрес + 4
0000	начальный адрес + 6

Как квадрослово это будет ^X 0000000000004040.

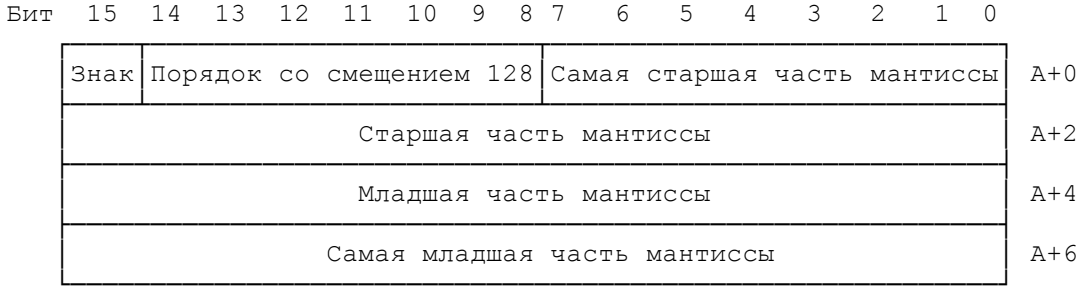


Рис 1.4 Формат D

Представление чисел с плавающей запятой в форматах типов F и D, используемое в ЭВМ семейства VAX, совпадает с представлением таких чисел на ЭВМ семейства PDP-11, выпускавшихся фирмой DEC. Это позволяет обеспечить программную совместимость этих двух семейств ЭВМ. Существует, однако, новый стандарт для представления чисел с плавающей запятой, который был одобрен Институтом инженеров по электротехнике и радиоэлектронике (IEEE-754).

Для представления числа в формате G (рис1.5) используется 64 бита, как и в формате D. Однако порядок занимает более длинное поле, а маннтисса - соответственно поле меньшей длины. Это означает, что формат G несколько менее точный, но охватывает значительно больший диапазон чисел, чем формат D (см. диапазоны и точность представления в табл.1.2).

Для представления чисел в формате H (рис.1.6) используется 128 битов, что в два раза больше, чем для форматов типа D и G. И для порядка, и для мантиссы отводятся дополнительные биты, что обеспечивает расширенный диапазон значений и значительно большую точность.

Существующие другие форматы данных в ЭВМ семейства VAX (для представления символьных и числовых строк и т.д.) в настоящем пособии не рассматриваются.

Таблица 1.2

Представление чисел с плавающей запятой в ЭВМ семейства VAX

Тип формата	Длина поля, бит			Диапазон значений	Число значащих цифр
	Полная	Порядок	Мантисса		
F	32	8	23	0.29*10**(-38) - 1.7*10**38	7
D	64	8	55	0.29*10**(-38) - 1.7*10**38	16
G	64	11	52	0.56*10**(-308)- 0.9*10**308	15
H	128	15	112	0.84*10**(-4932) - 0.59*10**4932	33

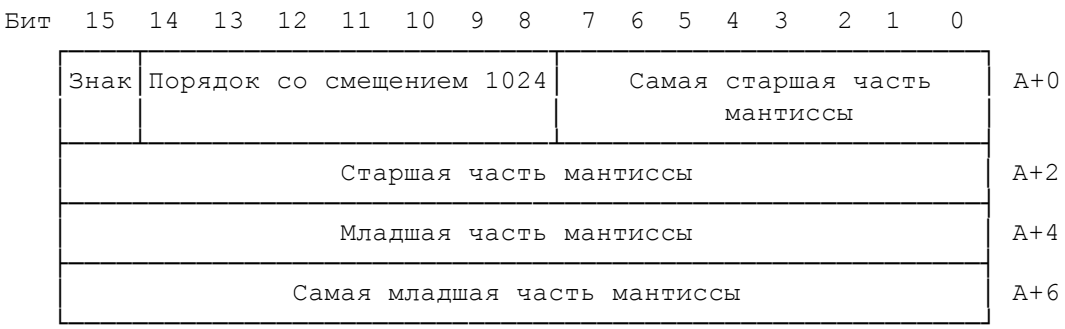


Рис 1.5 Формат G

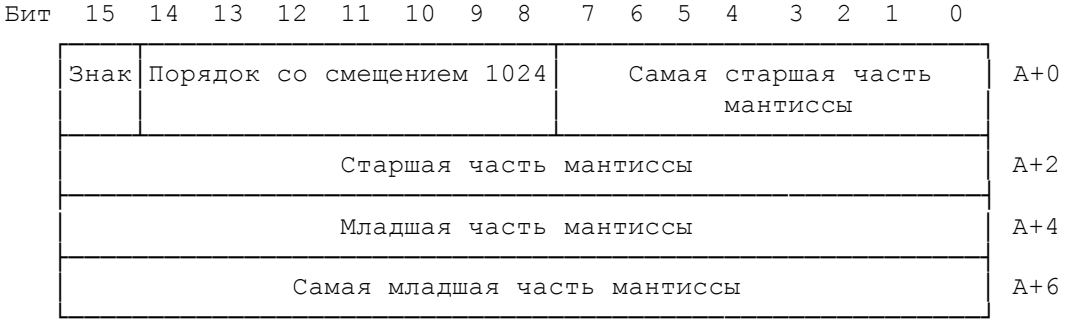


Рис 1.5 Формат G

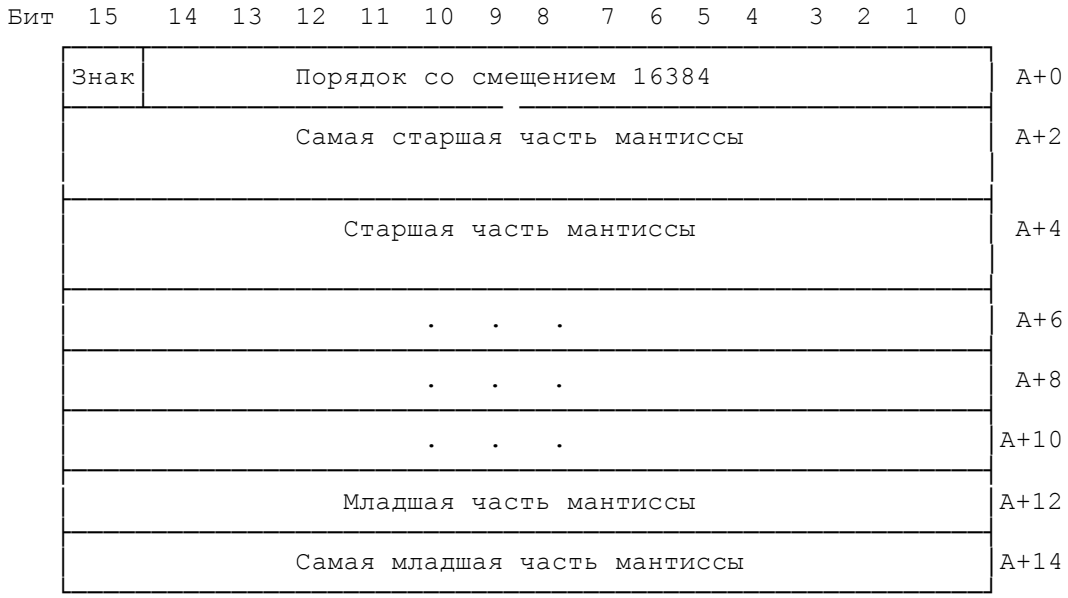


Рис.1.6. Формат H

1.3 ОРГАНИЗАЦИЯ ПАМЯТИ

Память ЭВМ семейства VAX можно рассматривать как набор ячеек памяти, каждая из которых содержит один байт (от ^X 00 до ^X FF). Каждая ячейка памяти однозначно идентифицируется адресом, который в ЭВМ семейства VAX представляет собой 32-битовое двоичное целое число (8 шестнадцатеричных цифр). Поскольку в 32 битах могут быть образованы (2**32), или 4292967926 двоичные комбинации, то в программе можно было бы адресовать 4 Гбайта памяти (^X 00000000 - ^X FFFFFFFF). Объем используемой в эмулирующей программе памяти, составляет 1024 (^X 400) байт. Диапазон возможных адресов в эмулирующей программе: ^X 00000000 - ^X 000003FF. Каждая из перечисленных единиц информации на рис.1.2 сформирована из последовательно расположенных байт и всегда адресуется так же, как младший байт в этой группе. На рис. 1.7 показано возможное содержимое первых 11 байтов памяти.

Шестнадцатеричное содержимое	Шестнадцатеричный адрес
90	00000000
9F	00000001
34	00000002
12	00000003
00	00000004
00	00000005
9F	00000006
EF	00000007
CD	00000008
0B	00000009
00	0000000A

Рис. 1.7 Пример шестнадцатеричного представления содержимого памяти

В памяти ЭВМ 16-битовое слово запоминается в двух смежных адресах. Например, на рис.1.3 слово ^X 1234 запоминается в байтах памяти с адресами ^X 00000002 и ^X 00000003. Причем, младшие разряды слова ^X 34 запоминаются в байте с младшим адресом ^X 00000002, а старшие разряды слова ^X 12 - в байте со старшим адресом ^X 00000003. При таком представлении байты слова необходи мо читать снизу вверх. Чтобы устранить это неудобство, слова в памяти часто представляются в следующем виде:

Шестнадцатеричное содержимое	Шестнадцатеричный адрес
1234	00000002

Здесь указывается только один адрес, однако при этом понимается, что в байте с адресом ^X 00000002 содержится ^X 34, а в байте с адресом ^X 00000003 содержится ^X 12.

В памяти ЭВМ 32-битовые длинные слова запоминаются в четырех смежных байтах. Например, на рис.1.7 длинное слово ^X 000BCDEF запоминается в памяти, начиная с адреса ^X 00000007.

Шестнадцатеричное содержимое	Шестнадцатеричный адрес
EF	00000007
CD	00000008
0B	00000009
00	0000000A

По другому длинное слово в памяти можно преставить в следующем виде:

Шестнадцатеричное содержимое	Шестнадцатеричный адрес
000BCDEF	00000007

Программист должен помнить, что в данном случае длинное слово в действительности занимает байты с адресами от ^X 00000007 до ^X 0000000A, а младшие разряды длинного слова ^X EF содержатся в байте с адресом ^X 00000007.

Квадрослово (64 бита) запоминается в памяти в восьми смежных байтах, а октаслово (128 бит) - в 16-ти смежных байтах.

Предположим, что в 11 байтах , на рис. 1.7, представлены три байта, два слова и одно длинное слово в следующем порядке:

1. Байты располагаются по адресам ^X 00000000, ^X 00000001 и ^X 00000006.
2. Слова начинаются с адресов ^X 00000002 и ^X 00000004.
3. Длинное слово начинается с адреса ^X 00000007.

Содержимое памяти на рис.1.7 может быть представлено и более компактно, как показано на рис.1.8.

Важно понимать, что на рис.1.7 и 1.8 изображена одна и та же информация, которая является просто различным представлением одного и того же содержимого памяти. Отметим, что содержимое ^X 9F байта памяти с адресом ^X 00000006 в один момент времени может об рабатываться программой как байт, в другой момент времени – как часть слова, а в последующее время - как часть длинного слова.

Не существует способа, который позволил бы, посмотрев на байт в памяти, определить, как он будет использоваться.

Шестнадцатеричное содержимое	Шестнадцатеричный адрес
90	00000000
9F	00000001
1234	00000002
0000	00000004
9F	00000006
000BCDEF	00000007

Рис.1.8 Компактное представление содержимого памяти

1.4 РЕЖИМЫ АДРЕСАЦИИ

Формат подавляющего большинства команд ЭВМ семейства VAX включает поле кода операции (КОП) и несколько полей операндов, количество которых определяется соответствующим КОП. Для некоторых операций операнды могут отсутствовать. В этом случае команда состоит только из поля кода операции. Поле КОП имеет самый младший адрес в команде и, как правило, занимает один байт, хотя есть отдельные команды, у которых это поле занимает 2 байта. Затем располагаются поля операндов, число которых может быть от 1 до 6. Поле операнда в ЭВМ семейства VAX называется спецификацией операнда и может размещаться в одном или нескольких байтах, что зависит от способа адресации этого операнда (рис. 1.9).

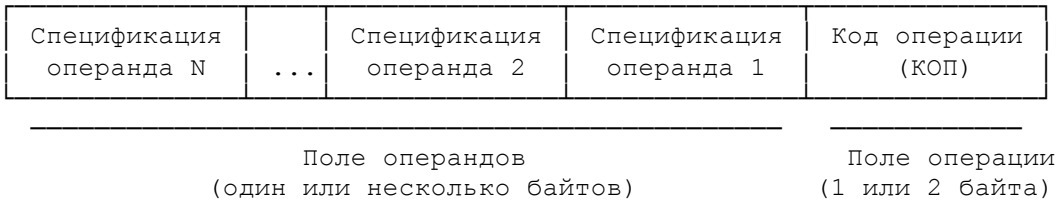


Рис.1.9 Структура машинной команды

Спецификация операнда, как правило, содержит байт-спецификатор и может содержать дополнительное адресное поле. В байте-спецификаторе указывается режим адресации и номер используемого регистра. Наличие, длина и использование дополнительного адресного поля в основном определяется режимом адресации. Адресация в ЭВМ семейства VAX осуществляется достаточно большим числом вариантов, что соответствует разнообразию способов адресации, которые являются развитием способов адресации 16-разрядных ЭВМ семейства PDP-11 и могут быть разделены на 4 группы :

адресация через регистры общего назначения;
адресация через счетчик команд (PC);
адресация с индексацией;
адресация в командах перехода.

1.4.1 Способы адресации через регистры общего назначения

В этих способах адресации для указания адреса операнда используются 15 регистров общего назначения (R0 - R14) и одиннадцать режимов адресации (табл.1.3). Формат поля операнда, в котором используются эти режимы, имеет несколько модификаций. Для первых пяти способов адресации поле операнда содержит только 1 байт-спецификатор:

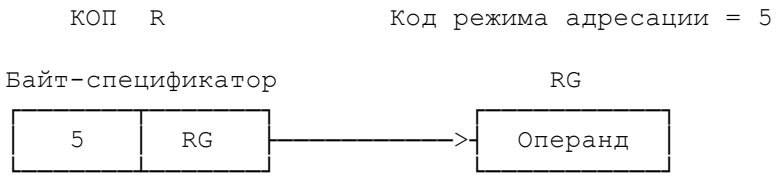
Коды режимов адресации приведены в табл.1.3. Одного байта вполне достаточно для хранения номера регистра (от 0 до 14) и кода режима адресации (всего их 16).

Режимы адресации через регистры общего назначения Таблица 1.3

Режим (способ) адресации	Код режима	Формат
Регистровая адресация	5	R
Косвенно-регистровая адресация	6	(R)
Адресация с автоувеличением	8	(R)+
Косвенная адресация с автоувеличением	9	@(R)+
Адресация с автоуменьшением	7	- (R)
Адресация по смещению	A,C,E	смещение (R)
Косвенная адресация по смещению	B,D,F	@ смещение (R)

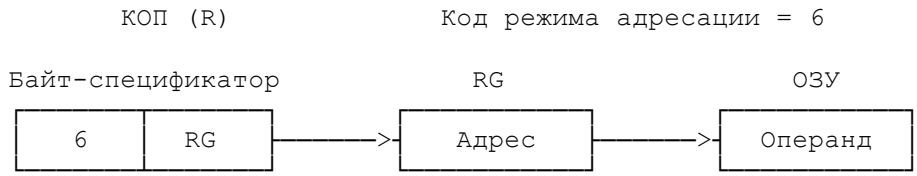
Для режимов адресации со смещением, кроме номера регистра и кода режима, необходимо задавать дополнительную информацию, которая определяет смещение. Величина смещения в дополнительном коде указывается в дополнительном адресном поле, в связи с чем для этих режимов поле операнда имеет длину более одного байта.

1.4.1.1 Прямая регистровая адресация



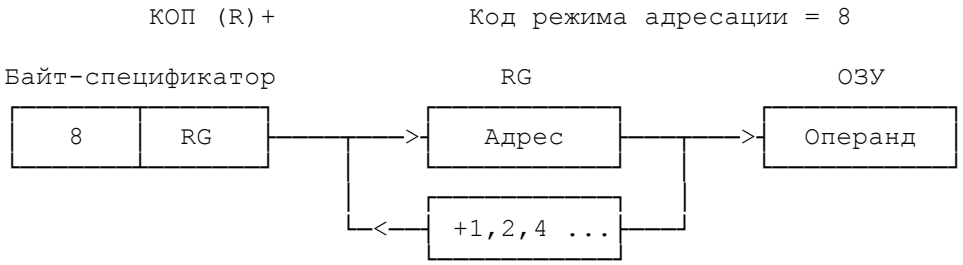
В этом режиме операндом является содержимое РОН, к которому происходит обращение. Этот способ адресации обеспечивает наиболее высокую скорость обработки информации.

1.4.1.2 Косвенно-регистровая (простая косвенная) адресация



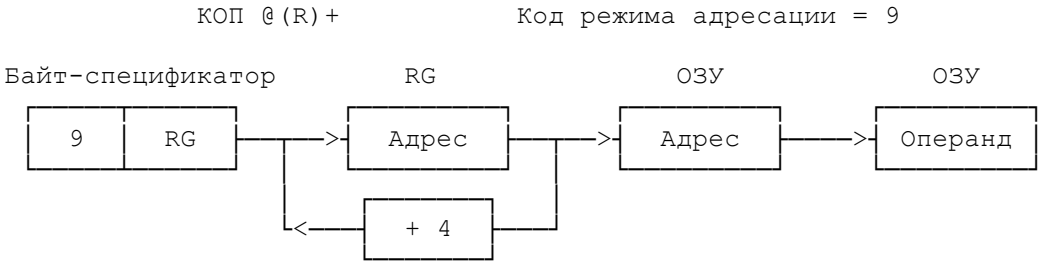
В этом режиме операндом является содержимое поля, адрес которого содержится в используемом регистре. Размер поля (формат операнда) - байт (B), слово (W), длинное слово (L), квадрослово (Q) или октаслово (O), а для операций с плавающей точкой, соответственно, формат числа с плавающей точкой - определяется кодом операции (КОП).

1.4.1.3 Адресация с автоувеличением (простая косвенная с автоувеличением)



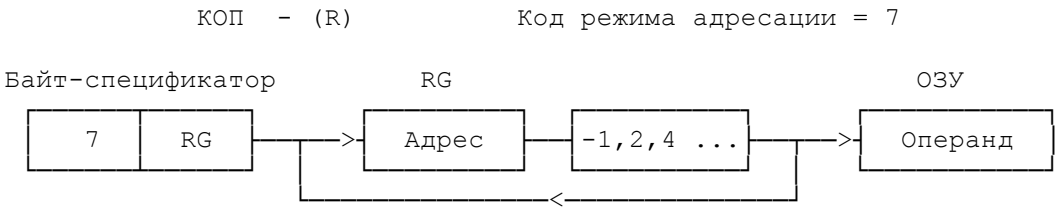
В этом режиме используемый регистр содержит адрес операнда. После выборки операнда производится увеличение содержимого регистра на 1, 2, 4, 8 или 16 соответственно для операндов в формате байта, слова, длинного слова, квадрослова или октаслова. Формат операнда определяется КОП. Режим с автоувеличением особенно эффективен при работе с массивами данных одного формата, так как позволяет без дополнительных команд получать в регистре адрес следующего элемента массива, а также при обработке таблиц и организации стека.

1.4.1.4 Косвенная адресация с автоувеличением (двойная косвенная с автоувеличением)



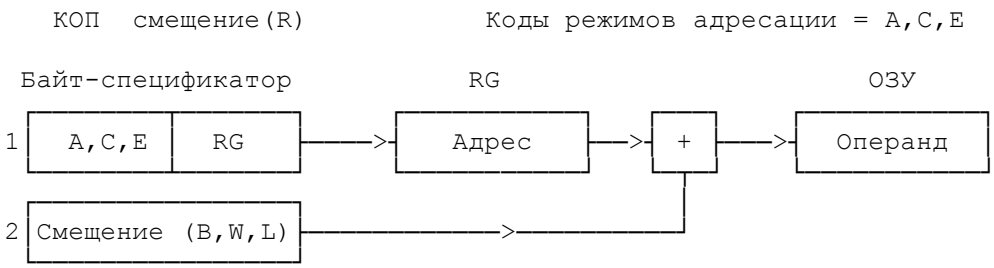
Используемый в данном режиме регистр содержит адрес поля, содержимое которого является адресом операнда. Поле занимает в ОЗУ адреса A, A+1, A+2, A+3. После выборки адреса операнда содержимое указанного регистра увеличивается на 4 вне зависимости от формата используемых в операции данных.

1.4.1.5 Адресация с автоуменьшением (простая косвенная с автоуменьшением)

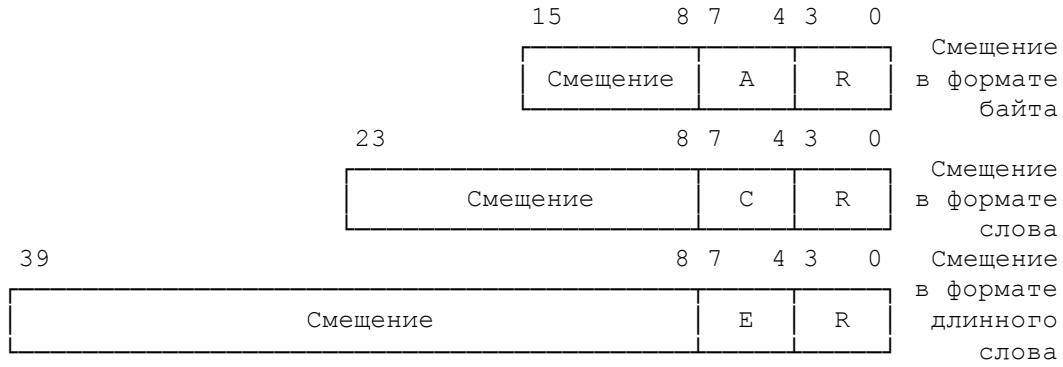


В этом режиме перед выполнением операции содержимое используемого регистра уменьшается на размер, определяемый форматом данных операнда (по коду операции), после чего регистр содержит адрес операнда. Этот режим эффективен при обработке данных в массивах в порядке, обратном по отношению к адресации с автоувеличением.

1.4.1.6 Адресация по смещению

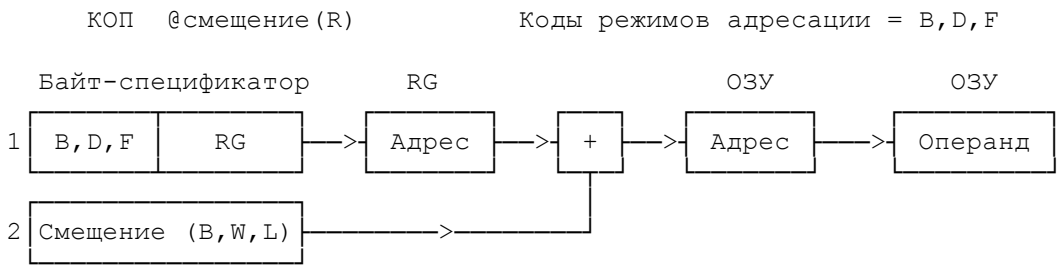


В этом режиме адрес операнда определяется суммой содержимого используемого регистра и смещения. Значение смещения хранится в дополнительном коде непосредственно в поле операнда в формате байта, слова или длинного слова со знаком (табл.1.4). В соответствии с этим формат поля операнда может иметь три модификации:



Если смещение задано в формате байта или слова, то при вычислении адреса операнда выполняется расширение старшего разряда смещения до формата длинного слова.

1.4.1.7 Косвенная адресация по смещению



Форматы поля операнда для этого режима совпадают с форматами предыдущего режима. Однако в отличие от режима адресации по смещению, сумма содержимого используемого регистра и смещения образует не адрес операнда, а адрес ячейки ОЗУ, в которой находится адрес операнда.

Возможные варианты смещений — Таблица 1.4

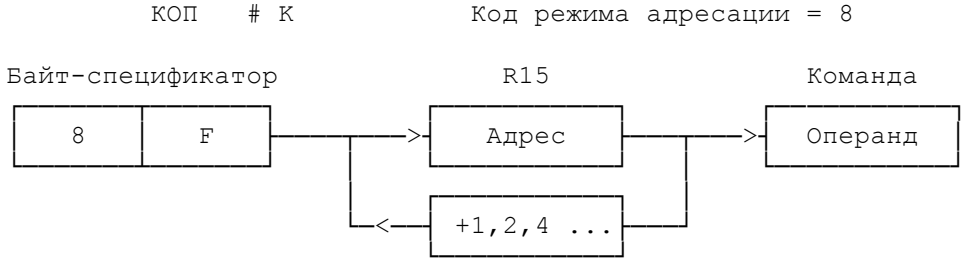
Смещение	Код режима адресации	
	По смещению	Косвенный по смещению
Байт	^X A	^X B
Слово	^X C	^X D
Длинное слово	^X E	^X F

1.4.2 Способы адресации через счетчик команд PC (R15)

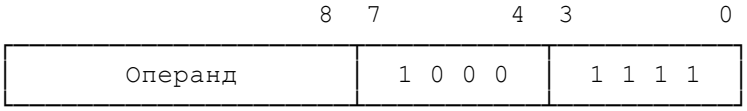
В этих способах адресации вместо регистров общего назначения R0-R14 используется программный счетчик PC (R15). Использовать программный счетчик можно во всех режимах точно так же, как и регистры общего назначения R0-R14, но нежелательно, так как это может привести к непредсказуемым результатам. Основные варианты использования R15 в качестве регистра-указателя при адресации данных приведены в табл.1.5.

Перед выполнением команды программный счетчик содержит начальный адрес этой команды. В процессе выполнения команды центральный процессор после выборки очередного байта увеличивает содержимое программного счетчика на 1, поэтому программный счетчик всегда содержит адрес того байта, который будет использоваться (и соответственно, выбираться) на следующем этапе (шаге) работы центрального процессора.

1.4.2.1 Непосредственная адресация



Для этого способа адресации формат поля операнда выглядит следующим образом:



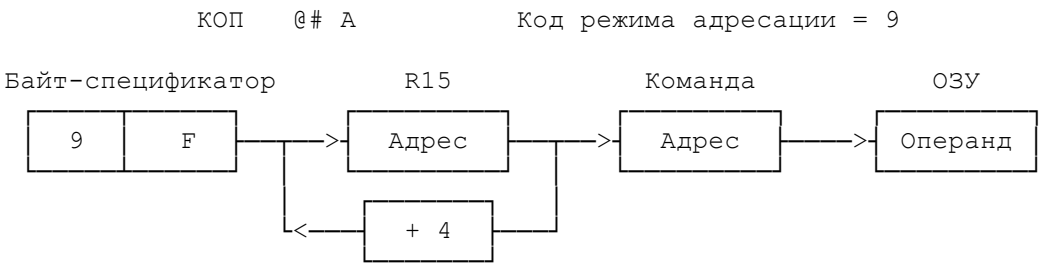
Непосредственная адресация - это режим адресации с автоувеличением, в котором вместо регистра общего назначения R0-R14 используется счетчик команд PC. В соответствии с режимом адресации (автоинкрементная) после выборки из памяти операнда содержимое регистра, а в данном случае программного счетчика, увеличивается на длину операнда в байтах. Таким образом, в R15 после выборки операнда будет находиться начальный адрес следующей спецификации операнда. Так как непосредственная адресация используется в программах только для определения констант, то она неприменима для операндов-приемников (за исключением случаев самомодифицирующихся программ). Непосредственная адресация обеспечивает уменьшение времени доступа к константам.

Основные способы адресации с использованием программного счетчика — Таблица 1.5

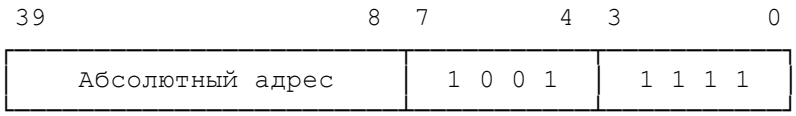
Способ адресации	Спецификация операнда	Код режима	Формат
Непосредственная адресация	x...x 8F	8	# K
Абсолютная адресация	xxxxxxxx 9F	9	@# A
Относительная адресация			
(смещение - байт)	xx AF	A	A
(смещение - слово)	xxxx CF	C	A
(смещение - длинное слово)	xxxxxxxx EF	E	A
Косвенная относительная адресация			
(смещение - байт)	xx BF	B	@A
(смещение - слово)	xxxx DF	D	@A
(смещение - длинное слово)	xxxxxxxx FF	F	@A

Примечание: В описании формата символ "A" обозначает адрес, символ "K" обозначает константу. Символы "xx" - значения в дополнительном поле спецификации операнда.

1.4.2.2 Абсолютная адресация

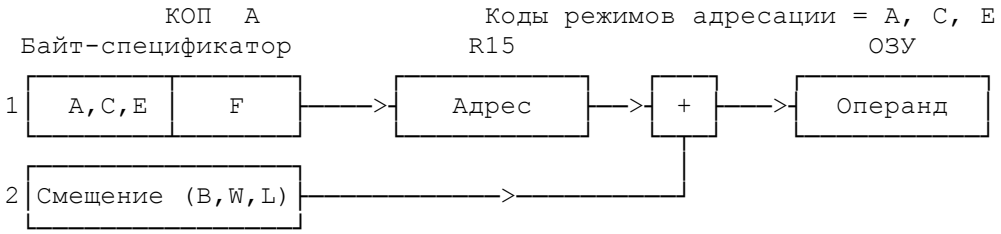


Формат поля операнда при этом режиме имеет следующий вид:



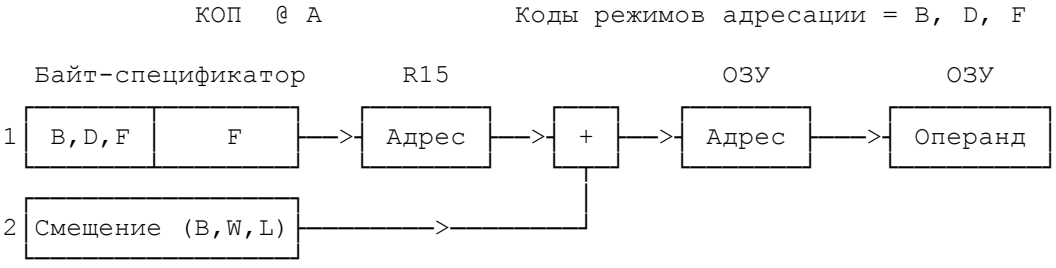
Абсолютная адресация реализуется как двойная косвенная адресация с автоувеличением по счетчику команд PC (R15). В соответствии с режимом адресации в указанном в байте-спецификаторе регистре (в данном случае PC) хранится адрес поля, в котором содержится адрес операнда. После определения адреса операнда содержимое программного счетчика увеличивается на 4 и будет содержать или начальный адрес следующей спецификации операнда, или начальный адрес следующей команды.

1.4.2.3 Относительная адресация



Данный способ адресации реализуется с помощью режима адресации по смещению в формате байта, слова или длинного слова, только вместо регистра общего назначения R0-R14 используется регистр R15. Исполнительный адрес операнда определяется как сумма смещения и текущего значения PC, которое равно начальному адресу или следующей спецификации операнда, или следующей команды. Этот режим адресации очень эффективен при построении позиционно-независимых программ, допускающих перемещение в памяти.

1.4.2.4 Косвенная относительная адресация

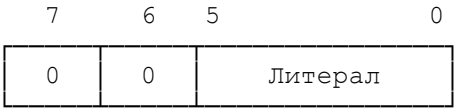


Этот способ адресации реализуется с помощью режима косвенной адресации по смещению в формате байта, слова или длинного слова, только вместо регистра общего назначения R0-R14 используется регистр R15. Адрес операнда находится в ячейке ОЗУ, адрес которой определяется как сумма смещения и текущего значения PC, которое равно начальному адресу или следующей спецификации операнда, или следующей команды.

1.4.3. Литеральная адресация

```
КОП  #литерал              Коды режимов адресации = 0,1,2,3
```

Литеральная адресация является особым специальным случаем короткой непосредственной адресации. Признаком этой адресации являются два нуля в старших (6 и 7) разрядах байта-спецификатора. В младших шести разрядах (с 0 по 5) байта-спецификатора вместо номера регистра и младших разрядов кода способа адресации помещается значение операнда. Таким образом, литеральная адресация (один способ адресации) использует как бы четыре режима адресации (с 0 по 3) и в нем отсутствует указание регистра. Для способа литеральной адресации формат поля операнда имеет следующий вид:

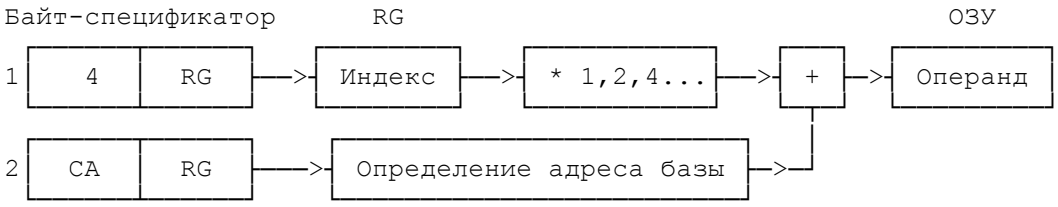


Так как старшие разряды байта-спецификатора при литеральной адресации равны нулю, то литерал – всегда положительное число. Значение литерала для целых чисел находится в диапазоне от 0 до 63. Формат короткого непосредственного операнда (литерала) – целое число или число с плавающей запятой – определяется кодом операции.

Литеральная адресация применяется для экономии памяти при небольших значениях операнда и разрешена только для операнда-источника.

1.4.4. Режимы адресации с индексацией

```
КОП   база [RX]          Код режима адресации = 4
```

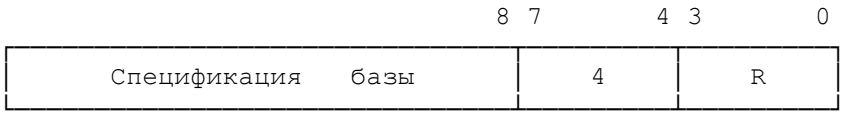


Индексирование - один из способов организации работы с массивами. Для определения адреса произвольного элемента массива вычисляется сумма адреса начала массива (базы) и смещения, которое получается произведением размера элемента на его номер (значение номера первого элемента массива - всегда нуль). Порядковый номер элемента в массиве часто называют индексом.

Для работы с массивами можно использовать любой из отмеченных выше режимов адресации, по отдельности определяя базовый адрес массива и смещение, которое задается индексом. Однако при таком подходе ухудшается наглядность программы и приходится выполнять достаточно много тривиальных операций, которые отвлекают от реализации основного алгоритма.

Режимы адресации с индексацией позволяют объединить процедуры получения базы и смещения (с последующим их сложением), что не только упрощает программирование работы с массивами, но и существенно ускоряет выполнение программы, так как при этом соответствующие операции сложения и умножения выполняются аппаратно.

Поскольку режим с индексацией определяет два значения - базовый адрес и смещение, это нашло свое отражение и в формате поля операнда, который состоит из двух частей и выглядит следующим образом:



Первый байт-спецификатор содержит код режима адресации с индексацией (4) и номер регистра, содержимое которого используется как значение индекса. Далее следует спецификация базы, по которой определяется базовый адрес массива. Спецификация базы имеет такой же формат, что и спецификация операнда. То есть содержит байт-спецификатор и может содержать, в зависимости от способа адресации, дополнительное адресное поле. При этом базовый адрес массива определяется точно также, как и адрес операнда. Очевидно, что при таком порядке определения базового адреса массива в спецификации базы нельзя использовать прямую регистровую, непосредственную и литеральную адресации, а также адресацию с индексацией и адресацию в инструкциях относительного перехода.

Исполнительный адрес операнда определяется как сумма базового адреса массива, который определяется по спецификации базы, и смещения, которое равно произведению содержимого индексного регистра (индекса) на размер элемента массива в байтах. Размер элемента массива определяется по типу данных, указанных в коде операции.

Сочетая индексацию с другими режимами адресации, можно образовать способы адресации, указанные в табл. 1.6.

Способы адресации с индексацией — Таблица 1.6

Способ адресации	Формат
Косвенно-регистровая адресация с индексацией	(RN) [RX]
Индексная адресация с автоувеличением	(RN)+ [RX]
Индексная адресация с автоуменьшением	- (RN) [RX]
Косвенная адресация с автоувеличением и индексацией	@ (RN)+ [RX]
Адресация по смещению с индексацией	смещение (RN) [RX]
Косвенная адресация по смещению с индексацией	@смещение (RN) [RX]
Абсолютная адресация с индексацией	@ # адрес [RX]
Относительная адресация с индексацией	адрес [RX]
Косвенная относительная адресация с индексацией	@ адрес [RX]

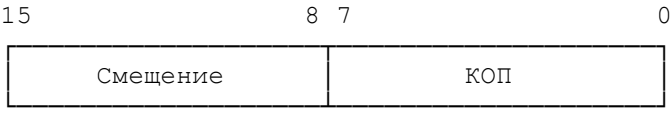
Примечание. (RN) и [RX] - любые из регистров общего назначения R0 ... R14.

При индексации с автоувеличением, при косвенной адресации с автоувеличением и индексацией и при индексации с автоуменьшением не рекомендуется в спецификации базы использовать индексный регистр RX, так как это может привести к непредсказуемым результатам.

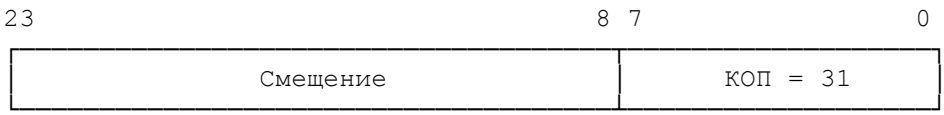
1.4.5. Адресация в командах перехода

Данный режим адресации используется в командах условного перехода, в командах безусловного перехода BRB и BRW, а также в командах организации цикла типа ACB. Так как переходы в программах, как правило, выполняются на ограниченном адресном пространстве относительно счетчика команд, адрес перехода в указанных командах определяется как относительный к программному счетчику PC. При этом способ адресации и регистр-указатель (PC) определяются неявно через код операции. Таким образом, в указанных командах в спецификации операнда, определяющей адрес перехода, указывается только величина смещения относительно текущего значения PC. В командах условного перехода и в команде BRB смещение задается в формате байта, в остальных командах - в формате слова.

Команды условного перехода и команда BRB имеют следующий формат:



Команда BRW отличается на величину смещения:



Смещение рассматривается как число со знаком, представленное в дополнительном коде. При вычислении адреса перехода старший разряд смещения расширяется на старшие разряды до длинного слова. Если смещение задано в формате байта, то максимальное положительное смещение (в сторону увеличения значения PC) будет 127 байт и наибольшее отрицательное смещение (в сторону уменьшения значения PC) - 128 байт. Для команд перехода значение программного счетчика, относительно которого рассчитывается смещение, всегда равно начальному адресу команды, следующей за командой перехода.

1.5 СИСТЕМА КОМАНД

ЭВМ семейства VAX обладают развитой системой команд, которая включает более 300 операций, реализуемых аппаратно. В эмулирующей программе реализовано 87 операций.

Команды ЭВМ семейства VAX - переменной длины, с выравниванием по границе байта. Такой формат делает команды компактными и позволяет расширять их набор. Код операции может занимать один или два байта. В зависимости от команды за кодом операции следует от 0 до 6 спецификаций операндов.

Команды, реализованные в эмулирующей программе, можно разделить на следующие основные группы:

- выполнения целочисленных арифметических и логических операций,
- пересылки,
- безусловного переходов и обращения к подпрограммам,
- условных переходов по кодам условий,
- организации циклов,
- специального назначения.

В данных методических указаниях не рассматриваются команды, не вошедшие в состав эмулирующей программы (выполнение операций над числами с плавающей запятой и символьными строками, команды деления и умножения для целых чисел, команды работы со списками, указателями и т.д.).

Для удобства изложения информация по каждой группе команд сведена в отдельные таблицы. Кроме того, в Приложении приводится общий список команд, реализованных в эмулирующей программе. В описаниях команд операнды имеют обозначение "опN", где N - целое число от 1 до 6. Коды операций приводятся в шестнадцатеричной системе счисления. В мнемоническом обозначении операции после названия могут быть указаны: тип данных, с которым работает данная команда (B,W,L) и число используемых операндов (2 или 3). Для тех команд, которые формируют новые значения признаков (N, Z, V, C), указывается, какие именно признаки устанавливаются по результатам выполнения этих команд. При этом используются следующие обозначения:

* - значение признака устанавливается по результатам выполнения данной команды ;

\- - значение признака в результате выполнения данной команды не изменяется, т.е. сохраняется предыдущее значение.

1.5.1 Команды выполнения арифметических и логических операций над целыми числами

Команды, относящиеся к этой группе приведены в табл. 1.7

Целочисленные арифметические и логические операции Таблица 1.7

Мнемокод	КОП	Операция	Флаги N Z V C	Описание
ADDB2 ADDW2 ADDL2	80 A0 C0	оп2 < оп2 + оп1 Сложение двухоперандное	* * * *	Содержимое оп1 и оп2 складывается, результат запоминается по адресу оп2
ADDB3 ADDW3 ADDL3	81 A1 C1	оп3 < оп2 + оп1 Сложение трехоперандное	* * * *	Содержимое оп1 и оп2 складывается, результат запоминается по адресу оп3
SUBB2 SUBW2 SUBL2	82 A2 C2	оп2 < оп2 - оп1 Вычитание двухоперандное	* * * *	Содержимое оп1 вычитается из содержимого оп2, результат запоминается по адресу оп2
SUBB3 SUBW3 SUBL3	83 A3 C3	оп3 < оп2 - оп1 Вычитание трехоперандное	* * * *	Содержимое оп1 вычитается из содержимого оп2, результат запоминается по адресу оп2
ADWC	D8	оп2 < оп2+оп1+C Сложение с переносом	* * * *	Значение суммы двух операндов (оп1+оп2) складывается со значением кода условия C. Результат запоминается по адресу оп2
SBWC	D9	оп2 < оп2-оп1-C Вычитание с заемом	* * * *	Содержимое оп1 и признака C вычитаются из содержимого оп2. Результат запоминается по адресу оп2
INCB INCW INCL	96 B6 D6	оп1 < оп1 + 1 Инкремент	* * * *	К содержимому оп1 прибавляется 1. Результат запоминается по адресу оп1
DECB DECW DECL	97 B7 D7	оп1 < оп1 - 1 Декремент	* * * *	Из содержимого оп1 вычитается 1. Результат запоминается по адресу оп1
MNEGB MNEGW MNEGL	8E AE CE	оп2 < - оп1 Пересылка дополнения	* * * *	Дополнение содержимого оп1 пересылается по адресу оп2
MCOMB MCOMW MCOML	92 B2 D2	оп2 < $\overline{\text{оп1}}$ Пересылка инверсии	* * 0 -	Инверсия содержимого оп1 пересылается по адресу оп2
ASHL ASHQ	78 79	оп3<оп2*(2**оп1) Арифметический сдвиг (L - формат длинного слова, Q - формат квадрослова)	* * * 0	Содержимое оп2 сдвигается на число разрядов, указанных в оп1. Результат запоминается по адресу оп3. При положительном оп1 - сдвиг влево, при отрицательном - вправо

Окончание табл.1.7

Мнемокод	КОП	Операция	Флаги N Z V C	Описание
CLRB CLRW CLRL CLRQ CLRO	94 B4 D4 7C 7CFD	оп1 < 0 Очистка слова	0 1 0 -	По адресу оп1 заносится 0
BISB2 BISW2 BISL2	88 A8 C8	оп2 < оп2 v оп1 Установка разрядов	* * 0 -	Логическое сложение оп1 и оп2, результат запоминается по адресу оп2
BICB2 BICW2 BICL2	8A AA CA	оп2 < оп2 & $\overline{\text{оп1}}$ Очистка разрядов	* * 0 -	Содержимое оп1 логически умножается на инвертированное содержимое оп1, результат запоминается по адресу оп2
XORB2 XORW2 XORL2	8C AC CC	оп2 < оп2(+)оп1 Исключающее ИЛИ	* * 0 -	Исключающее ИЛИ между содержимым оп1 и оп2 запоминается по адресу оп2
CMPB CMPW CMPL	91 B1 D1	оп1 - оп2 Сравнение	* * 0 *	Из содержимого оп1 вычитается содержимое оп2. Результат не запоминается. Используется для установки кодов условий, которые анализируются командами переходов
TSTB TSTW TSTL	95 B5 D5	оп1 < оп1 Опрос слова (Тестирование)	* * 0 0	В соответствии с содержимым оп1 устанавливаются коды условий
BITB BITW BITL	93 B3 D3	оп2 & оп1 Проверка разрядов	* * 0 -	Содержимое оп1 логически умножается на содержимое оп1. Результат не запоминается. Используется для установки кодов условий

Большинство команд, приведенных в табл.1.7, просты в использовании и не требуют дополнительных пояснений. Исключение составляют лишь команды сдвига и сложения (вычитания) с переносом, которые и рассмотрим ниже. Следует также отметить, что код операции CLRO занимает 2 байта (значение младшего байта - FD, старшего байта - 7C).

Команда сдвига

Команды ASHL и ASHQ выполняют арифметический сдвиг, в результате которого знак числа сохраняет свое положение. Команда ASHL используется для сдвига длинных слов, а команда ASHQ - для сдвига квадрослов. Команды имеют следующий формат:

```
ASHL    оп1,оп2,оп3
ASHQ    оп1,оп2,оп3
```

Первый операнд - оп1 - 8-разрядное число со знаком, определяющее количество разрядов, на которое следует произвести сдвиг и направление сдвига. Если оп1 > 0 , то производится сдвиг влево. Если оп1 < 0, то происходит сдвиг вправо. Фактически это аналогично умножению на 2 в степени n или на 2 в степени -n. Если оп1 = 0 - сдвиг не выполняется.

При выполнении команды ASHx содержимое оп2 сдвигается на число разрядов, указанных в оп1 и результат помещается по адресу оп3. Содержимое оп2 в результате выполнения команды сдвига не изменяется. Команды сдвига устанавливают биты N и Z в зависимости от значения результата. Бит V будет установлен, если при сдвиге влево получен результат, значение которого выходит за пределы диапазона представимых чисел (т.е. старшая значащая единица выходит за пределы разрядной сетки, что приводит к изменению знака числа). Так как эти инструкции предназначены для работы только с числами со знаком, бит C всегда сбрасывается.

Следующие примеры поясняют действие команды ASHL.

```
┌──────────┬──────────┬──────────┬──────────┐
│ 11111111 │ 11111111 │ 11110000 │ 11110000 │ Исходное содержимое R5
└──────────┴──────────┴──────────┴──────────┘
    ASHL  #5,R5,R5    -    сдвиг влево на 5 разрядов
┌──────────┬──────────┬──────────┬──────────┐
│ 11111111 │ 11111110 │ 00011110 │ 00000000 │ Новое содержимое R5
└──────────┴──────────┴──────────┴──────────┘
    ASHL  # -8,R5,R5   -   сдвиг вправо на 8 разрядов
┌──────────┬──────────┬──────────┬──────────┐
│ 11111111 │ 11111111 │ 11111110 │ 00011110 │ Новое содержимое R5
└──────────┴──────────┴──────────┴──────────┘
```

Команды сложения и вычитания с переносом

В ЭВМ семейства VAX бит C показывает наличие переноса при сложении или заема при вычитании младших частей больших чисел. Следовательно, для учета переноса (заема) необходимо прибавить (вычесть) значение бита C к (из) старшей части результата. Конечно, можно было бы проверить значение бита C, а затем увеличить (или уменьшить) результат на 1. Чтобы не выполнять последовательно эти действия, разработчики ЭВМ VAX ввели следующие команды: ADWC - сложить с переносом, SBWC - вычесть с заемом. Обе команды работают с операндами, имеющими формат длинных слов. Эти команды могут быть использованы для выполнения вычислений с двойной точностью, например, с 64-битовыми словами, совместно с другими арифметическими операциями.

Пусть числа два 64-разрядных числа A и B содержатся в длинных словах с адресами AL,AR,BL,BR, где буквы L и R обозначают левую и правую половину числа соотвественно. Сложение с двойной точностью осуществляется следующей последовательностью команд:

```
ADDL2  AR, BR
ADWC   AL, BL
```

Аналогично вычитание A из B производится командами:

```
SUBL2  AR, BR
SBWC   AL, BL
```

1.5.2 Команды пересылки (табл.1.8)

Команды пересылки — Таблица 1.8

Мнемоника	КОП	Операция	Флаги N Z V C	Описание
MOVB MOVW MOVL MOVQ MOVO	90 B0 D0 7D 7DFD	оп2 < оп1 Пересылка операндов различного формата	* * 0 -	Содержимое оп1 пересылается по адресу оп2

Код операции MOVO (переслать октаслово) состоит из двух байтов.

1.5.3 Команды безусловного перехода и обращения к подпрограммам

Описание команд этой группы приведено в табл.1.9. При их выполнении содержимое регистра слова состояния процессора PSW не изменяется.

Команды безусловного перехода и обращения к подпрограммам — Таблица 1.9

Мнемоника	КОП	Название и формат	Описание
BRB BRW	11 31	Безусловный переход BRB смещение BRW смещение	Переход по адресу, получаемому путем прибавления к содержимому PC смещения в формате байта или слова
JMP	17	Универсальный переход JMP оп1	В PC заносится адрес оп1
JSB	16	Переход к подпрограмме JSB оп1	Содержимое PC заносится в стек, затем в PC заносится адрес оп1
RSB	05	Возврат из подпрограммы RSB	В PC заносится длинное слово, выбранное из стека

Логика выполнения команд BRB и BRW рассмотрена в п.1.4.4. Поэтому, приведем лишь пример, поясняющий выполнение команды BRB.

Команда	Адрес
60 11	00000100

Команда записана в памяти, начиная с адреса ^X 00000100. Процессор считывает код операции BRB - ^X 11 из ячейки с адресом ^X 00000100, после чего считывает смещение ^X 60 из ячейки с адресом ^X 00000101. К этому моменту содержимое PC достигло ^X 00000102. Процессор прибавляет ^X 60 к ^X 00000102 и результат суммирования ^X 00000162 помещает в программный счетчик. Выборку следующей команды процессор будет осуществлять по адресу ^X 00000162.

Команды BRB и BRW имеют ограничения, связанные с тем, что смещение, указываемое в этих командах позволяет осуществлять переход лишь на определенное число байтов (в зависимости от разрядности смещения). Например, команда BRW может осуществлять переход на 32765 байтов назад и на 32770 байтов вперед от адреса команды BRW. Если необходимо осуществить переход на большее число байтов, то использовать данную команду нельзя.

Команда JMP не имеет таких ограничений, как рассмотренные выше команды, и позволяет осуществлять переход по всему адресному пространству. В отличие от других команд перехода она имеет формат, согласующийся с форматами остальных команд ЭВМ VAX. При этом в счетчик команд PC заносится адрес, определяемый в соответствии со способом адресации, указанным в спецификации операнда.

Рассмотрим следующий пример.

```
          Команда                    Адрес
00040000    EF  17                00004000
```

^X 17 - код операции JMP. Байт-спецификатор операнда ^X EF определяет относительную адресацию со смещением, заданным в формате длинного слова. К тому моменту, когда из памяти производится выборка смещения ^X 00040000, содержимое PC увеличивается до ^X 00004006. Таким образом, по этой команде будет выполняться переход по адресу ^X 00004006 плюс ^X 00040000, т.е. по адресу ^X 00044006.

В рассмотренном примере для передачи управления по данному адресу нельзя использовать команды BRB и BRW. Дополнительное преимущество команды JMP состоит в том, что она может применяться с большим числом режимов адресации, что позволяет легко организовать сложные управляющие конструкции.

Команда перехода к подпрограмме JSB по формату аналогична команде JMP. Для хранения адреса возврата в основную программу при обращении к подпрограмме служит специально отведенная область ОЗУ, называемая стек. Для обращения к стеку используется регистр R14 (SP) - указатель стека. При использовании этого регистра в других целях необходимо быть внимательным, так как это может привести к нарушению правильных связей в программе при очередном вызове подпрограммы или возврате из нее. По команде JSB значение указателя стека уменьшается на 4, в стеке сохраняется значение программного счетчика, после чего в PC заносится адрес, сформированный в соответствии с указанным режимом адресации. Выполнение команды JSB оп1 эквивалентно выполнению следующих команд:

```
MOVL   PC, -(SP)
JMP    оп1
```

Команда возврата из подпрограммы RSB не имеет операндов. По этой команде считывается адрес возврата из стека по адресу, указанному в регистре R14 и записывается в PC, после чего указатель стека увеличивается на 4. Команда RSB эквивалентна команде

```
MOVL   (SP)+, PC
```

1.5.4 Команды перехода по кодам условий

Для наглядности изложения команды перехода по кодам условий (табл.1.10) выделены отдельно от других команд перехода. При выполнении этих команд содержимое регистра PSW не изменяется.

Команды условного перехода BGEQ,BGTR,BLSS и BLEQ используются после сравнения чисел со знаком. Их не следует применять при работе с числами без знака. Например, если рассматривать содержимое длинных слов как число без знака, то число ^X FFFFFFFF больше числа ^X 00000000. Однако операция сравнения, предшествующая команде BGTR даст противоположный результат, поскольку число ^X FFFFFFFF будет рассматриваться как отрицательное, а именно как -1. Для решения этой проблемы предусмотрены команды BGEQU, BGTRU, BLSSU и BLEQU, применяемые для перехода после сравнения чисел без знака. Эти команды имеют смысл только тогда, когда они используются после операций сравнения.

Команды перехода по кодам условий — Таблица 1.10

Мнемоника	КОП	Название	Значение кодов условий в PSW для перехода
BNEQ	12	Переход, если не равно (со знаком)	Z = 0
BNEQU	12	Переход, если не равно (без знака)	Z = 0
BEQL	13	Переход, если равно (со знаком)	Z = 1
BEQLU	13	Переход, если равно (без знака)	Z = 1
BGTR	14	Переход, если больше (со знаком)	Z v [N(+)V] = 0
BLEQ	15	Переход, если меньше или равно (со знаком)	Z v [N(+)V] = 1
BGEQ	18	Переход, если больше или равно (со знаком)	N (+) V = 0
BLSS	19	Переход, если меньше (со знаком)	N (+) V = 1
BGTRU	1A	Переход, если больше (без знака)	C v Z = 0
BLEQU	1B	Переход, если меньше или равно (без знака)	C v Z = 1
BVC	1C	Переход, если нет переполнения	V = 0
BVS	1D	Переход, если есть переполнение	V = 1
BGEQU	1E	Переход, если больше или равно (без знака)	C = 0
BCC	1E	Переход, если нет переноса	C = 0
BLSSU	1F	Переход, если меньше (без знака)	C = 1
BCS	1F	Переход, если есть перенос	C = 1

Сравнение или проверка на равенство осуществляется одинаково для чисел со знаком и без знака. Однако для того, чтобы избавить программиста от необходимости помнить, когда надо использовать различные переходы для чисел со знаком и без знака, предусмотрены мнемонические обозначения BEQLU и BNEQU, для которых коды операций (КОП) совпадают с кодами операций BEQL и BNEQ. Это позволяет программисту быть последовательным и использовать символ U каждый раз, когда он имеет дело с числами без знака.

Для анализа переполнения при обработке чисел без знака (наличия переноса из старшего разряда результата при выполнении арифметической операции) служат команды BCS и BCC, а для анализа переполнения при выполнении операций над числами со знаком используются команды BVS и BVC.

Любопытно отметить, что команда BLSSU идентична команде BCS (код операции ^X 1F), а команда BGEQU - команде BCC (код операции ^X 1E).

1.5.5 Команды организации циклов (табл.1.11)

Команды организации циклов Таблица 1.11

Мнемоника	КОП	Название и формат	Описание
ACBB ACBW ACBL	9D 3D F1	Сложение, сравнение и переход ACB[B,W,L] оп1, оп2, оп3, смещение	Содержимое оп2 и оп3 складываются, сумма запоминается в оп3. Затем содержимое оп3 и оп1 сравнивается, и если выполняется одно из 2-х условий (условие1: (оп2) >= 0 и (оп3) <= (оп1) ; условие2: (оп2) < 0 и (оп3) >= (оп1)), то происходит переход по адресу, полученному путем прибавления смещения к содержимому PC

Команды ACBB,ACBW,ACBL включены в систему команд VAX для формирования циклов с проверкой условия их окончания после выполнения операций тела цикла. Эти команды эквивалентны оператору языков высокого уровня FOR...TO. Рассмотрим подробнее команду ACBL.

Символическое имя кода операции команды ACBL содержит в себе начальные буквы названий выполняемых операций (Add - приращение, Compare - сравнение, Branch - передача управления, а также указание на размер сравниваемых данных, Longword - длинные слова).

Обобщенный формат команды ACBL имеет следующий вид:

```
ACBL  предел, шаг, параметр_цикла, адрес_перехода.
```

Рассмотрим пример организации цикла для сложения чисел от 1 до 10 с использованием команды ACBL и без нее.

```
     CLRL   R2                          CLRL   R2
     MOVL   #10,R1                      MOVL   #10,R1
M1:  ADDL2  R1,R2                  M1:  ADDL2  R1,R2
     ACBL   #1,#-1,R1,M1                DECL   R1
                                        BNEQ   M1
```

Из примера можно сделать вывод, что использование команды ACBL позволяет организовать цикл с меньшим числом команд.

1.5.6 Команды специального назначения

Команды специального назначения Таблица 1.12

Мнемоника	КОП	Название и формат	Описание
HALT	00	Останов HALT	Производится останов выполнения программы
NOP	01 B9	Нет операции NOP	Не производится никакой операции
BICPSW		Очистка разрядов слова состояния процессора BICPSW оп1	Содержимое PSW логически умножается на инвертированное значение маски (содержимое оп1). Разряды 8-15 маски должны быть равны 0
BISPSW	B8	Установка разрядов слова состояния процессора BISPSW оп1	Содержимое PSW логически складывается с маской (содержимое оп1). Разряды 8-15 маски должны быть равны 0

Команды BICPSW и BISPSW позволяют присваивать значения 0 или 1 содержимому отдельных разрядов регистра PSW (разряды 0 - 3) или их комбинации. Эти команды действуют аналогично командам BICW2 и BISW2 соответственно. Каждая команда имеет только один операнд - маску длиной 16 бит. Так как пользователь в эмулирующей программе может использовать только разряды 0 - 3 регистра PSW, то рекомендуется в указанных командах неиспользуемые разряды маски (4 – 15 разряды) обнулять.

2. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Лабораторная работа V1

ПРЕДСТАВЛЕНИЕ ДАННЫХ В ЭВМ. СПОСОБЫ АДРЕСАЦИИ. ФОРМАТЫ КОМАНД. АРИФМЕТИКО-ЛОГИЧЕСКИЕ ОПЕРАЦИИ С ЦЕЛОЧИСЛЕННЫМИ ДАННЫМИ.

Цель работы: Изучение архитектуры процессора VAX-11, изучение форматов команд и данных процессора VAX-11, изучение системы арифметико-логических команд процессора VAX-11, изучение типов адресации процессора VAX-11. Выполнение простейших программ арифметико-логической обработки регистровых данных и данных из памяти с использованием различных способов адресации.

Методические указания

В данной работе изучается представление данных и форматы основных команд в ЭВМ типа VAX-11, а также осуществляется выполнение простейших программ арифметико-логической обработки данных с помощью эмулирующей программы.

Данные в эмулирующей программе могут храниться, и соответственно, загружаться и считываться:

а) в регистрах общего назначения (РОН),

б) в памяти по адресам [0-3FF].

Команды в программе могут храниться (соответственно, загружаться и считываться) в памяти по адресам [0-3FF].

При этом между командами и данными в хранении, записи и чтении нет принципиальной разницы.

При выполнении работы особое внимание необходимо обратить на то, что в ЭВМ типа VAX-11 машинные операции выполняются со словами различной длины: от одного до шестнадцати байт, а также на порядок построения и анализа команд. Здесь необходимо учитывать особенности организации адресации слов и байтов памяти, а также особенности шестнадцатеричного кодирования информации при работе со словами и байтами соответственно.

При выполнении команд форматы операндов и результата определяются исключительно кодом операции. Так если указана операция со словами двойной длины (типа L), а операнды были определены размером в один байт, то операция будет выполнена со словами двойной длины. При этом старшие разряды операндов определяются текущими значениями старших трех байтов длинных слов по адресам указанных операндов. Если указана операция с байтами, а операнды были определены как слова, то в операции будут участвовать только указанные байты. Если для указанных в заданиях арифметико-логических операций существует несколько машинных операций с данными различной длины, то необходимо выбирать тот код операции, который соответствует формату результата операции.

При составлени программы и трассировке ее алгоритма также необходимо учитывать особенности адресации операндов различной длины и порядок выполнения команд. В процессе выполнения команды сначала выбирается и анализи-

руется код операции, затем вычисляется адрес первого операнда (со всеми изменениями регистров) и выбирается первый операнд, потом вычисляется адрес второго операнда и т.д.

Практическая часть

Практическая часть работы включает выполнение следующих действий:

а) формирование числовых значений в соответствии с индивидуальным заданием, перевод их в шестнадцатеричную систему счисления и определения минимального формата представления исходных данных как целых чисел;

б) определения минимального формата и представление исходных данных как чисел с плавающей запятой (кроме X9);

в) запись целочисленных данных в РОН;

г) запись целочисленных данных в память по заданным адресам;

д) запись чисел с плавающей запятой в память по заданным адресам;

е) по заданному алгоритму составление и выполнение простейшей программы работы с целочисленными данными, хранящимися в РОН;

ж) по заданному алгоритму составление и выполнение простейшей программы работы с целочисленными данными, хранящимися в памяти, с использованием различных способов косвенной адресации;

з) по заданному алгоритму составление простейшей программы работы с целочисленными данными с использованием заданных способов адресации по смещению и через счетчик команд, причем непосредственная адресация должна быть по возможности заменена на литеральную.

Правильность разработки и выполнения программ арифметико-логической обработки данных контролируется путем ручной трассировки заданных алгоритмов с последующим сравнением результатов работы программ с результатами ручной трассировки.

Варианты заданий

```
Значения исходных данных определяются выражениями:
   X1 = [ (-1) ** ( NB + 0 ) ] * [ ( NB + NГ ) * 3 ]
   X2 = [ (-1) ** ( NB + 1 ) ] * ( NB + NГ + 17 )
   X3 = [ (-1) ** ( NB + 2 ) ] * [ ( NB + NГ + 29 ) ** 2 ]
   X4 = [ (-1) ** ( NB + 3 ) ] * [ ( NB + NГ + 23 ) ** 2 ]
   X5 = X3 ** 2
   X6 = (-1) * ( X4 ** 2 )
   X7 = (-1) * [ X5 * ( 2 ** 28 ) ]
   X8 = (-1) * [ X6 * ( 2 ** 20 ) ]
   X9 = [ X7 * ( 2 ** 52 ) ] - 12
 Здесь и далее: все числовые значения в исходных данных задаются
в десятичной системе счисления,
 NB - номер варианта, определяется как младшая цифра кода ASCII пер-
вой буквы фамилии,
 NГ - младшая цифра номера группы,
  ** - возведение в степень.
```

По п.в) размещение данных определяется в табл.2.1.

По п.г) адреса данных определяются выражениями:

```
Адр(X1) = ( NB * NГ )
Адр(X2) = ( NB * NГ ) + 10
Адр(X3) = ( NB * NГ ) + 20
Адр(X4) = ( NB * NГ ) + 30
Адр(X5) = ( NB * NГ ) + 40
Адр(X6) = ( NB * NГ ) + 50
Адр(X7) = ( NB * NГ ) + 60
Адр(X8) = ( NB * NГ ) + 70
Адр(X9) = ( NB * NГ ) + 80
```

По п.д) адреса данных определяются выражениями:

```
Адр(X1) = NB + 100
Адр(X2) = NB + 110
Адр(X3) = NB + 120
Адр(X4) = NB + 130
Адр(X5) = NB + 140
Адр(X6) = NB + 150
Адр(X7) = NB + 160
Адр(X8) = NB + 170
```

По п.е) начальный адрес размещения программы определяется выражением:

```
Адр = NB * 10 + 200
```

Варианты алгоритмов программ приведены на рис.2.1.

Варианты размещения операндов в РОН приведены в табл.2.1.

По п.ж) начальный адрес размещения программы определяется выражением:

```
Адр =  NB + NГ + 230
```

Варианты алгоритмов программ приведены на рис.2.2.

В табл.2.2 указаны типы используемой адресации для каждого операнда, где 6 - косвенная регистровая (простая косвенная) адресация, 8 - автоинкрементная (простая косвенная с автоувеличением), 7 - автодекрементная (простая косвенная с автоуменьшением) и 9 - косвенная автоинкрементная адресация (двойная косвенная с автоувеличением).

Промежуточные ячейки, используемые при реализации косвенной адресации, должны быть расположены с адреса:

```
Адр = ( NB * NГ ) + 250
```

РОН, используемые при адресации данных, выбираются произвольно.

По п.з) начальный адрес размещения программы определяется выражением:

```
Адр =  NB + NГ + 300
```

Варианты алгоритмов программ приведены на рис.2.3.

В табл.2.3 указаны типы используемой адресации для каждого операнда. Если в табл.3.3 явно не указан номер используемого регистра, то он выбирается произвольно.

Промежуточные ячейки, используемые при реализации косвенной адресации, должны быть расположены в памяти, начиная с адреса, определяемого выражением:

```
Адр = ( NB * NГ ) + 270
```

Таблица 2.1

Данные	Номера РОН по вариантам																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
X1	0	1	2	9	B	5	2	3	9	A	B	C	D	4	3	E	6	7	8	0
X2	1	2	3	8	A	4	3	4	A	B	C	D	E	5	5	0	7	8	9	6
X3	2	3	4	7	9	3	4	5	8	1	3	E	0	6	7	4	A	B	C	D
X4	3	4	A	6	8	2	5	8	7	2	4	0	1	7	9	8	B	C	D	E
X5	8	5	B	5	7	1	6	9	6	3	5	1	2	8	B	C	0	4	A	7
X6	9	6	C	4	6	0	7	A	5	4	9	2	3	9	C	1	8	D	E	B
X7	A	C	0	2	4	6	8	1	3	5	7	9	B	D	0	2	1	2	3	4

Типы адресации

Таблица 2.2

Номер оператора	Операнд	Варианты																	
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	ОП1	6	7	8	9	7	8	6	6	7	7	8	9	8	6	6	7	6	8
	ОП2	-	9	7	-	-	7	8	-	-	8	7	-	-	9	9	-	-	9
2	ОП1	8	9	6	7	7	6	9	6	6	7	9	8	7	8	9	6	7	6
	ОП2	9	6	9	8	9	9	7	9	7	9	7	9	9	9	8	7	8	8
3	ОП1	7	8	6	6	9	7	7	8	9	7	7	7	7	6	7	6	9	6
	ОП2	8	-	-	8	6	-	-	7	6	-	-	6	9	-	-	6	7	-
4	ОП1	6	8	7	9	6	8	8	9	7	9	6	7	6	7	7	9	8	9
	ОП2	7	6	8	8	8	7	6	7	8	8	8	6	8	8	8	8	9	7
	ОП3	8	7	8	7	6	9	9	8	8	6	6	8	7	6	9	8	8	7
5	ОП1	9	7	9	8	7	6	6	7	9	8	9	6	8	8	7	9	6	8
	ОП2	6	8	8	7	8	8	7	6	7	7	8	8	9	7	6	7	7	6

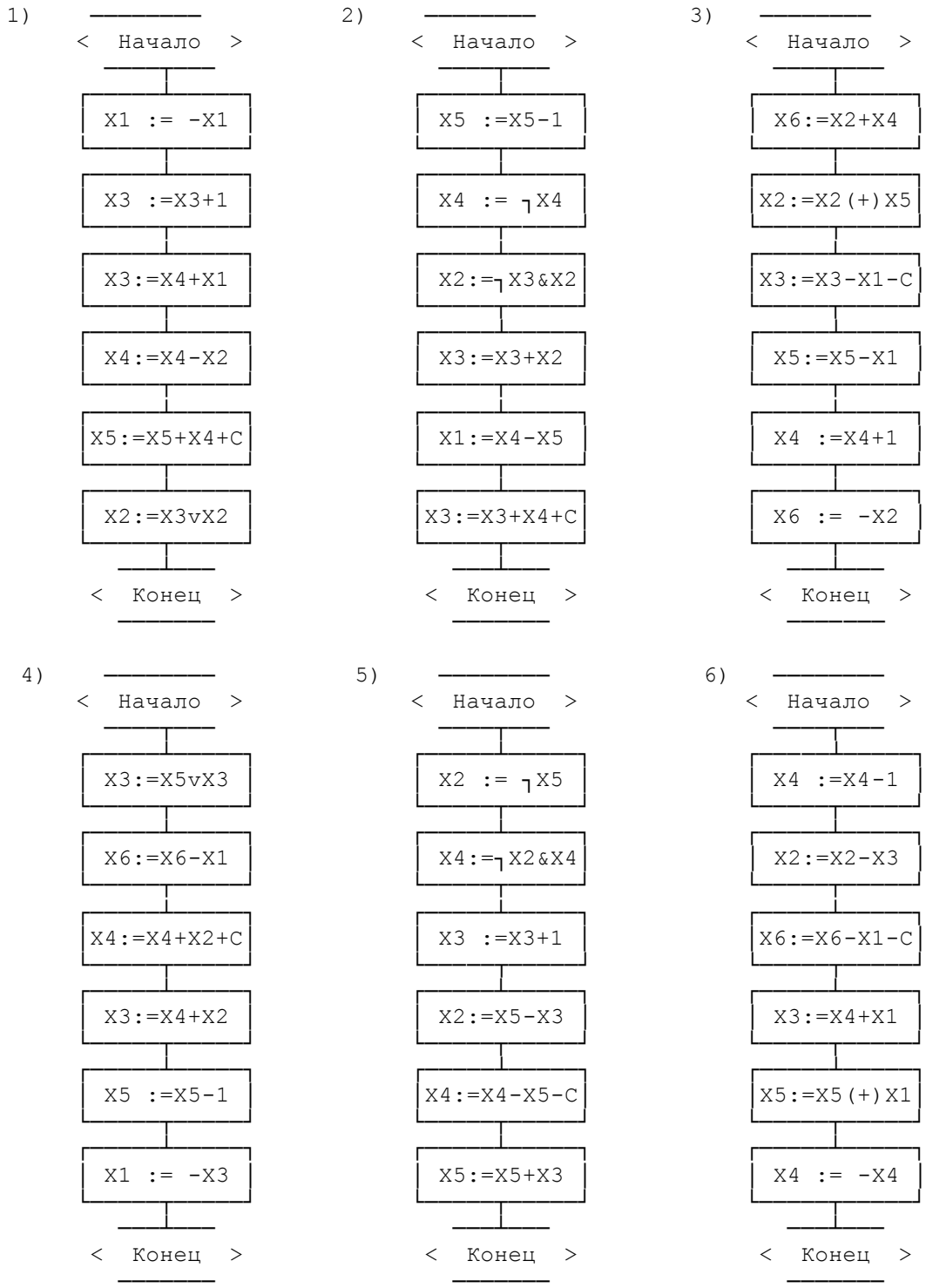

1)
Начало
X1 := -X1
X3 :=X3+1
X3:=X4+X1
X4:=X4-X2
X5:=X5+X4+C
X2:=X3vX2
Конец
2)
Начало
X5 :=X5-1
X4 := ┐X4
X2:=┐X3&X2
X3:=X3+X2
X1:=X4-X5
X3:=X3+X4+C
Конец
3)
Начало
X6:=X2+X4
X2:=X2(+)X5
X3:=X3-X1-C
X5:=X5-X1
X4 :=X4+1
X6 := -X2
Конец
4)
Начало
X3:=X5vX3
X6:=X6-X1
X4:=X4+X2+C
X3:=X4+X2
X5 :=X5-1
X1 := -X3
Конец
5)
Начало
X2 := ┐X5
X4:=┐X2&X4
X3 :=X3+1
X2:=X5-X3
X4:=X4-X5-C
X5:=X5+X3
Конец
6)
Начало
X4 :=X4-1
X2:=X2-X3
X6:=X6-X1-C
X3:=X4+X1
X5:=X5(+)X1
X4 := -X4
Конец

Рис.2.1

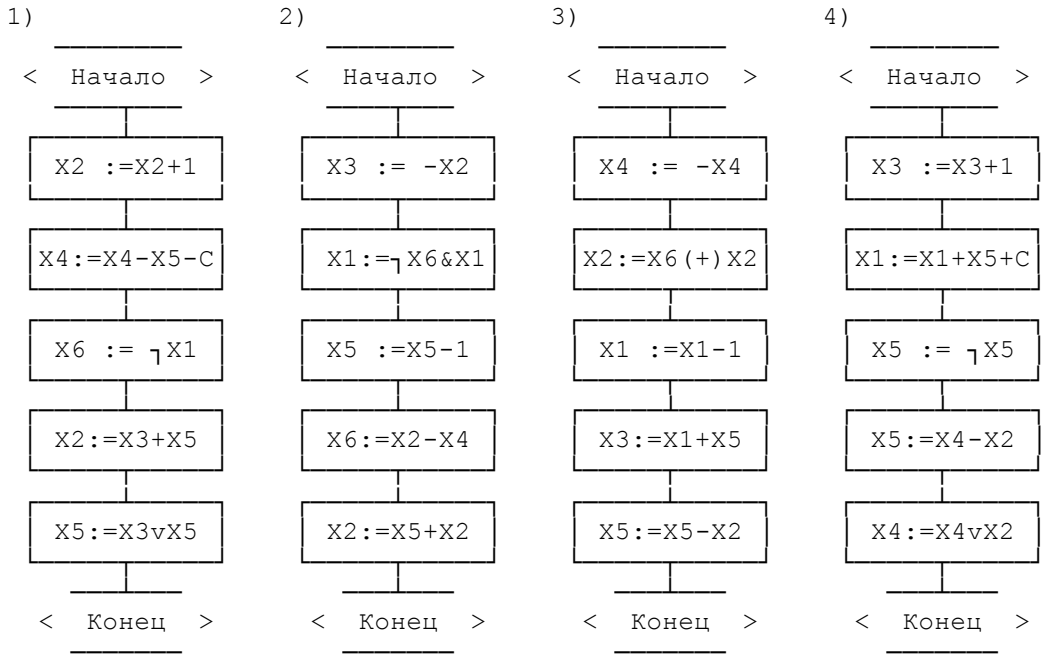
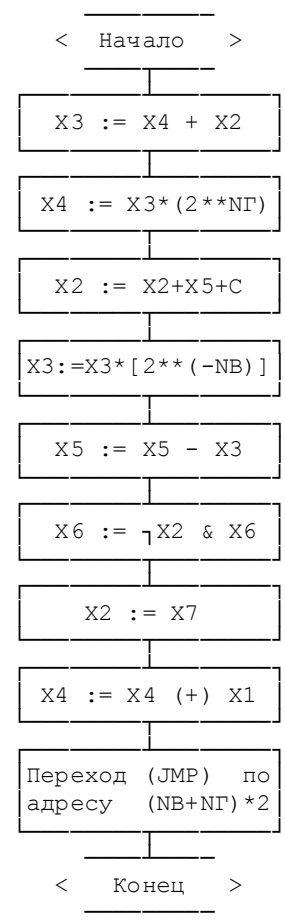


Рис.2.2

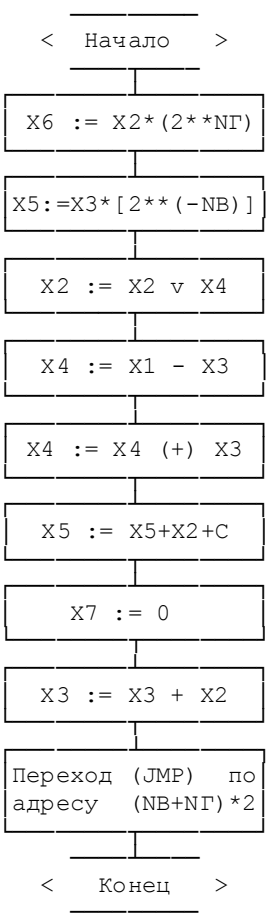
Типы адресации Таблица 2.3

Оператор	Операнд	Варианты																		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	ОП1	Ax	8F	8F	8F	BF	8F	8F	AF	Dx	8F	8F	Fx	CF	8F	8F	Bx	Ex	8F	8F
	ОП2	BF	Cx	8F	AF	Dx	EF	AF	Fx	EF	AF	Bx	8F	8F	Dx	9F	9F	AF	AF	Dx
	ОП3	Dx	EF	AF	Dx	Cx	AF	Bx	Ax	Fx	Dx	9F	Bx	Ex	AF	Dx	CF	9F	9F	BF
2	ОП1	8F	8F	Dx	8F	8F	8F	8F	8F	8F	8F	Dx	8F	8F	8F	Ax	8F	8F	8F	CF
	ОП2	Cx	AF	Bx	Fx	EF	Dx	9F	Bx	9F	Bx	8F	9F	Ax	9F	BF	DF	Dx	Fx	DF
	ОП3	EF	Dx	Fx	Ax	9F	Bx	Dx	9F	Bx	9F	Ax	CF	DF	Fx	CF	Ex	Cx	Bx	EF
3	ОП1	Fx	8F	9F	Bx	Bx	9F	Ax	CF	8F	Fx	BF	DF	Cx	8F	DF	8F	EF	DF	Ax
	ОП2	9F	Bx	Ax	9F	CF	Fx	BF	DF	Cx	Ax	CF	Ex	BF	Bx	FF	FF	Fx	Ex	FF
4	ОП1	8F	Fx	8F	8F	8F	Ax	8F	8F	8F	BF	8F	8F	8F	CF	8F	8F	8F	8F	8F
	ОП2	AF	9F	BF	CF	DF	8F	CF	Ex	CF	CF	DF	FF	Fx	DF	EF	EF	Bx	Ax	Ex
	ОП3	Bx	Ax	CF	DF	Ex	BF	DF	FF	9F	DF	FF	EF	Dx	Ex	Ax	Cx	DF	Bx	Cx
5	ОП1	CF	BF	DF	Ex	AF	CF	Ex	8F	Ax	8F	EF	Cx	Bx	Ax	Ex	9F	9F	FF	FF
	ОП2	DF	CF	--	FF	FF	DF	--	EF	BF	Ex	--	9F	AF	Bx	--	Ax	CF	9F	--
6	ОП1	9F	DF	Ex	EF	Ax	Ex	FF	Cx	Cx	Bx	Ex	Ax	EF	FF	8F	BF	FF	EF	Cx
	ОП2	Ex	Ex	FF	Cx	BF	Bx	EF	9F	Ex	Cx	Ax	BF	9F	9F	FF	EF	Ax	Cx	BF
7	ОП1	FF	Bx	EF	9F	8F	Cx	Cx	Ax	Fx	9F	FF	EF	Cx	EF	BF	Cx	8F	Bx	9F
	ОП2	Ax	--	Cx	Ax	Cx	--	9F	BF	DF	--	BF	Cx	FF	--	Cx	9F	BF	--	Ax
8	ОП1	8F	FF	9F	BF	Ex	9F	Ax	EF	AF	FF	Cx	9F	Bx	Cx	9F	Ax	CF	Dx	8F
	ОП2	BF	Cx	Ax	EF	9F	FF	Ex	Cx	Dx	EF	9F	Ax	CF	Bx	Ax	EF	Ax	9F	Fx
9	ОП1	9F	Ax	Bx	Cx	Dx	Ex	Fx	AF	BF	CF	DF	EF	FF	9F	Dx	DF	Cx	Fx	AF

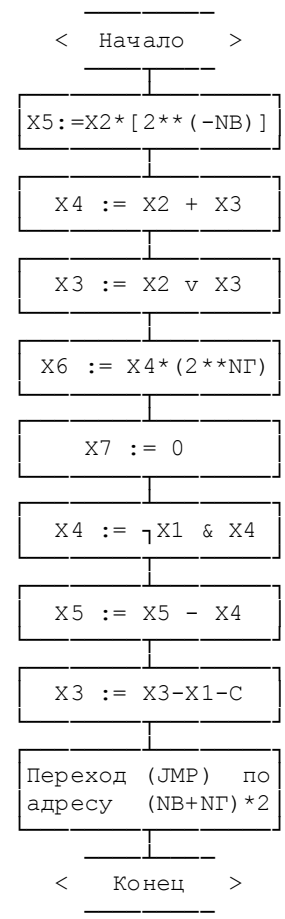
1)


Начало
X3 := X4 + X2
X4 := X3*(2**NГ)
X2 := X2+X5+C
X3:=X3*[2**(-NB)]
X5 := X5 - X3
X6 := ┐X2 & X6
X2 := X7
X4 := X4 (+) X1
Переход (JMP) по адресу (NB+NГ)*2
Конец

2)


Начало
X6 := X2*(2**NГ)
X5:=X3*[2**(-NB)]
X2 := X2 v X4
X4 := X1 - X3
X4 := X4 (+) X3
X5 := X5+X2+C
X7 := 0
X3 := X3 + X2
Переход (JMP) по адресу (NB+NГ)*2
Конец

3)


Начало
X5:=X2*[2**(-NB)]
X4 := X2 + X3
X3 := X2 v X3
X6 := X4*(2**NГ)
X7 := 0
X4 := ┐X1 & X4
X5 := X5 - X4
X3 := X3-X1-C
Переход (JMP) по адресу (NB+NГ)*2
Конец

4)

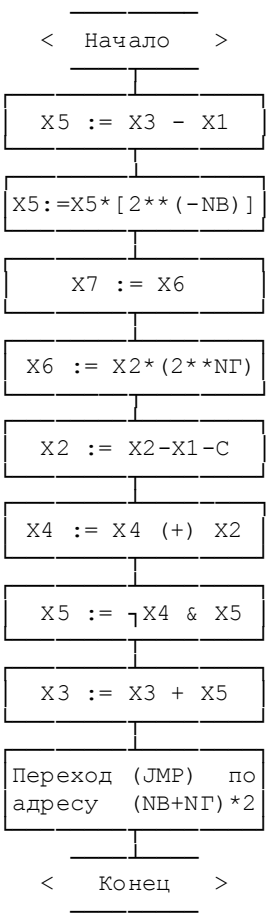

Начало
X5 := X3 - X1
X5:=X5*[2**(-NB)]
X7 := X6
X6 := X2*(2**NГ)
X2 := X2-X1-C
X4 := X4 (+) X2
X5 := ┐X4 & X5
X3 := X3 + X5
Переход (JMP) по адресу (NB+NГ)*2
Конец

Рис.2.3

Порядок выполнения работы

1. Выбрать исходные данные в соответствии с номером варианта.
Например, для варианта (NB = 33 , NГ = 4) исходные данные имеют следующий вид:

```
X1 = [ (-1) ** ( 29 + 0 ) ] * [ ( 33 + 4 ) * 3 ] =
   = [ (-1) ** 29 ] * [ 37 * 3 ] = - 111
X2 = [ (-1) ** 30 ] * ( 37 + 17 ) = 54
X3 = [ (-1) ** 31 ] * [ ( 37 + 29 ) ** 2 ] = - 4356
X4 = [ (-1) ** 32 ] * [ ( 37 + 23 ) ** 2 ] = 3600
   и т.д.
```

2. По п.п.а)-б) перевести значения величин X1-X9 и адреса их размещения в шестнадцатеричную систеиму счисления.
Например, для варианта (NB = 33 , NГ = 4) исходные данные имеют следующий вид:

```
X1 = - 111 = 91
X2 = 54 = 36
X3 = - 4356 = EEFC
X4 = 3600 = 0E10
   и т.д.
```

3. Составить карту распределения памяти под команды и данные.

4. По п.е) составить программу заданного алгоритма в мнемонических и машинных кодах.

5. Произвести трассировку заданного в п.е) алгоритма с использованием заданных исходных данных.

6. По п.ж) составить программу заданного алгоритма в мнемонических и машинных кодах.

7. Произвести трассировку заданного в п.ж) алгоритма с использованием заданных исходных данных, при этом в таблице трассировки должны быть отражены также и значения информмации в ячейках памяти и регистрах, используемых для реализации различных способов адресации.

8. По п.з) составить программу заданного алгоритма в мнемонических и машинных кодах.

9. Произвести трассировку заданного в п.з) алгоритма с использованием заданных исходных данных, при этом в таблице трассировки должныбыть отражены также и значения информмации в ячейках памяти и регистрах, используемых для реализации различных способов адресации.

10. Осуществить запись данных по п.п. в)-д) задания.

11. Записать в память программы, составленные в соответствии с п.е)-з) задания.

12. Записать в память промежуточную информацию, необходимую для реализации различных способов адресации.

13. Выполнить программы, занесенные в память.

14. Проверить результаты выполнения программ, сравнивая их с результатами ручной трассировки алгоритмов.

15. Оформить отчет по лабораторной работе работе.

Содержание отчета

1. Титульный лист.
2. Текст задания.
3. Перевод исходных данных в шестнадцатеричную систему счисления.
4. Схемы алгоритмов программ.
5. Тексты программ в мнемонических и машинных кодах.
6. Карта распределения памяти под команды и данные.
7. Таблицы трассировки программ.

Пример оформления карты памяти приведен в табл.2.4, текста программы - в табл.2.5, а в табл.2.6 - структура таблицы трассировки.

Карта распределения памяти Таблица 2.4

Число	Десятичное значение	Шестнадцатиричный код	Адрес загрузки
X1	- 111	91	RD
X2	54	36	RE
X3	- 4356	EEFC	R0
...	. . . .	. . . . .	...
X3	- 4356	EEFC	98
X4	3600	0E10	A2
X5	18974736	01218810	AC
...	. . . .	. . . . .	...
		Текст программы 1	3DE
		Текст программы 2	2BF

Текст программы Таблица 2.5

Оператор	Адрес	Шестнадцатиричный код	Мнемокод	Комментарии
1	3DE	52 D6	INCL R2	X5 :=X5+1
2	3E0	51 51 B2	MCOML R1,R1	X4 := ┐X4
3	3E3	5E 50 8A	BICB2 R0,RE	X2 :=┐X3&X2
...	...	. . . . .	. . . . .	. . . . .
		00	HALT	ОСТАНОВ

Таблица трассировки Таблица 2.6

Номер шага	Номер регистра	Расчетные значения: До выполнения команды	Расчетные значения: После выполнения команды	Значения, полученные в лаборатории	Адрес ячейки	Расчетные значения: До выполнения команды	Расчетные значения: После выполнения команды	Значения, полученные в лаборатории

Лабораторная работа V2

УСЛОВНЫЕ И БЕЗУСЛОВНЫЕ ПЕРЕХОДЫ. ОБРАБОТКА МАССИВОВ.

Цель работы: изучение особенностей адресации в командах условного и безусловного переходов, изучение адресации с индексированием, изучение организации циклических вычислений, выполнение программы арифметико-логической обработки массивов данных, хранящихся в памяти.

Методические указания

В данной лабораторной работе исследуются команды условного и безусловного перехода, а также возможности ЭВМ типа VAX-11 по организации циклических вычислений и обработке массивов.

В архитектуре VAX-11 используются две независимые системы адресации: одна используется в командах преобразования-тестирования данных, а также в команде безусловного перехода JMP и команде обращения к подпрограмме JSB, а другая - в командах условного перехода, в команде безусловного перехода BR и командах организации циклов типа команды ACB. Введение для команд перехода отдельной системы адресации связано с тем, что чаще всего переходы делаются внутри тела программы. Для того, чтобы программа была свободно перемещаемая в памяти, необходимо адреса перехода определять как смещения относительно некоторого адреса в программе. Естественным указателем на программу является счетчик команд (R15), поэтому адрес перехода лучше всего определять как относительный по счетчику команд. Тогда, если адреса перехода определяются как относительные по счетчику команд, способ адресации и номер регистра-указателя определяются по коду операции (по умолчанию), а в команде указывается только смещение.

В командах условного перехода величина смещения определяется в формате байта, и в этих командах за байтом кода операции следует байт, в котором указывается смещение от начала следующей команды до начала той команды, на которую делается переход. Величина этого смещения лежит в интервале от +127(10) до -128(10). В командах безусловного перехода типа BR и обращения к подпрограмме типа BSB величина смещения определяется или в формате байта (команды BRB и BSBB), или в формате слова (команды BRW и BSBW). В командах организации цикла типа команды ACB величина смещения всегда определяется в формате слова (W).

В тех случаях, когда адрес перехода отличается от текущего значения счетчика команд PC на величину, превышающую 65536 (65K) байт, используются команды JMP и JSB. В этих командах адрес перехода определяется по полной схеме в соответствии с указанным в байте-спецификаторе способом адресации и значением в регистре-указателе. При этом, если по способу адресации требуется определение формата данных (например, при модификации указателя или при адресации с индексированием), то, так как команды представляются в побайтовой форме, принимается, что данные представлены в байтовом формате. При выполнении вычислений на массивах для выборки элементов массива (ов) как правило используются или различные варианты косвенной адресации, или адресации с индексированием. (Заметим, что адресация с индексированием может рассматриваться как один из вариантов адресации по смещению.

При адресации по смещению одно из слагаемых исполнительного адреса является постоянной величиной, не изменяемой в процессе выполнения программы; а при адресации с индексированием оба слагаемых исполнительного адреса могут модифицироваться в ходе выполнения программы. Кроме того в адресации с индексированием дополнительно учитывается формат данных.)

При использовании косвенной адресации в регистре находится адрес элемента массива (т.е. содержимое регистра как бы указывает на нужное слово, и поэтому довольно-таки часто, как содержимое регистра, так и сам регистр, содержащий адрес слова, называют "указателем"). Если к каждому элементу массива происходит однократное обращение, то используется косвенная адресация с модификацией указателя (инкрементная или декрементная адресация). При этом значение указателя автоматически модифицируется на длину слова, т.е. указатель переустанавливается на следующий элемент массива. При многократных обращениях к каждому злементу массива цепочка последовательных обращений к одному элементу делается с использованием простой косвенной адресации и только последнее обращение (при прямом переборе злементов массива) или первое обращение (при обратном переборе) делается с модификацией указателя.

При использовании адресации с индексированием в индексном регистре, как правило, но не обязательно, находится номер текущего элемента массива, т.е. индексный регистр одновременно может использоваться как счетчик элементов (длины) массива. При этом переход к следующему элементу массива происходит при изменении счетчика элементов массива. Таким образом, при любом числе обращений к одному элементу массива не нужно заботиться об определении исполнительного адреса следующего элемента массива.

Окончание массива определяется или по достижении предельного значения счетчика (инкрементного или декрементного), или по достижении одного из граничных адресов массива.

Для выполнения циклических вычислений в ЭВМ VAX-11 предусмотрены специальные команды организации циклов. По каждой такой команде в машине выполняется сразу несколько действий: во-первых, модифицируется некоторое значение (например, некоторый счетчик или адрес элемента), затем это модифицированное значение сравнивается с контрольным значеним (например, предельным значением счетчика или граничным адресом) и по результатам этого сравнения выполняется условный переход. Самой универсальной из этой группы команд является команда ACB.

Практическая часть.

Практическая часть работы включает выполнение следующих действий:

а) в соответствии с индивидуальным заданием составление двух программ обработки массивов, содержащих не менее 10 целых чисел; одна программа для обращения к элементам массивов должна использовать косвенные способы адресации, а другая - адресацию с индексированием; во второй программе для организации цикла необходимо использовать команду ACB, а в первой программе использование команды организации цикла ACB запрещено;

б) формирование и занесение в память исходных значений массивов, определение и занесение в память и РОНы необходимых вспомогательных данных;

в) запись программ обработки массивов данных, хранящихся в памяти;

г) выполнение программ;

д) контроль результатов работы программ.

Правильность разработки и выполнения программ арифметико-логической обработки данных контролируется путем ручной трассировки заданных алгоритмов с последующим сравнением результатов работы программ с результатами ручной трассировки.

Варианты заданий.

Тексты заданий приведены в табл.2.7.

Допустимые способы проверки конца массива (организации цикла) приведены в табл.2.8.

Способы адресации, используемые для вычисления базового адреса при адресации с индексированием, приведены в табл.2.9.

Размеры элементов массива определены в табл.2.10..

Исходные данные должны располагаться в памяти, начиная с адреса, определяемого выражением:

```
Адр = NB * NГ
```

Начальный адрес размещения программ определяется выражением:

```
Адр = ( NB * NГ ) + 60
```

Вспомогательные значения, необходимые для выполнения программ, должны быть расположены в памяти, начиная с адреса, определяемого выражением:

```
Адр = ( NB * NГ ) + 300
```

Порядок выполнения работы.

А. В процессе самостоятельной работы

1. Выбрать исходные данные в соответствии с номером варианта.
2. Составить алгоритмы программ для решения поставленной задачи.
3. Составить программы вычислений в мнемонических и машинных кодах.
4. Составить карту распределения памяти под команды и данные.
5. Произвести ручную трассировку программ с использованием заданных исходных данных, при этом в таблице трассировки должны быть отражены значения информации в ячейках памяти и используемых регистрах.
6. Оформить отчет по лабораторной работе.

Б. В учебной лаборатории

1. Записать в память и РОН исходные данные, программы и дополнительную информацию, необходимую для реализации заданных алгоритмов и различных способов адресации.
2. Выполнить программы, занесенные в память.
3. Проверить результаты выполнения программ, сравнивая их с результатами ручной трассировки алгоритма.

Содержание отчета

1. Титульный лист.
2. Текст задания.
3. Перевод исходных данных в шестнадцатеричную систему счисления.
4. Схемы алгоритмов программ.
5. Тексты программ в мнемонических и машинных кодах.
6. Карта распределения памяти под команды и данные.
7. Таблицы трассировки программ.

Реализуемые алгоритмы — Таблица 2.7

Номер	Задание
1	Найти максимальный элемент массива (его номер и значение)
2	Найти минимальный элемент массива (его номер и значение)
3	Заменить элементы массива: X(i) := 0 , если X(i) < 0 X(i) := FF , если X(i) = 0
4	Найти сумму положительных элементов массива
5	Найти сумму отрицательных элементов массива
6	Найти логическую сумму элементов массива , содержащих 0 в младшем разряде
7	Найти логическое произведение элементов массива, содержащих 1 в старшем разряде
8	Найти сумму модулей элементов массива
9	Осуществить сдвиг влево положительных элементов массива на 4 разряда
10	Найти арифметическую сумму элементов массива, имеющих четный номер
11	Построить ряд из чисел Фибоначчи { F(1) = F(2) = 1 ; F(i+1) = F(i) + F(I-1) }
12	Заменить элементы массива: X(i) := -X(i) , если X(i) - четное X(i) := 0 , если X(i) - нечетное
13	Найти арифметическую сумму элементов массива, значения которых лежат в интервале -10 < X(i) < 20
14	Найти логическую сумму элементов массива, по абсолютной величине не превышающих 40
15	Найти минимальный положительный элемент массива (его номер и значение)
16	Найти элемент массива, имеющий максимальное абсолютное значение (его номер и значение)
17	Найти количество элементов массива, имеющих отрицательное значение и четный номер
18	Найти отрицательный элемент массива, имеющий максимальное абсолютное значение (его номер и значение)
19	Найти количество элементов массива, значения которых лежат в интервале -20 < X(i) < 50
20	Найти количество положительных, нулевых и отрицательных элементов массива

Варианты организации цикла Таблица 2.8

Номер	Способ проверки конца массива
1	По инкрементному счетчику (по достижению максимального значения номера массива)
2	По декрементному счетчику (по достижению минимального значения номера массива)
3	По достижению максимального адреса в массиве
4	По достижению минимального адреса в массиве

Варианты вычисления базового адреса массива Таблица 2.9

Вариант	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Способ адресации	6x	9F	Ax	Bx	Cx	Dx	Ex	Fx	AF	BF	CF	DF	EF	FF

Формат элементов массива Таблица 2.10

Вариант	1	2	3	4	5
Элемент массива	Байт (B)	Слово (W)	Двойное слово (L)	Слово (W)	Двойное слово (L)

Лабораторная работа V3

ОРГАНИЗАЦИЯ РАБОТЫ С ПОДПРОГРАММАМИ "ДЛИННЫЕ" АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ

Цель работы: изучение возможностей ЭВМ типа VAX-11 по организации работы с подпрограммами, отработка алгоритмов выполнения "длинных" арифметических операций (умножение, деление).

Методические указания

При составлении программы выполнения "длинной" арифметической операции заданный фрагмент повторяющейся части вычислений необходимо выделить в отдельную подпрограмму.

Обращение к подпрограмме осуществляется с помощью команды JSB, действие которой можно представить в виде эквивалентной последовательности команд:

```
MOVL PC,-(SP)            ; запоминание в стеке адреса возврата
JMP  адрес подпрограммы  ; переход на подпрограмму.
```

Возврат из подпрограммы происходит по команде RSB, действие которой можно представить следующей эквивалентной командой:

```
MOVL (SP)+,PC            ; восстановление в счетчике команд
                         ; адреса вызывающей программы.
```

При обращении к подпрограммам пользователя для запоминания адреса возврата используется основной пользовательский стек, расположенный в пользовательской области оперативной памяти. Указателем этого стека всегда служит регистр R14. Поэтому в прикладных программах использовать R14 в любом другом качестве не рекомендуется.

При выполнении лабораторной работы необходимо выделить часть памяти для организации стека и определить значение R14, учитывая механизм обращения к подпрограммам.

Практическая часть.

Практическая часть работы включает выполнение следующих действий:

а) в соответствии с индивидуальным заданием разработка алгоритма выполнения "длинной" арифметической операции (умножения или деления);

б) в соответствии с разработанным алгоритмом составление программы выполнения "длинной" арифметической операции, причем заданный фрагмент алгоритма должен быть оформлен как подпрограмма;

в) формирование и занесение в память и РОНы исходных значений и необходимых вспомогательных данных;

г) запись в память программы выполнения "длинной" арифметической операции;

д) выполнение программы;

е) контроль результатов работы программы.

Правильность разработки и выполнения программы выполнения "длинной" арифметической операции контролируется путем ручной трассировки контрольных примеров с последующим сравнением результатов работы программы с результатами ручной трассировки.

Варианты заданий.

Варианты индивидуальных заданий приведены в табл.2.11 - табл.2.13. Во всех вариантах операнды должны быть представлены в формате длинного слова L (32 разряда).

Программируемая "длинная" операция Таблица 3.11

Вариант	Операция
1	Умножение целых чисел со знаком в дополнительном коде с неподвижной суммой частичных произведений, сдвигом множимого вправо и анализом множителя, начиная со старших разрядов
2	Умножение целых чисел со знаком в дополнительном коде с неподвижной суммой частичных произведений, сдвигом множимого влево и анализом множителя, начиная с младших разрядов
3	Умножение целых чисел со знаком в дополнительном коде со сдвигом суммы частичных произведений влево, неподвижным множимым и анализом множителя, начиная со старших разрядов
4	Умножение целых чисел со знаком в дополнительном коде со сдвигом суммы частичных произведений вправо, неподвижным множимым и анализом множителя, начиная с младших разрядов
5	Деление целых чисел без знака для получения целого числа с восстановлением остатка с неподвижным делимым и сдвигом делителя
6	Деление целых чисел без знака для получения целого числа без восстановления остатка с неподвижным делимым и сдвигом делителя
7	Деление целых чисел без знака для получения целого числа с восстановлением остатка с неподвижным делителем и сдвигом делимого
8	Деление целых чисел без знака для получения целого числа без восстановления остатка с неподвижным делителем и сдвигом делимого
9	Деление чисел с фиксированной запятой без знака для получения числа с фиксированной запятой с восстановлением остатка с неподвижным делимым и сдвигом делителя
10	Деление чисел с фиксированной запятой без знака для получения числа с фиксированной запятой без восстановления остатка с неподвижным делимым и сдвигом делителя
11	Деление чисел с фиксированной запятой без знака для получения числа с фиксированной запятой с восстановлением остатка с неподвижным делителем и сдвигом делимого
12	Деление чисел с фиксированной запятой без знака для получения числа с фиксированной запятой без восстановления остатка с неподвижным делителем и сдвигом делимого
13	Деление числа с фиксированной запятой без знака на целое число без знака с восстановлением остатка с неподвижным делимым и сдвигом делителя

Окончание табл. 3.11

Вариант	Операция
14	Деление числа с фиксированной запятой без знака на целое число без знака без восстановления остатка с неподвижным делимым и сдвигом делителя
15	Деление числа с фиксированной запятой без знака на целое число без знака с восстановлением остатка с неподвижным делителем и сдвигом делимого
16	Деление числа с фиксированной запятой без знака на целое число без знака без восстановления остатка с неподвижным делителем и сдвигом делимого
17	Умножение чисел с фиксированной запятой со знаком в дополнительном коде с неподвижной суммой частичных произведений, сдвигом множимого вправо и анализом множителя, начиная со старших разрядов
18	Умножение чисел с фиксированной запятой со знаком в дополнительном коде с неподвижной суммой частичных произведений, сдвигом множимого влево и анализом множителя, начиная с младших разрядов
19	Умножение чисел с фиксированной запятой со знаком в дополнительном коде со сдвигом суммы частичных произведений влево, неподвижным множимым и анализом множителя, начиная со старших разрядов
20	Умножение чисел с фиксированной запятой со знаком в дополнительном коде со сдвигом суммы частичных произведений вправо, неподвижным множимым и анализом множителя, начиная

Таблица 3.12

Вариант	Фрагмент алгоритма, оформляемый как подпрограмма
1	Анализ очередного разряда множителя
2	Сдвиг множимого
3	Анализ очередного разряда множителя
4	Формирование и сдвиг суммы частичных произведений
5	Сдвиг делителя
6	Формирование очередного разряда частного
7	Формирование остатка (со сдвигом)
8	Формирование очередного разряда частного
9	Формирование очередного разряда частного
10	Формирование остатка (нового значения делимого)
11	Формирование очередного разряда частного
12	Формирование очередного разряда частного
13	Формирование остатка (нового значения делимого)
14	Сдвиг делителя
15	Сравнение остатка и делителя
16	Формирование остатка (со сдвигом)
17	Сдвиг множимого
18	Анализ очередного разряда множителя
19	Формирование и сдвиг суммы частичных произведений
20	Анализ очередного разряда множителя

Размещение операндов и результата операции Таблица 3.13

Вариант	Размещение операндов и результата операции		
	1 операнд	2 операнд	Результат
1	В РОН	В РОН	В РОН
2	В РОН	В РОН	В памяти
3	В РОН	В памяти	В РОН
4	В РОН	В памяти	В памяти
5	В памяти	В РОН	В РОН
6	В памяти	В РОН	В памяти
7	В памяти	В памяти	В РОН
8	В памяти	В памяти	В памяти

Порядок выполнения работы.

А. В процессе самостоятельной работы

1. Выбрать исходные данные в соответствии с номером варианта.
2. Составить алгоритм программы для решения поставленной задачи.
3. Составить программу вычислений в мнемонических и машинных кодах.
4. Составить карту распределения памяти под команды и данные.
5. Произвести ручную трассировку программы с использованием самостоятельно выбранных исходных данных, при этом в таблице трассировки должны быть отражены значения информмации в ячейках памяти и используемых регистрах, а также значения в регистре слова состояния PSW.
6. Оформить отчет по лабораторной работе.

Б. В учебной лаборатории

1. Записать в регистры и в память исходные данные, программу и дополнительную информацию, необходимую для реализации заданного алгоритма.
2. Выполнить программу, занесенную в память.
3. Проверить результаты выполнения программы, сравнивая их с результатами ручной трассировки алгоритма.

Содержание отчета

1. Титульный лист.
2. Текст задания.
3. Перевод исходных данных в шестнадцатеричную систему счисления.
4. Схема алгоритма программы.
5. Текст программы в мнемонических и машинных кодах.
6. Карта распределения памяти под команды и данные.
7. Таблица трассировки программы.

Лабораторная работа V4

ПРОГРАММИРОВАНИЕ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЙ НАД ЧИСЛАМИ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

Цель работы: отработка алгоритмов выполнения арифметических операций над числами с плавающей запятой.

Методические указания

Выполнение операций над числами с плавающей запятой может быть разбито на несколько этапов:

- выделение из машинной записи числа с плавающей запятой порядка и мантиссы с учетом скрытого бита;
- сравнение и выравнивание порядков с денормализацией мантиссы числа с меньшим порядком, если выполняется операция сложения или вычитания; сложение или вычитание порядков, если выполняется операция умножения или деления;
- выполнение операции над мантиссами;
- нормализация результата с коррекцией порядка;
- формирование числа со скрытым битом для записи в память.

При выполнении операций над числами с плавающей запятой порядки, как правило, остаются в формате характеристик, т.е. смещенными. При этом, когда выполняется сравнение характеоистик, смещения взаимно сокращаются и в результате остается истинная разность порядков, а когда выполняется сложение или вычитание характеристик, полученная сумма или разность корректируется на величину смещения.

Операции над мантиссами, как правило, выполняются в прямых кодах. При этом знаки операндов анализируются отдельно. По результатам этого анализа определяется знак произведения или частного или тип операции: сложение или вычитание.

В операциях над числами с плавающей запятой может возникнуть переполнение, когда порядок результата операции больше максимального, или антипереполнение или исчезновение порядка, когда порядок результата операции меньше минимального. При сложении (или вычитании) чисел с плавающей запятой переполнение или антипереполнение может возникнуть только в ходе нормализации суммы (или разности) мантисс. При умножения (или деления) чисел с плавающей запятой переполнение или антипереполнение выявляется при сложении (или вычитании) порядков. Причем в ходе нормализации мантиссы произведения (или частного) может исчезнуть переполнение или возникнуть антипереполнение. Так как порядки чисел с плавающей запятой представлены в форме характеристик и являются числами без знака, то наличие переполнения или исчезновения порядка определяется после сложения (или вычитания) характеристик путем анализа признака переноса (или заема) С и старшего разряда суммы (или разности) характеристик.

Чтобы упростить процедуру формирования суммы или разности мантисс и исключить сложную процедуру анализа знаков, мантиссы чисел перед выполнением операции сложения или вычитания переводятся в дополнительный код, а после операции делается обратное преобразование мантиссы результата в прямой код.

Аналогично, при выполнении умножения или деления, характеристики заменяются порядками со знаком в дополнительном коде. При этом возникновение переполнения или исчезновения порядков (антипереполнения) при сложении (вычитании) порядков может быть определено по признаку переполнения V.

По результатам выполнения операций над числами с плавающей запятой должны быть сформированы признаки PSW. Признак N определяется знаком результата операции. Признак V устанавливается в 1, если в результате выполнения операции возникло переполнение; при этом полученные значения мантиссы и характеристики записываются как результат операции. Признак Z устанавливается в 1, если возникло антипереполнение или мантисса резульмата равна 0; при этом знак результата операции сохраняется, а мантисса и характеристика сбрасываются в 0. Признак С после выполнения операции над числами с плавающей запятой должен быть сброшен в 0.

Практическая часть

Практическая часть работы включает выполнение следующих действий:

а) в соответствии с индивидуальным заданием разработка алгоритмов выполнения двух арифметических операций над числами с плавающей запятой: одна операция - сложение или вычитание, вторая операция - умножение или деление;

б) в соответствии с разработанными алгоритмами составление программ выполнения арифметических операций над числами с плавающей запятой;

в) формирование и занесение в память и РОНы исходных значений и необходимых вспомогательных данных;

г) запись в память программ выполнения арифметических операций над числами с плавающей запятой;

д) выполнение программ;

е) контроль результатов работы программ.

Правильность разработки и выполнения программ выполнения арифметических операций над числами с плавающей запятой контролируется путем ручной трассировки контрольных примеров с последующим сравнением результатов работы программы с результатами ручной трассировки.

Варианты заданий.

Варианты индивидуальных заданий приведены в табл. 2.14 и табл. 2.15.

Порядок выполнения работы.

А. В процессе самостоятельной работы

1. Выбрать исходные данные в соответствии с номером варианта.
2. Составить алгоритмы программ для решения поставленной задачи.
3. Составить программы вычислений в мнемонических и машинных кодах.
4. Составить карту распределения памяти под команды и данные.
5. Произвести ручную трассировку программ с использованием самостоятельно выбранных исходных данных, при этом в таблице трассировки должны быть отражены значения информмации в ячейках памяти и используемых регистрах, а также значения в регистре слова состояния PSW.
6. Оформить отчет по лабораторной работе.

Б. В учебной лаборатории

1. Записать в регистры и в память исходные данные, программы и дополнительную информацию, необходимую для реализации заданных алгоритмов.
2. Выполнить программы, занесенные в память.
3. Проверить результаты выполнения программ, сравнивая их с результатами ручной трассировки алгоритмов.

Содержание отчета

1. Титульный лист.
2. Текст задания.
3. Перевод исходных данных в шестнадцатиричную систему счисления.
4. Схемы алгоритмов программ.
5. Тексты программ в мнемонических и машинных кодах.
6. Карта распределения памяти под команды и данные.
7. Таблицы трассировки программ.

Программируемые операции Таблица 2.14

Вариант	Формат данных	1 операция	Действия над мантиссами	2 операция	Действия над порядками
1	F	Сложение	В прямых кодах	Деление	В дополнительных кодах
2	D	Вычитание	В прямых кодах	Деление	В дополнительных кодах
3	G	Сложение	В дополнительных кодах	Деление	В формате характеристик
4	H	Вычитание	В дополнительных кодах	Деление	В формате характеристик
5	F	Сложение	В прямых кодах	Умножение	В формате характеристик
6	D	Вычитание	В прямых кодах	Умножение	В дополнительных кодах
7	G	Сложение	В дополнительных кодах	Умножение	В дополнительных кодах
8	H	Вычитание	В дополнительных кодах	Умножение	В формате характеристик
9	F	Сложение	В дополнительных кодах	Умножение	В формате характеристик
10	D	Вычитание	В дополнительных кодах	Умножение	В дополнительных кодах
11	G	Сложение	В прямых кодах	Умножение	В формате характеристик
12	H	Вычитание	В прямых кодах	Умножение	В дополнительных кодах
13	F	Сложение	В дополнительных кодах	Умножение	В дополнительных кодах
14	D	Вычитание	В прямых кодах	Умножение	В дополнительных кодах
15	G	Сложение	В дополнительных кодах	Умножение	В формате характеристик
16	H	Вычитание	В прямых кодах	Умножение	В дополнительных кодах
17	D	Сложение	В прямых кодах	Деление	В дополнительных кодах
18	H	Вычитание	В прямых кодах	Деление	В дополнительных кодах
19	G	Сложение	В дополнительных кодах	Деление	В формате характеристик
20	F	Вычитание	В дополнительных кодах	Деление	В формате характеристик

Алгоритмы выполнения умножения или деления Таблица 2.15

Вариант	Базовый алгоритм
1	Деление с восстановлением остатка с неподвижным делимым и сдвигом делителя
2	Деление без восстановления остатка с неподвижным делимым и сдвигом делителя
3	Деление с восстановлением остатка с неподвижным делителем и сдвигом делимого
4	Деление без восстановления остатка с неподвижным делителем и сдвигом делимого

5	Умножение с неподвижной суммой частичных произведений, сдвигом множимого вправо и анализом множителя, начиная со старших разрядов
6	Умножение с неподвижной суммой частичных произведений, сдвигом множимого влево и анализом множителя, начиная с младших разрядов
7	Умножение со сдвигом суммы частичных произведений влево, неподвижным множимым и анализом множителя, начиная со старших разрядов

Алгоритмы выполнения умножения или деления Таблица 2.15

Вариант	Базовый алгоритм
1	Деление с восстановлением остатка с неподвижным делимым и сдвигом делителя
2	Деление без восстановления остатка с неподвижным делимым и сдвигом делителя
3	Деление с восстановлением остатка с неподвижным делителем и сдвигом делимого
4	Деление без восстановления остатка с неподвижным делителем и сдвигом делимого
5	Умножение с неподвижной суммой частичных произведений, сдвигом множимого вправо и анализом множителя, начиная со старших разрядов
6	Умножение с неподвижной суммой частичных произведений, сдвигом множимого влево и анализом множителя, начиная с младших разрядов
7	Умножение со сдвигом суммы частичных произведений влево, неподвижным множимым и анализом множителя, начиная со старших разрядов
8	Умножение со сдвигом суммы частичных произведений вправо, неподвижным множимым и анализом множителя, начиная с младших разрядов
9	Умножение с неподвижной суммой частичных произведений, сдвигом множимого влево и анализом множителя, начиная с младших разрядов
10	Умножение со сдвигом суммы частичных произведений влево, неподвижным множимым и анализом множителя, начиная со старших разрядов
11	Умножение со сдвигом суммы частичных произведений вправо, неподвижным множимым и анализом множителя, начиная с младших разрядов
12	Умножение с неподвижной суммой частичных произведений, сдвигом множимого вправо и анализом множителя, начиная со старших разрядов
13	Умножение со сдвигом суммы частичных произведений влево, неподвижным множимым и анализом множителя, начиная со старших разрядов
14	Умножение со сдвигом суммы частичных произведений вправо, неподвижным множимым и анализом множителя, начиная с младших разрядов
15	Умножение с неподвижной суммой частичных произведений, сдвигом множимого вправо и анализом множителя, начиная со старших разрядов
16	Умножение с неподвижной суммой частичных произведений, сдвигом множимого влево и анализом множителя, начиная с младших разрядов
17	Деление без восстановления остатка с неподвижным делителем и сдвигом делимого
18	Деление с восстановлением остатка с неподвижным делителем и сдвигом делимого
19	Деление без восстановления остатка с неподвижным делимым и сдвигом делителя
20	Деление с восстановлением остатка с неподвижным делимым и сдвигом делителя

Литература

1. Остапенко Г.П., Толмачева Н.А., Горский В.Е. Макроассемблер для СМ-1700. - М.: Финансы и статистика, 1990.

2. Ч.Кэпс, Р.Стаффорд VAX-11: Программирование на языке ассемблера и архитектура: Пер. с англ. М.: Радио и связь,1991

3. Злобин В.К., Григорьев В.Л. Программирование арифметических операций в микропроцессорах: Учебное пособие для технических ВУЗов, - М.: Высшая школа, 1991.

Система адресации данных в ЭВМ семейства VAX-11

Способы адресации в ЭВМ семейства VAX-11 Таблица П.1

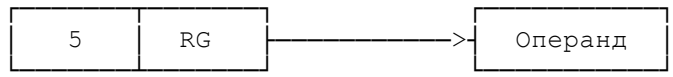
Способ адресации	Спецификация операнда	Код режима	Мнемокод
Литеральная адресация	00111111	0,1,2,3	# Литерал
Адресация с индексацией	Спец.базы 4R	4	Спец.базы [R]
Регистровая адресация	5R	5	R
Косвенно-регистровая адресация	6R	6	(R)
Адресация с автоувеличением	8R	8	(R)+
Косвенная адресация с автоувеличением	9R	9	@ (R)+
Адресация с автоуменьшением	7R	7	- (R)
Адресация по смещению			
(смещение - байт)	xx AR	A	Смещение (R)
(смещение - слово)	xxxx CR	C	Смещение (R)
(смещение - длинное слово)	xxxxxxxx ER	E	Смещение (R)
Косвенная адресация по смещению			
(смещение - байт)	xx BR	B	@ Смещение (R)
(смещение - слово)	xxxx DR	D	@ Смещение (R)
(смещение - длинное слово)	xxxxxxxx FR	F	@ Смещение (R)
Непосредственная адресация	x...x 8F	8	# Const
Абсолютная адресация	xxxxxxxx 9F	9	@# Адрес
Относительная адресация			
(смещение - байт)	xx AF	A	Адрес
(смещение - слово)	xxxx CF	C	Адрес
(смещение - длинное слово)	xxxxxxxx EF	E	Адрес
Косвенная относительная адресация			
(смещение - байт)	xx BF	B	@ Адрес
(смещение - слово)	xxxx DF	D	@ Адрес
(смещение - длинное слово)	xxxxxxxx FF	F	@ Адрес

Способы адресации с индексацией Таблица П.2

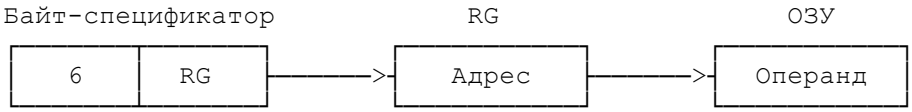
Способ адресации	Мнемокод
Косвенно-регистровая адресация с индексацией	(Rn) [Rx]
Индексная адресация с автоувеличением	(Rn)+ [Rx]
Индексная адресация с автоуменьшением	- (Rn) [Rx]
Косвенная адресация с автоувеличением и индексацией	@ (Rn)+ [Rx]
Адресация по смещению с индексацией	смещение (Rn) [Rx]
Косвенная адресация по смещению с индексацией	@ смещение (Rn) [Rx]
Абсолютная адресация с индексацией	@ # адрес [Rx]
Относительная адресация с индексацией	адрес [Rx]
Косвенная относительная адресация с индексацией	@ адрес [Rx]

Схемы выборки операндов при различных способах адресации

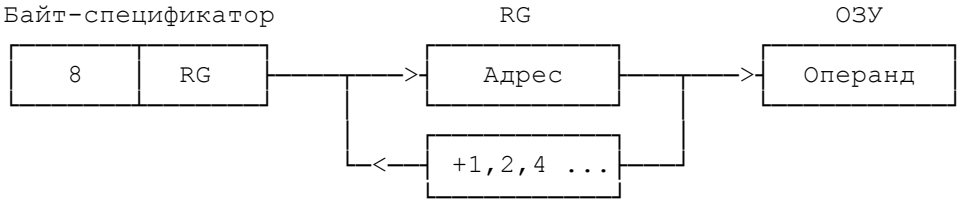
Прямая регистровая адресация
Код режима адресации = 5



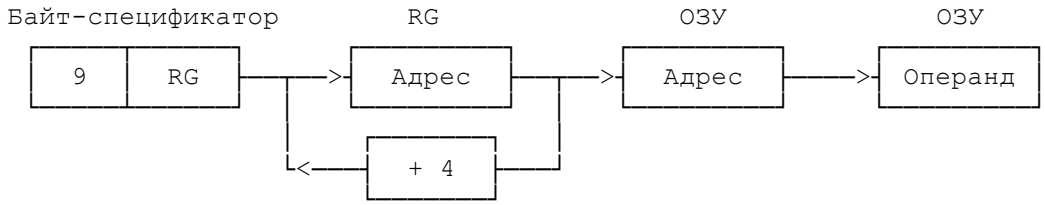
Косвенно-регистровая (простая косвенная) адресация
Код режима адресации = 6



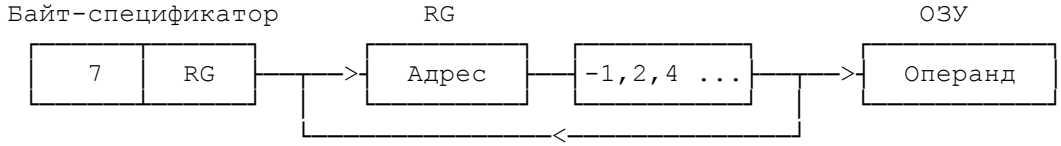
Адресация с автоувеличением (простая косвенная с автоувеличением)
Код режима адресации = 8



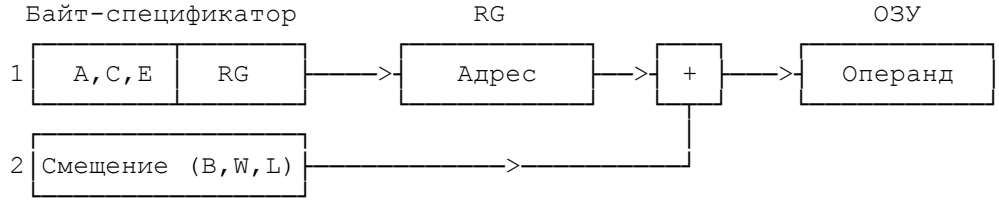
Косвенная адресация с автоувеличением (двойная косвенная с автоувеличением)
Код режима адресации = 9



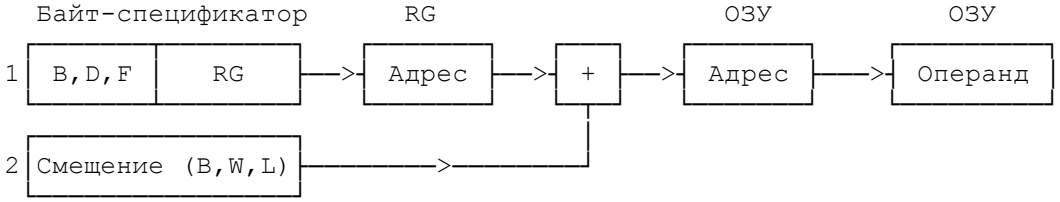
Адресация с автоуменьшением (простая косвенная с автоуменьшением)
Код режима адресации = 7



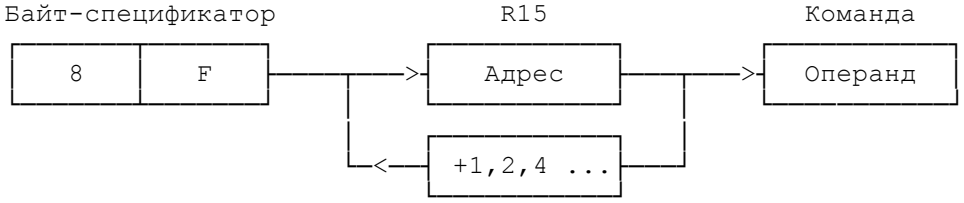
Адресация по смещению
Коды режимов адресации = A,C,E



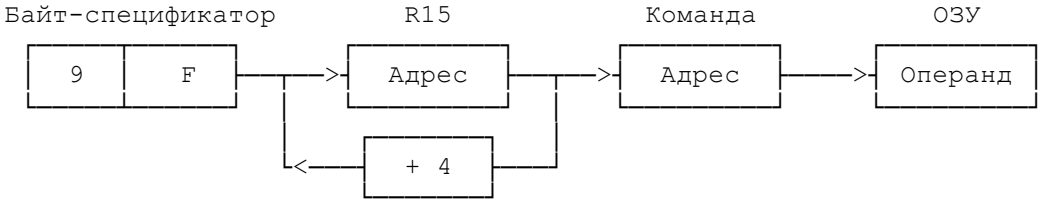
Косвенная адресация по смещению
Коды режимов адресации = B,D,F



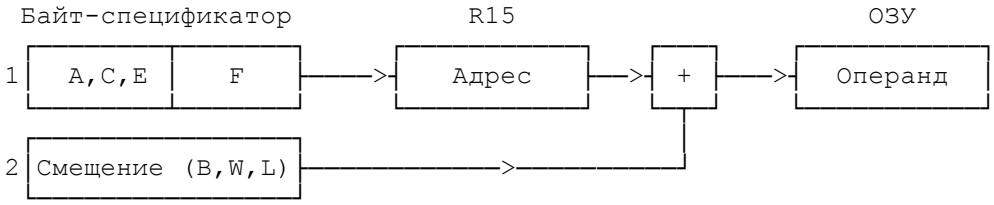
Непосредственная адресация
Код режима адресации = 8



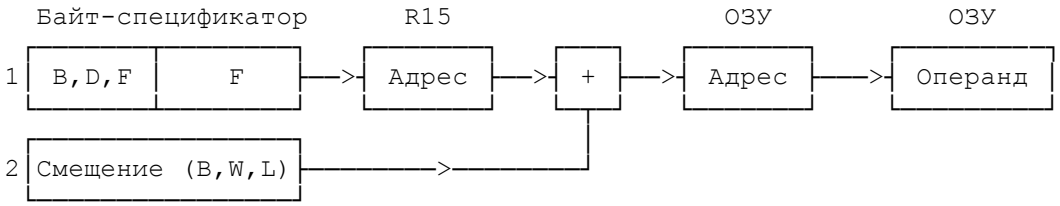
Абсолютная адресация
Код режима адресации = 9



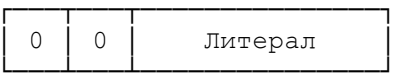
Относительная адресация
Коды режимов адресации = A, C, E



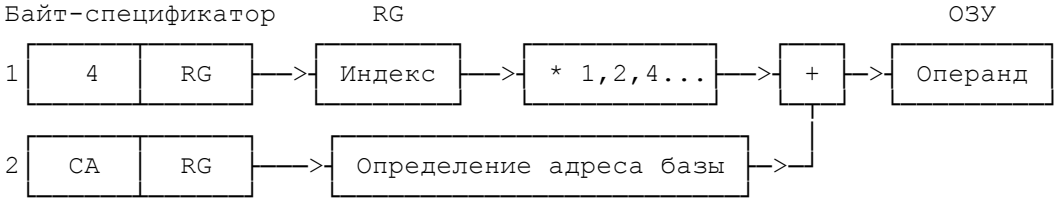
Косвенная относительная адресация
Коды режимов адресации = B, D, F



Литеральная адресация
Коды режимов адресации = 0,1,2,3



Адресация с индексацией
Код режима адресации = 4



Набор команд системы VAX-11, реализуемых в эмулирующей программе

Таблица П.3

Команда	Код	Выполняемая функция	Флаги N Z V C
ADDB2 ADDW2 ADDL2	80 A0 C0	Сложение двухоперандное форматов B,W,L (оп2) = (оп2) + (оп1)	* * * *
ADDB3 ADDW3 ADDL3	81 A1 C1	Сложение трехоперандное форматов B,W,L (оп3) = (оп2) + (оп1)	* * * *
ADWC	D8	Сложение двухоперандное с переносом формата L: (оп2) = (оп2) + (оп1) + C	* * * *
SUBB2 SUBW2 SUBL2	82 A2 C2	Вычитание двухоперандное форматов B,W,L (оп2) = (оп2) - (оп1)	* * * *
SUBB3 SUBW3 SUBL3	83 A3 C3	Вычитание трехоперандное форматов B,W,L (оп3) = (оп2) - (оп1)	* * * 0
SBWC	D9	Вычитание двухоперандное с заемом формата L: (оп2) = (оп2) - (оп1) - C	* * * *
INCB INCW INCL	96 B6 D6	Инкремент операндов форматов B,W,L (оп) = (оп) + 1	* * * *
DECB DECW DECL	97 B7 D7	Декремент операндов форматов B,W,L (оп) = (оп) - 1	* * * *
ASHL ASHQ	78 79	Арифметический сдвиг операндов форматов L,Q (оп3) = (оп2) * [2 ** (оп1)]	* * * 0
MOVB MOVW MOVL MOVQ MOVO	90 B0 D0 7D 7D FD	Пересылка операндов разного формата (оп2) = (оп1)	* * 0 -
MCOMB MCOMW MCOML	92 B2 D2	Пересылка операнда в обратном коде (инверсия), форматы операндов B,W,L (оп2) = not (оп1)	* * 0 -

Продолжение табл.П.3

Команда	Код	Выполняемая функция	Флаги N Z V C
MNEGB MNEGW MNEGL	8E AE CE	Пересылка операнда в дополнительном коде (смена знака), форматы операндов B,W,L (оп2) = 0 - (оп1)	* * * *
CLRB CLRW CLRL CLRQ CLRO	94 84 D4 7C 7C FD	Очистка операндов форматов B,W,L,Q,O (оп) = 0	0 1 0 -
BICB2 BICW2 BICL2	8A AA CA	Очистка разрядов операндов форматов B,W,L (оп2) = (оп2) and [not (оп1)]	* * 0 -
BISB2 BISW2 BISL2	88 A8 C8	Установка разрядов операндов форматов B,W,L (оп2) = (оп2) or (оп1)	* * 0 -
XORB2 XORW2 XORL3	8C AC CC	Двухоперандное "исключающиее ИЛИ" операндов форматов B,W,L (оп2) = (оп2) xor (оп1)	* * 0 -
ACBB ACBW ACBL	9D 3D F1	Сложение, сравнение операндов B,W,L и переход (оп3) = (оп3) + (оп2) ; [(оп3) <=> (оп1)] -> BRW оп4	* * * -
BICPSW	B9	Сброс флагов PSW: PSW = PSW and [not (оп1)], оп = W	* * * *
BISPSW	B8	Установка флагов PSW: PSW = PSW or (оп1), оп = W	* * * *
BITB BITW BITL	93 B3 D3	Проверка разрядов слова форматов B,W,L PSW <- [(оп2) and (оп1)]	* * 0 -
CMPB CMPW CMPL	91 B1 D1	Сравнение двух операндов форматов B,W,L PSW <- [(оп1) - (оп2)] (! Вычитание из первого ОП !)	* * 0 *
TSTB TSTW TSTL	95 B5 D5	Установка флагов в соответствии со значением ОП, форматы операндов: B,W,L PSW <- (оп1)	* * 0 0
BEQL	13	Переход по "равно нулю" [Z = 1]	- - - -
BNEQ	12	Переход по "не равно нулю" [Z = 0]	- - - -

Окончание табл.П.3

Команда	Код	Выполняемая функция	Флаги N Z V C
BGEQ	18	Переход по "больше или равно (со знаком)" [N (+) V = 0]	- - - -
BGEQU (BCC)	1E	Переход по "больше или равно (без знака)" по "нет переноса" [C = 0]	- - - -
BGTR	14	Переход по "больше (со знаком)" [Z v [N(+)V] = 0]	- - - -
BGTRU	1A	Переход по "больше (без знака)" [C v Z = 0]	- - - -
BLEQ	15	Переход по "меньше или равно (со знаком)" [Z v [N(+)V] = 1]	- - - -
BLEQU	1B	Переход по "меньше или равно (без знака)" [C v Z = 1]	- - - -
BLSS	19	Переход по "меньше (со знаком)" [N (+) V = 1]	- - - -
BLSSU (BCS)	1F	Переход по "меньше (без знака)" по "есть перенос" [C =1]	- - - -
BVC	1C	Переход по "нет переполнения" (со знаком) [V = 0]	- - - -
BVS	1D	Переход по "есть переполнение" (со знаком) [V = 1]	- - - -
BRB	11	Безусловный переход по смещению Byte	- - - -
BRW	31	Безусловный переход по смещению Word	- - - -
JMP	17	Безусловный переход по адресу оп (оп - байт)	- - - -
JSB	16	Переход на П/П по адресу оп (оп - байт)	- - - -
RSB	05	Возврат из П/П	- - - -
HALT	00	Останов	- - - -
NOP	01	Пустая операция	- - - -

ОПИСАНИЕ МОДЕЛИРУЮЩЕЙ ПРОГРАММЫ VAX_OLD

ЗАПУСК ПРОГРАММЫ

Для запуска моделирующей программы требуется указать в командной строке DOS имя основного файла эмулятора vax.exe, и в случае необходимости, полный путь к этому файлу. В командной строке после имени эмулятора можно указывать имена файлов данных, которые будут автоматически загружены и помещены в соответствующие окна моделирования. Файлы, указываемые в командной строке, должны реально существовать. Не стоит указывать более двух файлов для загрузки, так как система в состоянии одновременно обрабатывать не более двух окон моделирования.

Например, после указания в командной строке vax.exe sub.vax, где sub.vax файл данных, запускается эмулирующая программа и появляется следующее окно:

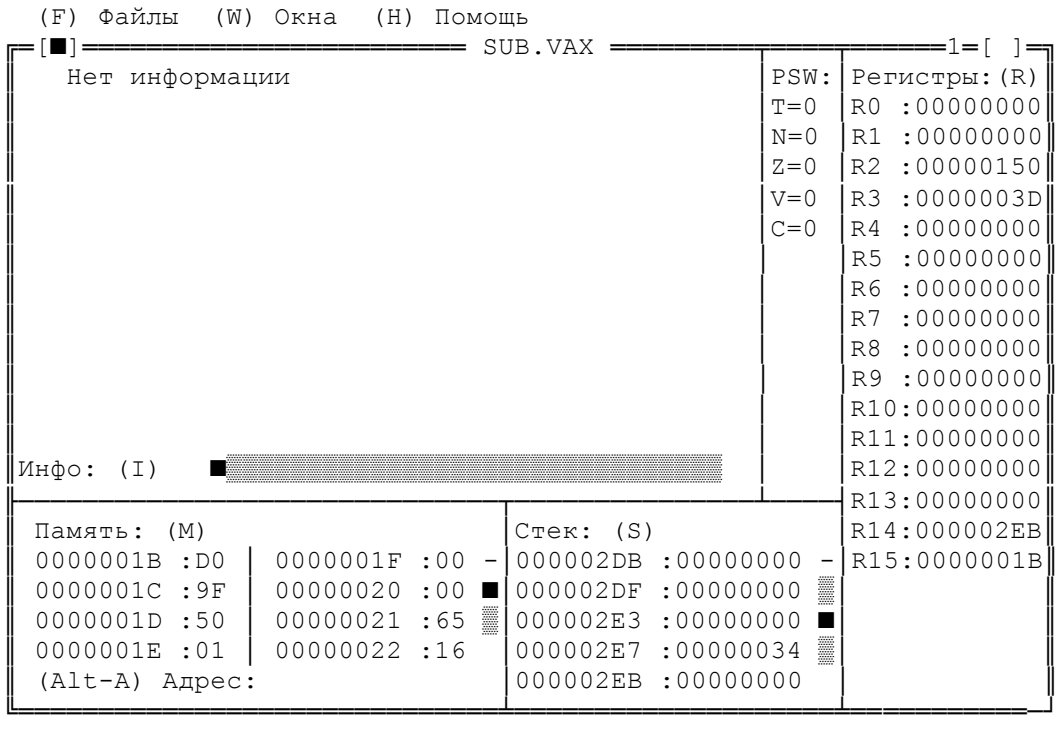


Рис. П.1 Окно моделирующей программы

Окно на рис. П.1 содержит: в верхней части - строку главного меню (для входа в которое необходимо нажать функциональную клавишу [F10]), в центральной части - окно моделирования, которое отражает полную информацию моделирующей ЭВМ, в нижней части – строку подсказки, в которой указывается назначение "горячих" клавиш, дублирующих команды основного меню.

РАБОТА С ФАЙЛАМИ

Для работы с файлами служат команды меню [(F) Файлы]. Для входа в меню можно использовать комбинацию клавиш [Alt - F].

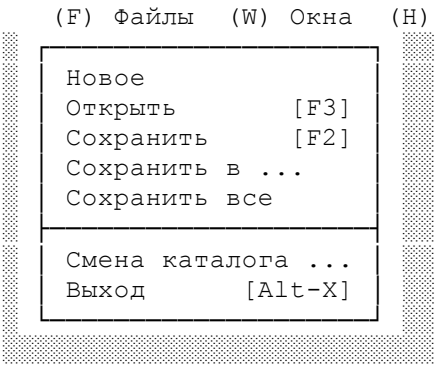


Рис. П.2 Команды меню [(F) Файлы]

Система эмуляции позволяет:
- создавать новые файлы, содержащие информацию о состоянии моделируемой ЭВМ в соответствующем окне моделирования;
- сохранять информацию из любого окна моделирования в файле с произвольным именем;
- открывать существующие файлы данных, загружая информацию в новое или текущее окно моделирования.

Рассмотрим перечисленные возможности системы эмуляции.

Создание нового окна моделирования.

Для создания нового окна моделирования используется пункт меню [F(Файлы)] "Новое". При отсутствии открытых окон моделирования, эта же команда может быть выполнена при нажатии клавиши [F4] . При создании нового окна оно получает имя NONAME. При завершении работы с подобным окном или в случае выбора пункта меню [F(Файлы)] "Сохранить" пользователю будет предложено сохранить содержимое окна в файле данных с новым именем. После записи содержимого окна в файл оно будет переименовано в соответствии с именем файла данных, связанного с окном.

В данной системе для сохранения состояния моделируемой ЭВМ используются файлы специального формата : *.vax со стандартным расширением .vax . В файле содержится информация о содержимом ОЗУ, регистров общего назначения и слова состояния процессора (PSW) моделируемой ЭВМ.

Сохранение окна моделирования в файле

Для сохранения информации из текущего активного окна моделирования используется пункт меню [F(Файлы)] "Сохранить". При отсутствии открытых окон моделирования, эта команда запрещена. Данная команда может быть выполнена при нажатии клавиши [F2] , без входа в меню. Если текущее окно связано с файлом данных, то запись будет произведена в данный файл. Если окно имеет заголовок NONAME, то для записи будет вызвано служебное окно диалога и будет предложено указать имя файла для записи содержимого окна.

Сохранение содержимого окна моделирования в файле под новым именем

Для сохранения информации из текущего окна моделирования в файле данных с новым именем используется пункт меню [F(Файлы)] "Сохранить в ...". При выполнении данной команды будет открыто окно диалога и предложено указать новое имя файла. После успешного завершения операции в заголовок окна моделирования будет выведено новое имя.

Открытие существующего файла

Для загрузки информации из файла данных на диске в окно моделирования используется пункт меню [F(Файлы)] "Открыть". Эту же команду можно выполнить без входа в меню, используя "горячую" клавишу [F3]. После выполнения данной команды разворачивается окно диалога и предлагается указать имя файла, который будет связан с окном. После выбора имени файла, находясь в дилоговом окне следует, используя клавишу [Tab], комбинацию клавиш [Shift-Tab] или левую кнопку манипулятора типа "мышь", выбрать одну из кнопок, расположенных в диалоговом окне. При выборе кнопки "Открыть" и нажатии клавиши [Enter] на клавиатуре, будет окрыто новое окно моделирования, с которым будет связан открываемый файл. (Эта функция выполняется также по умолчанию при нажатии клавиши [Enter] сразу после выбора имени файла для загрузки. При выборе кнопки "Заместить" , файл будет загружен в текущее открытое окно моделирования. Если в нем есть некоторая информация, то будет предложено ее сохранить.

Сохранить все измененные окна

Для сохранения информации из всех открытых окон моделирования используется пункт меню [F(Файлы)] "Сохранить все". При выполнении данной команды будет произведена запись данных из всех измененных окон моделирования, а не только из текущего. В случае, если какое-либо из открытых окон имеет заголовок NONAME, для него будет развернуто окно диалога и будет предложено указать новое имя файла.

Изменение текущего каталога

Текущий каталог - это каталог, содержимое которого по умолчанию выводится в диалоговых окнах открытия и записи файлов. Для изменения текущего каталога следует воспользоваться пунктом меню [F(Файлы)] "Сменить каталог", после выполнения которого будет развернуто окно диалога, в котором следует указать имя нового каталога или диска.

РАБОТА С ОКНАМИ

Для работы с окнами в системе используются команды меню [W[Окна]) (рис.П.3). Для входа в меню можно использовать комбинацию клавиш [Alt - W].

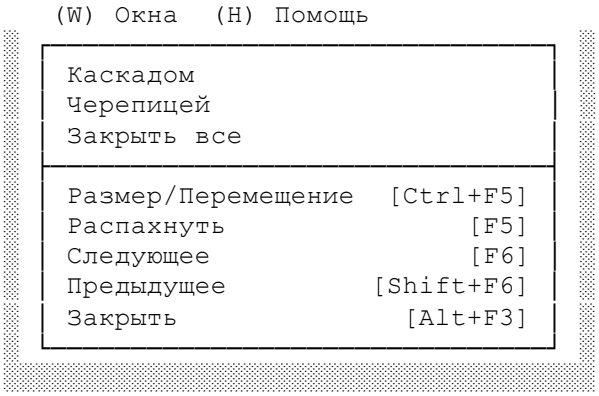


Рис. П.3 Команды меню [W (Окна)]

Система эмуляции позволяет:
- располагать окна одно за другим (каскадом) и в стык без наложения (черепицей) ;
- изменять величину и положение окон ;
- убирать с экрана активное окно или все окна ;
- увеличивать размеры активного окна для использования полного экрана ;
- переключаться между открытыми окнами.

Рассмотрим перечисленные возможности системы эмуляции.

Расположение окон "каскадом"

Выполнение данной команды расположит открытые окна одно за другим. Активное окно будет расположено первым и полностью доступно для просмотра. Второе окно будет располагаться за активным таким образом, что будет виден его заголовок.

Расположение окон "черепицей"

Выполнение данной команды приведет к расположению окон на экране без перекрытия. При этом уменьшится размер каждого окна, но можно будет одновременно наблюдать состояние двух окон. Активное окно выделяется двойной рамкой.

Изменение положения и размеров окон

Используя команду меню [W(Окна)] "Размер/Перемещение" можно изменить положение и размеры текущего окна. Данная команда может быть выполнена без входа в меню при нажатии комбинации клавиш [Ctrl-F5]. Для перемещения окна используются клавиши - "стрелки". Для изменения размеров окна, удерживая нажатой клавишу [Shift], необходимо использовать стрелочные клавиши. Для установки заданных размеров окна необходимо нажать клавишу [Enter]. Для отказа от операции и восстановления предыдущих размеров и положения окна служит клавиша [Esc].

Закрытие окон

Используя команды меню [W(Окна)] "Закрыть", "Закрыть все" можно убрать с экрана одно (текущее, активное) или все открытые окна. При уничтожении активного окна происходит переключение на следующее за ним окно (если такое существует). Если в уничтожаемом окне находится несохраненная информация, производится запрос у пользователя о необходимости ее сохранения. Команда "Закрыть" может быть выполнена без входа в меню при нажатии комбинации клавиш [Alt-F3].

Расширение окна на весь экран

Используя команду меню [W(Окна)] "Распахнуть" можно максимально увеличить текущее активное окно, которое займет весь доступный экран. Повторное выполнение команды приведет к восстановлению исходных размеров окна. Данную команду можно выполнить без входа в меню при помощи "горячей" клавиши [F5]. Эта команда может быть полезна при работе с окнами, расположенными черепицей.

Переключение между открытыми окнами

Для переключения между открытыми окнами можно использовать две команды меню [W(Окна)]:

"Следующее" - данная команда делает активным следующее по порядку активное окно ("горячая" клавиша [F6]).

"Предыдущее" - выполняет переключение в предыдущее активное окно (комбинация клавиш [Shift-F6]).

Все открытые окна образуют кольцевой список. Поэтому, используя любую из вышеуказанных команд, можно перебрать все активные окна. Для быстрого переключения в конкретное окно можно использовать комбинацию клавиш [Alt-Номер окна]. Номер окна - число (1 или 2), выводимое в рамке окна, рядом с заголовком. Такое переключение возможно только для окон, имеющих номер.

РАБОТА С ОКНОМ МОДЕЛИРОВАНИЯ

Любое окно моделирования, открытое в системе, автоматически связывается с файлом данных, имя которого отображается в заголовке окна. Если было создано новое окно, оно получает имя NONAME, а при сохранении содержимого такого окна будет предложено задать имя файлу данных (сохранение нового окна).

В окне моделирования (рис.П.4) расположены: редактор ОЗУ, редактор регистров, редактор флагов (PSW), монитор стека и монитор отладочной информации. Для переключения между элементами окна служат клавиша [Tab] или комбинация клавиш [Shift-Tab] (обратное направление переключения), а также специальные "быстрые" клавиши для активизации каждого из элементов : [Alt-M] (Memory) - для редактора ОЗУ, [Alt-R] (Register) - для редактора регистров, [Alt-I] (Information) - для монитора отладочной информации, [Alt-P] (PSW) - для редактора PSW, [Alt-S] (Stack) - для монитора стека. Аналогичные действия по переключению можно выполнить используя левую кнопку манипулятора типа "мышь".

В каждый момент времени в системе активным является только одно окно. Ввод данных и моделирование выполнения команд ЭВМ осуществляется только для активного окна.

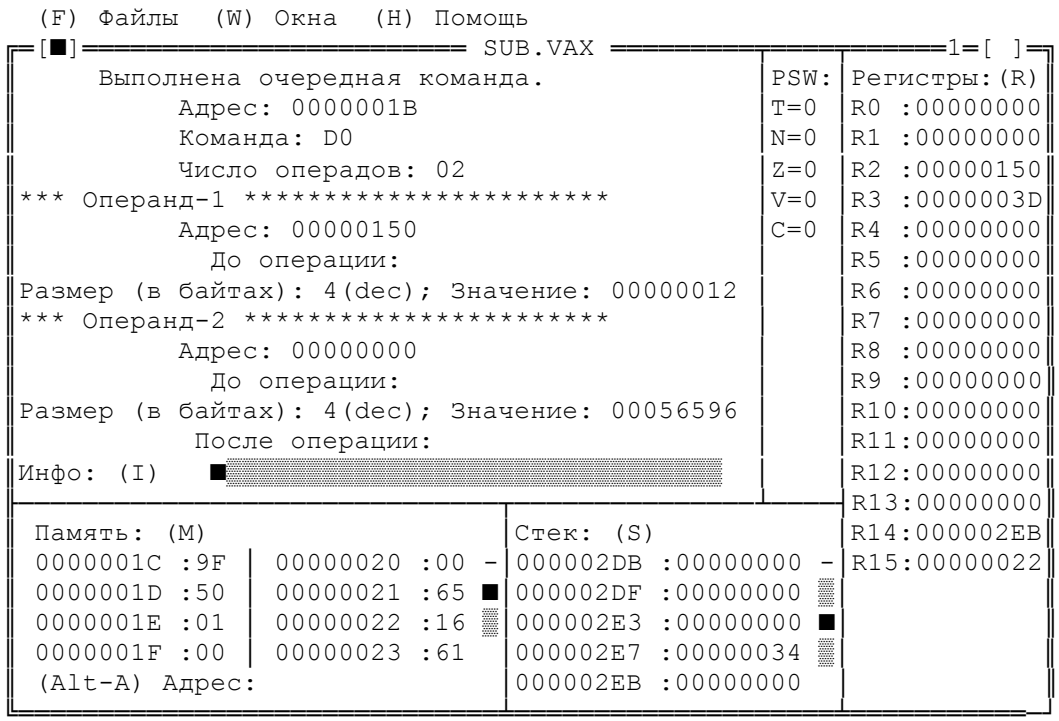


Рис. П.4 Окно моделирования

Редактирование ОЗУ

Для ввода и отображения информации в ОЗУ моделируемой ЭВМ служит информационная панель "Память" в окне моделирования. Для активизации редактора ОЗУ необходимо нажать [Alt-M].

Память адресуется побайтно. В данной моделирующей системе доступен 1 КБ оперативной памяти ЭВМ семейства VAX. Максимальный адрес - ^X 3FF, и вся информация представляется в шестнадцатеричном формате.

Для перемещения по содержимому памяти используются клавиши - "стрелки", [Home], [End] и [PgUp], [PgDn]. Использование этих клавиш позволяет перемещать указатель на нужный байт. Для быстрого перехода на определенный адрес необходимо воспользоваться комбинацией [Alt-A] и ввести необходимый адрес перехода.

Для редактирования значения байта нужно его выделить (поместить на него указатель) и набрать на клавиатуре новое значение в шестнадцатеричном формате. Ввод завершается нажатием клавиши [Enter]. При успешном вводе осуществляется автоматический переход указателя активного байта на следующий адрес. Например, на рис.П.4 указатель активного байта показывает на адрес ^X 00000022 ОЗУ. Текущее содержимое этого байта - ^X 16. Данный байт доступен для редактирования.

При моделировании выполнения команд из ОЗУ происходит перемещение указателя активного байта памяти в соответствии со значением счетчика команд - R15.

Редактирование регистров

Для ввода и отображения информации в регистрах центрального процессора моделируемой ЭВМ служит панель "Регистры" в окне моделирования. Для инициализации редактора регистров необходимо набрать на клавиатуре [Alt-R].

В системе моделируется 16 32-разрядных РОН центрального процессора. Вся информация представляется в шестнадцатеричном формате. Для перемещения по набору регистров используются клавиши - "стрелки", [Home], [End] и [PgUp], [PgDn]. Использование этих клавиш позволяет перемещать указатель на нужный регистр.

Для редактирования значения регистра надо его выделить (поместить на него указатель) и набрать на клавиатуре новое 32-битное значение в шестнадцатеричном формате. Ввод завершается нажатием клавиши [Enter].

Просмотр отладочной информации

Для отображения информации о результатах моделирования служит информационная панель "Инфо" в окне моделирования. Для активизации данной панели надо набрать на клавиатуре [Alt-I].

В мониторе отладочной информации в текстовом виде отображается информация о последней выполненной команде и ее операндах, либо о причинах сбоя в случае ошибки моделирования. Для просмотра всей информации, в случае, если она не помещается полностью в отведенное поле, используется скроллинг по двум направлениям с помощью "стрелок". Например, на рис.2.4 в мониторе отладочной информации можно видеть результат выполнения команды MOVL (код D0) , расположенной в ОЗУ по адресу ^X 0000001B. Команда - двухоперандная. Показаны значения операндов 1 и 2 до выполнения операции. Чтобы увидеть результат операции (значение операнда 2 после выполнения команды) необходимо воспользоваться скроллингом в вертикальном направлении.

Редактирование регистра флагов ЦП

Для отображения информации слова состояния ЦП моделируемой ЭВМ служит информационная панель "PSW" в окне моделирования. Активизация редактора регистра флагов осуществляется по [Alt-P]. Редактор предоставляет доступ к 4-м основным флагам ЦП (Переполнение - V, перенос - C, флаг нуля - Z, флаг знака - N). Флаг T (трассировка) смысловой нагрузки в данной модели не несет.

Для изменения значения любого флага необходимо, пользуясь клавишами "стрелки", подвести указатель к соответствующему флагу и нажать клавишу [Enter]. При этом значение флага изменится на противоположное.

Просмотр стека

Для отображения информации о состоянии стека ЦП моделируемой ЭВМ служит информационная панель "Стек" в окне моделирования. Активизация монитора стека осуществляется по [Alt-S].

Стек моделируемой ЭВМ представляет собой область ОЗУ, на которую установлен указатель стека (регистр R14). Информация из стека представляется на экране в виде 32-разрядных слов в шестнадцатеричном формате. Стек можно посмотреть побайтно, если воспользоваться редактором ОЗУ, установив его на соответствующий адрес. С помощью клавиш "стрелки" возможно перемещение указателя текущего элемента стека, чем обеспечивается просмотр его содержимого. Перемещение указателя активного элемента в окне стека не влияет на состояние R14 , и следовательно, на результаты моделирования.

Если в ходе изменения указателя стека он будет установлен на недействительный адрес памяти, информация в окне стека исчезнет, а при дальнейшей работе с монитором стека значение указателя стека будет игнорироваться до тех пор, пока не будет установлено верное значение указателя.

При установке указателя стека следует по возможности выравнивать стек на границу длинного слова (32 разряда, 4 байта).

МОДЕЛИРОВАНИЕ ВЫПОЛНЕНИЯ КОМАНД VAX

Данная система позволяет эмулировать выполнение команд центрального процессора ЭВМ семейства VAX, рассмотренных в разделе 1.5.

Моделирование производится для текущего активного окна моделирования (такое окно выделено двойной рамкой). Текущая выполняемая команда выбирается из памяти в соответствии со значением программного счетчика R15 и выполняется. Результаты выполнения команды влияют на содержимое ОЗУ, регистров ЦП и PSW и выводятся в панель отладочной информации.

Существует 3 режима выполнения команд:

- "Трассировка" - по нажатию клавиши [F7]. В этом режиме производится выполнение одной команды, адресуемой регистром R15.
- "Шаг" - по нажатию клавиши [F8]. Производится выполнение команды, адресуемой R15. Если данная команда является командой перехода на подпрограмму, то производится автоматический прогон всей подпрограммы, вплоть до выполнения команды возврата. При "зависании" моделируемой программы - следует нажать клавишу [Esc].
- "Прогон" - по нажатию клавиши [F9]. Этот режим предназначен для автоматического выполнения всей программы находящейся в ОЗУ моделируемой ЭВМ, до выполнения команды HALT ЦП VAX, либо до прерывания от пользователя (клавиша [Esc]).

ЗАВЕРШЕНИЕ РАБОТЫ С ПРОГРАММОЙ

При выборе пункта меню [F(Файлы)] "Выход" будут закрыты все открытые окна и завершена работа с моделирующей системой. Если в системе есть окна с несохраненной информацией, то будет предложено ее сохранить. Если операция сохранения будет отменена пользователем, то окна не будут закрыты и выхода не последует. Для выхода из программы без входа в меню, служит комбинация клавиш [Alt-X].