Государственный комитет по высшему образованию
Российской федерации

Ульяновский государственный технический университет

ФУНКЦИОНАЛЬНАЯ ОРГАНИЗАЦИЯ МИКРО-ЭВМ И МИКРОКОНТРОЛЛЕРОВ

Часть 1: PDP-11

Методические указания для студентов направления
"Информатика и вычислительная техника"

Составители: В.Н.Негода
И.А.Никищенков

Ульяновск 1996

УДК 681.3(076)

Функциональная организация микро-ЭВМ и микроконтроллеров. Часть 1: PDP-11. Методические указания для студентов направления “Информатика и вычислительная техника”/ Сост. В.Н.Негода, И.А.Никищенков. - Ульяновск:, УлГТУ, 1996. - 32 с.

Настоящие методические указания написаны в соответствии с рабочими программами дисциплин "Микропроцессорные системы", "Функциональная организация ЦВМ" и "Специализированные вычислительные системы и комплексы" для студентов направления "Информатика и вычислительная техника". Представлены справочные материалы по функциональной организации ЭВМ семейства PDP-11.

Приведенный материал ориентирован на выполнение различного рода заданий в ходе лабораторных и практических занятий, курсового проектирования, и сдачи экзаменов. Для всех команд приводится алгоритмическое описание их выполнения в ЭВМ.

Подготовлены на кафедре "Вычислительная техника".
Ил.2,табл. 1, библиогр.: 6 назв.

Рецензент:

Одобрено учебно-методической
комиссией ФИСТ

Оглавление

Введение 4
1. Основные термины и язык описания функциональной организации ЭВМ и микропроцессоров 5
1.1. Основные понятия 5
1.2. Язык описания алгоритмов выполнения команд 8
2. ЭВМ с ахитектурой PDP-11 10
2.1. Программно-доступные компоненты 10
2.2. Форматы команд 11
2.3. Способы адресации 12
2.4. Система команд 13
2.5. Ввод-вывод и прерывания 21
2.6. Диспетчер памяти 23
2.7. Основные директивы ассемблера и макрокоманды ОС RT-11 26
Список литературы 31

Введение

Рабочие программы учебных дисциплин "Микропроцессорные системы", “Функциональная организация ЦВМ" и "Специализированные вычислительные системы и комплексы" для студентов направления "Информатика и вычислительная техника" предусматривают изучение архитектур разнообразных микроконтроллеров (МК), микропроцессоров (МП) и ЭВМ на их основе. При выполнении лабораторных заданий, контрольных работ, курсовых проектов и решении экзаменационных задач приходится использовать большой объем фактологического материала. Производительнось учебной деятельности при этом во многом зависит от доступности и понятности студентам справочных данных по форматам и системе команд, способам адресации и программированию на ассемблере. Составители настоящих методических указаний попытались сфорировать справочные данные по различным архитектурам на основе единого стиля описания, что позволяет сократить время на вхождение в архитектуры различных МК, МП и ЭВМ. В первой части рассматривается функциональная организация ЭВМ PDP-11, которая оказала большое влияние на архитектуры многих семейств микропроцессоров. В следующих частях рассматриваются семейства МП и МК фирм Intel и Motorola.

Опыт преподавания вопросов организации ЭВМ и МП в различных дисциплинах показывает, что подавляющему большинству студентов для осмысления архитектуры требуется выполнение достаточно большого объема различного вида практических работ, основными из которых являются:

- ручное ассемблирование, дизассемблирование и анализ машинных программ;
- программирование на языках ассемблера;
- моделирование МК, МП и микропроцессорных устройст и систем.

Содержание методических указаний ориентировано на выполнение этих видов работ.

1. Основные термины и язык описания архитектуры

1.1. Основные понятия

В различных справочниках, учебниках и инженерных изданиях используются разные определения основных понятий, применяемых для описания архитектуры МП, МК и ЭВМ. Приводимый ниже список определений понятий не претендует на какую-то большую точность, строгость или полноту. Правильнее воспринимать этот список как соглашение типа "в данной работе это трактуется так", поэтому читатель должен быть готов к тому, что в других изданиях он может встретить несколько иные толкования.

Функциональная организация МП, МК или ЭВМ - совокупность программно-доступных компонентов, способов адресации, форматов и наборов команд.

Программно-доступные компоненты - любые объекты, содержимое которых может быть использовано или модифицировано с помощью программы.

Машинная программа (machine code) - набор данных в памяти ЭВМ, определяющий выполнение программы процессором ЭВМ.

Машинная команда (instruction) - минимальная единица машинной программы, представляющая собой совокупность данных, определяющих работу процессора при выполнении одной операции.

Адрес команды - адрес первого байта машинной команды.

Счетчик команд (PC - Program Counter, instruction counter, instruction pointer) - регистр процессора, где формируется адрес команды; обычно по мере выборки из памяти частей команды значение счетчика увеличивается на количество прочитанных байтов.

Формат команды (instruction format) - совокупность полей (групп разрядов) машинной команды с указанием местоположения и смысла данных, представляемых каждым полем.

Код операции (КОП, OpCode) - поле команды, определяющее операцию, которая должна быть выполнена по данной команде.

Операнд (operand) - данное, используемое при выполнении команды. Наиболее важными параметрами операнда являются длина (обычно в байтах) и местоположение (регистр процессора, ячейка памяти, порт ввода-вывода).

Операнд-источник - операнд, исходное значение которого используется при выполнении команды.

Операнд-приемник - операнд, местоположение которого совпадает с местоположением результата операции.

Методы адресации (addressing schemes) - методы определения местоположения операнда. Метод адресации может представляться в отдельном поле команды или неявно задаваться кодом операции.

Адресное поле - поле команды, используемое для определения местоположения операнда. Местоположение одного операнда может быть задано содержимым нескольких адресных полей: полем метода адресации, полем номера регистра, хранящего операнд или адрес, полем смещения, используемого в качестве слагаемого для вычисления адреса.

Эффективный адрес (EA - Effective Adress) - адрес операнда, вырабатываемый при обработке адресного поля команды в соответствии с заданным методом адресации. Иногда эффективный адрес задается неявно без использования адресных полей.

Физический адрес - адрес ячейки памяти, где находится операнд. В ЭВМ, где есть аппаратура поддержки распределения памяти между программами, физический адрес формируется из эффективного адреса и набора базовых адресов, указывающих на местоположение данных выполняемой программы.

Регистровая адресация - в адресном поле указывается номер регистра, где находится операнд.

Абсолютная (прямая) адресация (absolute addressing, direct addresing) - в адресном поле указывается EA.

Непосредственная адресация (immediate addresing) - в адресном поле приводится значение операнда.

Смещение (displacement, offset) - содержимое адресного поля, используемое как слагаемое для формирования адреса в индексной адресации.

Индексная адресация (indexed addresing) - EA определяется как сумма смещения и содержимого регистра.

Базовая адресация (base addresing) - аналогично индексной. Базовая и индексная адресации обычно различаются по смыслу обрабатываемых группой команд данных и способам организации доступа к ним.

Косвенный адрес (IA - Indirect Adress, deferred address) задает адрес, по которому находится EA.

Косвенная адресация (indirect addresing) - адресные поля операнда в соответствии с методом адресации определяют косвенный адрес. Косвенная адресация может сочетаться со многими другими адресациями.

Косвенная регистровая адресация - эффективный адрес EA находится в регистре, номер которого задан в адресном поле команды. При этом данный регистр выполняет функции адресного. В некоторых архитектурах функции адресного регистра могут выполнять только вполне определенные регистры. Имеются архитектуры, где любой регистр может выполнять функции адресного.

Неявная адресация (implied addresing, inherent addresing) - адресация, при которой местоположение операнда задается кодом операции без использования адресных полей.

Стек (stack) - область памяти, обращение к которой выполняется через стековую адресацию.

Стековая адресация - адресация с использование специального регистра - указателя стека (**SP** - **S**tack **P**ointer). Стековая адресация используется для занесения операндов в стек в одном порядке и извлечения в обратном порядке. До занесения операнда Op в стек содержимое SP уменьшается, затем используется в качесте EA операнда-приемника. При извлечении Op из стека в качестве EA операнда-источника используется содержимое SP, затем SP увеличивается. Таким образом, значение SP всегда указывает на последнее данное, находящееся в стеке. Стековая адресация часто бывает неявной - например в командах входа в подпрограмму и выхода из подпрограммы, где операндом является адрес

возврата. В некоторых ЭВМ имеется возможность организовать несколько стеков, используя в качестве указателей адресные регистры.

Косвенная регистровая адресация с автоувеличением или автоуменьшением содержимого регистра - адресация, при которой обращение к операнду по ЕА, равному содержимому используемого регистра, сопровождается автоматическим уменьшением или увеличением его содержимого на длину операнда. Различают предмодификацию (изменение содержимого регистра до обращения к операнду) и постмодификацию (изменение после обращения). Взаимнопротивоположный порядок модификации для адресации с автоувеличнием и автоуменьшением позволяет организовать стековую адресацию на основе любого регистра.

Адрес входа (entry point) - адрес первой команды подпрограммы или программы обработки прерывания.

Адрес возврата (return address) - адрес команды, перед выполнением которой вызвана подпрограмма или возникло прерывание.

Прерывание (interrupt) - приостановка выполнения текущей программы с возможностью после обработки прерывания продолжить ее выполнение. При входе в прерывание всегда сохраняется адрес возврата.

Вектор прерываний (interrupt vector) - набор данных, определяющих адрес входа в подпрограмму обработки прерываний и, возможно, новые значения данных, характеризующих состояние процессора.

Адрес вектора прерываний - адрес первого байта или слова вектора прерываний.

Язык машинных кодов - запись машинной программы в виде последовательности восьмиричных или шестнадцатиричных цифр, где группа цифр задает значение байта или слова машинной программы.

Язык ассемблера (assembly language) - машинно-ориентированный язык записи программы в виде последовательности операторов, каждый из которых представляет либо директиву ассемблера, либо машинную команду. В языке ассемблера коды операции, операнды, методы адресации представляются в мнемоническом виде с использованием имен констант, переменных, адресов перехода. Для различных архитектур ЭВМ, МП и МК используются различные языки ассемблера.

Ассемблер - программа, выполняющая трансляцию программы на языке ассемблера (ассемлер-программы) в машинный код.

Мнемокод (mnemonic code) - мнемоническая запись кода операции машинной команды, используемая в языке ассемблера.

1.2. Язык описания алгоритмов выполнения команд

При описании алгоритмов выполнения команд в данных методических указаниях используются операции логических и арифметических выражений языка программирования Си:

! - логическое отрицание;

~ - поразрядное отрицание;

& - поразрядное И в двухместной операции; определение адреса в одноместной;

&& - логическое И;

| - поразрядное ИЛИ;

|| - логическое ИЛИ;

^ - поразрядное исключающее ИЛИ;

++ - увеличение на единицу для данных и на длину операнда для адресов;

-- - уменьшение на единицу для данных и на длину операнда для адресов;

- - вычитание в двухместной операции; изменение знака операнда в одноместной операции;

+ - сложение;

***** - умножение в двухместной операции; обращение по адресу в одноместной операции;

/ - деление;

% - деление по модулю;

<< - сдвиг влево;

>> - сдвиг вправо;

==, **!=**, **<**, **>**, **<=**, **>=** - отношения равно, неравно, меньше, больше, меньше или равно, больше или равно;

= - присваивание;

-=, **+=**, ***=**, */=*, **%=**, **&=**, **^=**, **|=**, **<<=**, **>>=** - выполнение двухместной операции между левой и правой частями выражения с присвоением результата операнду, указанному в левой части;

, - операция запятая.

условие **?** *выраж1* **:** *выраж2* - условное выражение; если *условие* истинно, то вычисляется выражение *выраж1*, иначе - *выраж2*.

Кроме перечисленных знаков операций используются обозначения:

*число1***..***число2* - диапазон чисел от значения число1 до значения *число2* включительно;

*Op***[***номер***]** - значение разряда с заданным номером в операнде *Op*;

*Op***[ст]** - значение старшего разряда операнда;

*Op***[мл]** - значение младшего разряда операнда;

*Op***[***номер1***..***номер2***]** - выделение диапазона разрядов; *номер1* задает старший, а *номер2* - младший разряд из указанного диапазона;

[*] - каждый разряд операнда(применить операцию к каждому разряду); например, **A[*] = 1** - занести единицы во все разряды операнда A.

данное1•данное2 - конкатенация (объединение в единую последовательность) двух данных; *данное1* образует старшие разряды формируемого таким образом более длинного данного;

*****выражение* - содержимое ячейки памяти, адрес которой определяется выражением;

Op - значение операнда;

EA - эффективный адрес;

IA - косвенный адрес;

AV - адрес вектора прерываний.

2. ЭВМ с архитектурой PDP-11

ЭВМ с архитектурой PDP-11(фирма Digital Equimpent Corporation) в течении двух десятков лет были наиболее популярными мини-ЭВМ для управления технологическими процессами (в России это СМ-3, СМ-4, СМ-1420, СМ-1600, СМ-1300, Электроника100-25, Электроника-60 и др.). В России в настоящее время находятся в эксплуатации довольно много систем управления технологическими процессами на основе данной архитектуры. Архитектура PDP-11 оказала большое влияние на организацию современных микропроцессоров и ПЭВМ.

2.1. Программно-доступные компоненты

Программно-доступные компоненты базовой модели PDP-11 представлены на рис.1.

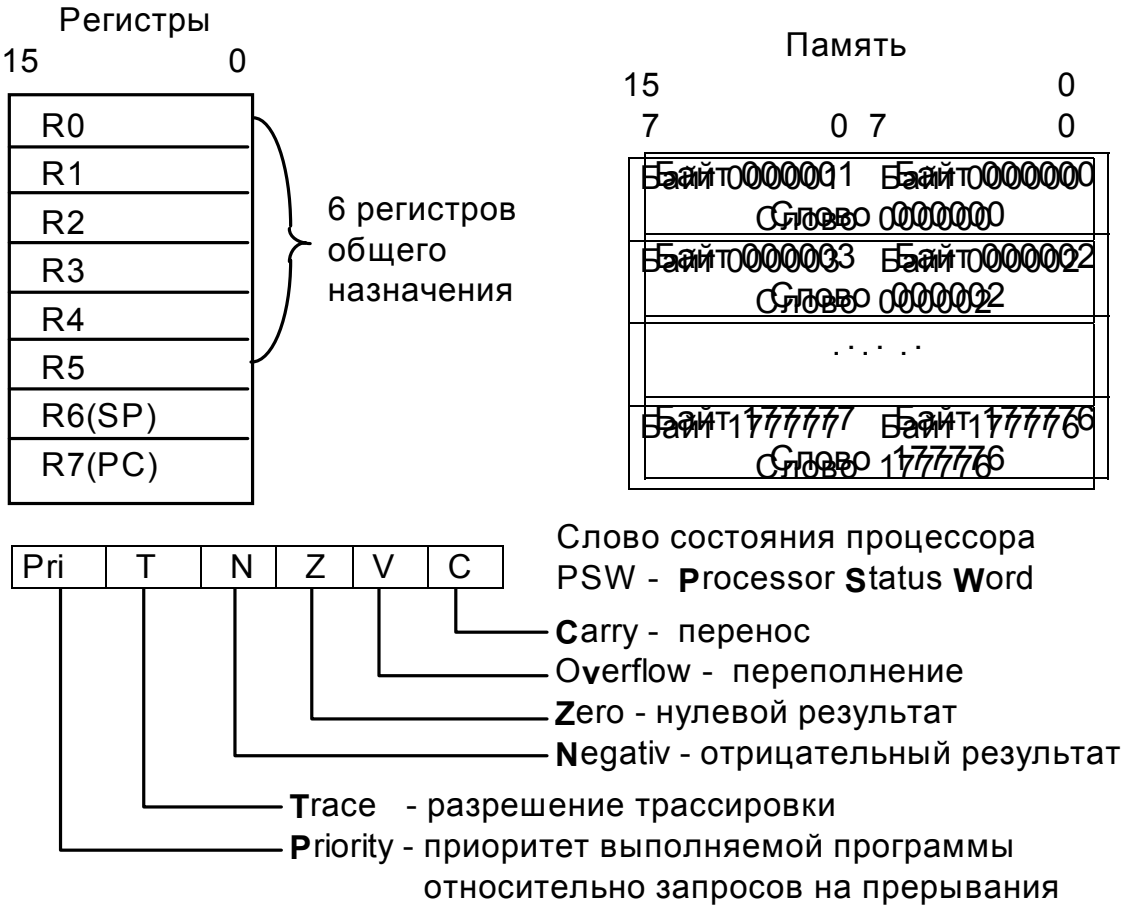


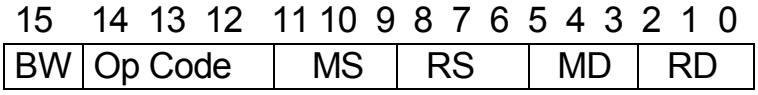
Рис. 1. Программно-доступные компоненты базовой архитектуры PDP-11

Операнды, размещаемые в регистрах, могут быть шестнадцатиразрядными (слова) и восьмиразрядными (байты). При этом восьмиразрядные операнды размещаются в младшем байте регистра. Байт в памяти может иметь любой адрес. Слово может иметь только четный адрес. Указатель стека SP и счетчик команд PC являются указателями на слова памяти, поэтому должны быть четными. Адреса байтов и слов на рис.1 приведены в восьмеричной системе счисления. В PDP-11 нет специальных команд ввода-вывода. Объекты ввода-вывода адресуются как ячейки памяти. Диапазон адресов 160000..177776 обычно используется для обращения к внешним устройствам. Здесь и далее для кодов и чисел по умолчанию используется восьмеричная система счисления.

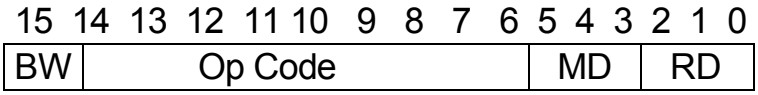
2.2. Форматы команд

Команда PDP-11 может размещаться в 1, 2 или 3 словах. Второе и третье слова всегда являются адресными и могут содержать непосредственный операнд, абсолютный адрес, смещение для индексной адресации. Первое слово определяет операцию, длину операнда и методы адресации. Основными являются следующие форматы первых слов команды:

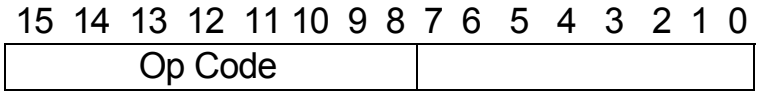
Двухадресные команды:



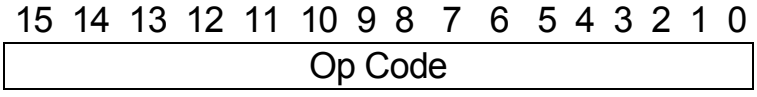
Одноадресные команды:



Команды ветвления:



Безадресные команды:



Здесь: BW - признак длины операнда(Byte/Word - байт(1)/слово(0));

MS, MD - методы адресации операнда-источника(Sourse) и операнда-приемника(Destination)

RS, RD - номера регистров, используемых при адресации операнда-источника и операнда-приемника.

2.3. Способы адресации

Способы адресации PDP-11 приведены в таблице 1.

Таблица 1. Способы адресации PDP-11

Метод-регистр	Мнемон. обознач.	Название	Местоположение операнда
0k	Rk	Регистровая	операнд = Rk
1k	@Rk	Регистр. косвенная	EA = Rk
2k	(Rk)+	Автоинкрементная	EA = Rk++
3k	@(Rk)+	Автоинкр. косвенная	IA = Rk++
4k	-(Rk)	Автодекрементная	EA = --Rk
5k	@-(Rk)	Автодекрем. косвен	IA = --Rk
6k	X(Rk)	Индексная	EA = Rk + *PC++
7k	@X(Rk)	Индексная косвенная	IA = Rk + *PC++
27	#Op	Непосредственная	операнд = *PC++
37	@#EA	Абсолютная	EA = *PC++
67	EA	Относительная	EA = PC + *PC++
77	@IA	Косвенно-относит	IA = PC + *PC++

Адресация одного операнда определяется парой "Метод - Номер регистра". Код метода адресации занимает в команде 3 разряда и представлен в табл. 1 одной восьмиричной цифрой. Номер регистра занимает также 3 бита и обозначен символом k. Последние 4 метода в таблице являются частным случаем соответственно автоинкрементной, автоинкрементной косвенной, индексной и индексной косвенной адресаций при k = 7, то есть при использовании счетчика команд. Адресуемые этими методами объекты непосредственно примыкают к команде и если рассматривать их как адресные поля, то получаются непосредственная, абсолютная, относительная и косвенно-относительная адресации в соответствии с определениями раздела 1.1.

Модификация Rk++ и --Rk выполняется на 1 для команд обработки байтов при автоинкрементной и автодекрементной адресации и k<=5. Во всех других случаях модификация выполняется на 2. Регистры PC и SP всегда рассматриваются как указатели на слова, а не байты, поэтому при описании действия PDP-11 в унарных операциях PC++, --SP и им подобных имеется ввиду модификация указателя на 2.

2.4. Система команд

Набор команд PDP-11 относительно невелик и, в то же время, обеспечивает широкие функциональные возможности ЭВМ в части обработки данных с самой различной организацией. Это происходит прежде всего за счет того, что для подавляющего большинства команд допустимы все возможные сочетания способов адресации операндов и используемых при этом регистров. Это свойство функциональной организации ЭВМ называется ортогональностью.

В описании команд, приводимом ниже, используется восьмиричное представление кодов команд. Причем для обозначения адресных полей, задающих местоположение операнда-источника или операнда-приемника используются буквосочетания SS или DD. В конкретной машинной команде на месте первой буквы фигурирует номер метода адресации, а на месте второй - номер регистра. В алгоритмическом описании действия команды операнд, адресуемый полями DD, обозначается символом D, а операнд, адресуемый полями SS - символом S. Для операнда, который адресуется только через регистровую адресацию, символ R в коде команды обозначает номер регистра, а в описании действия - содержимое регистра. Значение двойного слова, размещаемого в регистрах R(старшие разряды) и R+1(младшие разряды), обозначается Rp. Символ "*" в коде команды означает, что данная команда может обрабатывать либо байты (старший бит равен 1), либо слова(старший бит равен 0). Мнемокод команды обработки байта имеет префикс "B". Например: MOV - пересылка слова, MOVB - пересылка байта.

Система команд PDP-11 и правила их выполнения ориентированы прежде всего на следующие данные:

- символы(байты);
- целые числа со знаком из диапазона -128..+127(байты);
- целые числа без знака из диапазона 0..255(байты);
- целые числа со знаком из диапазона -32768..+32767(слова);
- целые числа без знака из диапазона 0..65535(слова);
- массивы символов, строки текста и массивы чисел;
- обработка отдельных разрядов(битовых полей).

Все знаковые числа внутри ЭВМ представлены в дополнительном коде. Старший разряд знаковый. Поскольку знаковый разряд двоичного числа в дополнительном коде участвует в арифметических операциях так же как и другие разряды, наличие знака не влияет на правила выполнения арифметических операций. Различение происходит в признаках результата. Если выполняется сложение двух беззнаковых чисел, то переполнение обнаруживается в признаке переноса C. Если же складываются знаковые числа, то процессор фиксирует переполнение в признаке V, а знак результата в N. Если сравниваются два числа без знака, то соотношения "меньше", "больше", "меньше или равно" и "больше или равно" определяются признаками Z и C. Если же сравниваются числа со знаком, то эти соотношения определяются признаками N, Z и V.

В таблице 2 представлены команды обработки данных. Установка признаков результата в этой таблице отображается одним символом из следующего набора:

"0" либо "1" - установка в значение 0 или 1;

"-" - неизменное значение;

"+" - установка признака в соответствии с результатом Res выполненной операции по следующим правилам:

Z = (Res == 0)? 1 : 0;

N = Res[ст];

C устанавается в 1, если есть перенос/заем для арифметических операций или при сдвиге выдвигаемый из значения операнда бит равен 1;

V устанавливается в 1, если при сдвиге происходит модификация знакового разряда операнда, либо результат арифметической операции не попадает в диапазон числа со знаком (-128..+127 при обработке байтов, -32768..+32767 при обработке слов).

Обработка массивов байтов и слов эффективно строится на основе автоинкрементной и автодекрементной адресации, которым соответствуют следуюшие конструкции языка Си:

*P++ - использование элемента, адресуемого указателем P с последующим продвижением указателя к следующему элементу массива;

*--P - продвижение указателя P к предыдущему элементу массива и последующее использование этого элемента.

Если для реализации указателя P используется регистр, то одна машинная команда с автоинкрементной или автодекрементной адресацией выполняет как обращение к элементу массива, так и модификацию указателя.

Для обработки отдельных разрядов и битовых полей используются команды логических операций и сдвигов. Команды BIS и BIC позволяют установить в 1 и сбросить в 0 разряды, значения которых равны 1 в операнде S. Комнда XOR позволяет инвертировать заданные разряды регистра R. Например:

```
BIS     #3,   R0   ; установка в 1 разрядов R0[1..0]
BIC     #4,   R1   ; очистка разряда R1[2]
XOR     #7,   R2   ; инвертирование разрядов R2[2..0]
```

Таблица 2. Одноадресные и двухадресные команды обработки данных.

Мне-мокод	Код	N Z V C	Операция	Действия
Команды пересылки и установки признаков результата:				
CLR	*050DD	0 1 0 0	Очистка	D = 0`
MOV	*1SSDD	+ + 0 -	Пересылка	D = S
SWAB	0003DD	+ - 0 0	Перестановка байтов	D=D[7..0]•D[15..8]
MFPS	1067DD	- - - -	Чтение PSW	D = PSW
MTPS	1064SS	+ + + +	Запись PSW	PSW = S
TST	*057DD	+ + 0 0	Проверка	D = D
Логические операции:				
COM	*051DD	+ + 0 1	Инвертирование	D = ^D
BIT	*3SSDD	+ + 0 -	Проверка разрядов	S & D
BIC	*4SSDD	+ + 0 -	Очистка разрядов	D &= ^S
BIS	*5SSDD	+ + 0 -	Логическое сложение	D |= S
XOR	074RDD	+ + 0 -	Исключающее ИЛИ	D ^= R
Арифметические операции и сдвиги:				
INC	*052DD	+ + + -	Увеличение на 1	D++
DEC	*053DD	+ + + -	Уменьшение на 1	D--
CMP	*2SSDD	+ + + +	Сравнение	S - D
ADD	06SSDD	+ + + +	Сложение	D += S
SUB	16SSDD	+ + + +	Вычитание	D -= S
ADC	*055DD	+ + + +	Добавление переноса	D += C
SBC	*056DD	+ + + +	Вычитание переноса	D -= C
NEG	*054DD	+ + + +	Изменение знака	D = 0 - D
SXT	0067DD	+ + + +	Расширение знака	D[*] = N
MUL	070RSS	+ + 0 +	Умножение	Rp = R * S
DIV	071RSS	+ + + +	Деление	Rp /= S
ASR	*062DD	+ + + +	Арифм. сдвиг вправо	D•C = D[ст] •D
ASL	*063DD	+ + + +	Арифм. сдвиг влево	C•D = D•0
ROR	*060DD	+ + + +	Цикл. сдвиг вправо	D•C = C•D
ROL	*061DD	+ + + +	Цикл. сдвиг влево	C•D = D•C
ASH	072RSS	+ + + +	Арифм. сдвиг на за-данное число разр.	S>0 ? R <<= S: R >>= -S
ASHC	073RSS	+ + + +	Арифм. сдвиг двой-ного слова	S>0 ? Rp <<= S: Rp >>= -S

Команды сдвига позволяют умножить и разделить на 2 в целой степени и позиционировать разряды в нужное положение. Например:

```
; Прибавление OP1.X += OP2,
; где: X - битовое поле, размещенное в OP1[7..4],
;       OP2 - операнд, размещенный в слове
; Считается, что переполнение поля X невозможно
```

```
MOV     OP2,    R0    ; R0 = OP2
ASH     #4,     R0    ; позиционирование копии OP2 против поля X
ADD     R0,OP1  ; подсуммирование копии OP2 к полю X

; Очистка бита с номером NUM в слове W
MOV     #1,     R0    ; создание маски с одной единицей
ASH     NUM,    R0    ; позиционирование 1 в разряд R0[NUM]
BIC     R0,     W     ; очистка бита W[NUM]
```

Для арифметической обработки данных повышенной точности используются команды ADC, SBC, SXT. Например, пусть имеются 16-разрядная переменная A16 и 32-разрядные переменные A32 и B32, объявленные в Си следующим образом:

```
short  A16;
long  A32, B32;
```

Пусть при обработки этих данных выполняются операторы:

```
A32 = (long)A16;
B32 +=  A32;
```

Эти операторы могут быть реализованы на ассемблере следующим образом:

```
; Преобразование 16-разрядного операнда A16 в 32-разрядный A32
MOV     A16,    A32     ; пересылка младшего слова
SXT     A32+2           ; формирование старшего слова операнда
; Подсуммирование A32 к 32-разрядному B32
ADD     A32,    B32     ; подсуммирование младшего слова
ADC     B32+2           ; подсуммирование переноса
ADD     A32+2,  B32+2   ; подсуммирование старшего слова
```

Изменение естественного порядка следования команд выполняется по специальным командам перехода. К этой группе команд в PDP-11 относятся команды безусловного перехода JMP, организации цикла SOB, обращения к подпрограмме JSR, возврата из подпрограммы RTS, возврата с восстановлением стека MARK, организации программных прерываний EMT, TRAP, IOT, BPT, RTI и RTT, а также большая группа команд ветвления BR, BEQ и др. Все эти команды приведены в таблицах 3 и 4.

Команды безусловного перехода JMP и перехода к подпрограмме в качестве адреса перехода используют эффективный адрес операнда. Поэтому для этих команд недопустима регистровая адресация. При входе в подпрограмму по команде JSR Ri, Adr содержимое регистра Ri сохраняется в стеке, в регистр Ri заносится адрес возврата и в PC заносится адрес перехода. При возврате из подпрограммы по команде RTS адрес возврата переносится из Ri в PC и старое содержимое Ri восстанавливается из стека. Поэтому команда RTS должна ссылаться на тот же Ri, что и команда вызова данной подпрограммы. Чаще всего в качестве регистра используется PC, то есть команды имеют вид JSR PC, *адрес* и RTS PC, а адрес возврата сохраняетеся в стеке.

Таблица 3. Команды управления программой и прерываниями

Мне-мокод	Код	N Z V C	Операция	Действия
Безусловный переход и организация циклов и подпрограмм:				
JMP	0001DD	- - - -	Безусловный переход	PC = &D
SOB	077RNN	- - - -	Организация цикла	if(--R != 0) PC -= 2*NN
JSR	004RDD	- - - -	Обращение к под-программе	*(--SP) = R, R=PC, PC=&D
RTS	00020R	- - - -	Возврат из п/п	PC=R,R=*SP++
MARK	0064NN	- - - -	Восстановление указателя стека	SP = PC+2*NN PC = R5 R5 = *SP++
Программные прерывания:				
EMT	104000 ..104377	из вектора	Прерывание для систем. программ с адресом вектора AV = 30	*--SP = PSW *--SP = PC PC = *(AV) PSW = *(AV+2)
TRAP	104400 ..104777	- " -	Прерывания пользов.	то же, но адр. вектора AV=34
IOT	000004	- - - -	Прерыв. для вв/выв.	AV=20
BPT	000003	- - - -	Прерыв. трассировки	AV=14
RTI	000002	из стека	Возврат из прерывания	PC = *SP++ PSW = *SP++
RTT	000006	из стека	Возврат из прерывания трассировщика	PC = *SP++ PSW = *SP++

Команды ветвления обеспечивают переход от текущей команды к заданной точке программы при истинности условия. Первый байт команды содержит код операции. Второй байт - смещение к точке перехода от текущего значения счетчика команд PC (то есть это команда с относительной адресацией). Смещение указывается в количестве слов, на которое нужно перейти и процессор выполняет увеличение PC на величину удвоенного смещения. Смещение рассматривается как байт со знаком, поэтому перед удвоением смещения выполняется его приведение к формату 16-разрядного целого со знаком путем расширения знака. При вычислении адреса перехода в PC находится продвинутый адрес, указывающий на следующую команду за командой ветвления. Поскольку смещение занимает всего один байт, длина “прыжка” по команде ветвления не может быть более 256. Если необходимо перейти на большее расстояние, то используется сочетание команды ветвления и команды JMP.

Таблица 4. Команды ветвления

Если условие ветвления выполняется, то PC += 2*(short)Disp, где: Disp - младший байт команды.

Мне-мокод	Диапазон кодов результата	Наименование условия	Значение признаков
BR	000400..000777	Безусловно	нет
BNE	001000..001377	Не нуль;неравенство	Z == 0
BEQ	001400..001777	Нуль; равенство	Z == 1
BPL	100000..100377	Плюс	N == 0
BMI	100400..100777	Минус	N == 1
BVC	102000..102377	Нет переполнения	V == 0
BVS	102400..102777	Есть переполнение	V == 1
BCC	103000..103377	Нет переноса	C == 0
BCS	103400..103777	Есть перенос	C == 1
Ветвления по неравенству чисел со знаком:			
BLT	002400..002377	Меньше	N ~ V = 1
BGT	003000..003377	Больше	Z | (N ~ V) == 0
BLE	003400..003777	Меньше или равно	Z | (N ~ V) == 1
BGE	002000..002377	Больше или равно	N ~ V == 0
Ветвления по неравенству чисел без знака:			
BLO	103400..103777	Меньше	C == 1
BHI	101000..101377	Больше	Z | C == 0
BLOS	101400..101777	Меньше или равно	Z | C == 1
BHIS	103000..103377	Больше или равно	C == 0

Ниже приводится фрагмент листинга ассемблера, содержащий команды ветвления. Директива ассемблера ". = <число>" назначет адрес, по которому должна транслироваться следующая строка.

```
Адрес    Код                    Метка  Мнемокод команды
-------------------------------------------------------------------------------
                                       . = 1000
001000   022720   000100         L1:    CMP  #64., (R0)+
001004   001403                         BEQ  L2
001006   100775                         BMI  L1
001010   000167   000764                JMP  L20
001014   000167   000764         L2:    JMP  L21
                                       . = 2000
002000   062700   000010         L20:   ADD  #10,  R1
002004   005015                 L21:   CLR  (R5)
-------------------------------------------------------------------------------
```

В таблице 5 приведены команы общего управления машиной. По команде HALT выполняется останов выполнения программы. Команда WAIT позволяет приостановить программу до поступления требования прерывания, подлежащего обработке (см. п.2.5.). Пустая команда NOP используется для исключения из работы фрагмента программы без модификации других фрагментов (“забой” машинных команд пустой командой) и для формирования задержки времени перед выполнением действий с периферийными устройствами.

Таблица 5. Команды управления

Мне-мокод	Код	Операция	Действия
HALT	000000	Останов	Переход на связь с консолью
WAIT	000001	Ожидание	Предоставление канала ВУ
RESET	000005	Сброс	Начальная установка ВУ
NOP	000240	Нет операции	Пустая команда

Признаки результата могут быть изменены специальными командами установки и сброса, которые представлены в таблице 6.

Таблица 6. Команды изменения признаков в PSW

Мне-мокод	Код	Действие	Мне-мокод	Код	Действие
CLN	000250	N = 0	SEN	000270	N = 1
CLZ	000244	Z = 0	SEZ	000264	Z = 1
CLV	000242	V = 0	SEV	000262	V = 1
CLC	000241	C = 0	SEC	000261	C = 1
CCC	000257	N=Z=V=C=0	SCC	000277	N=Z=V=C=1

Некоторые учебные задания по изучению функциональной организации ЭВМ предполагают анализ машинной программы с предварительным ручным дизассемблированием, т.е. формированием ассемблер-программы, соответствующей анализируемому машинному коду. При выполнении такой работы целесообразно пользоваться списком команд, упорядоченным по коду операции. Этот список представлен в таблице 7.

Таблица 7. Список команд в порядке возрастания кодов операций

Код	Мне-мокод	Табл	Код	Мне-мокод	Табл	Код	Мне-мокод	Табл
000000	HALT	3	000001	WAIT	3	000002	RTI	3
000003	BPT	3	000004	IOT	3	000005	RESET	5
000006	RTT	3	0001DD	JMP	3	00020R	RTS	3
000240	NOP	5	000241	CLC	6	000242	CLV	6
000244	CLZ	6	000250	CLN	6	000257	CCC	6
000261	SEC	6	000262	SEV	6	000264	SEZ	6
000270	SEN	6	000277	SCC	6	0003DD	SWAB	3
000[4..7]XX	BR	4	001[0..3]XX	BNE	4	001[4..7]XX	BEQ	4
002[0..3]XX	BGE	4	002[4..7]XX	BLT	4	003[0..3]XX	BGT	4
003[4..7]XX	BLE	4	004RDD	JSR	3	0050DD	CLR	2
0051DD	COM	2	0052DD	INC	2	0053DD	DEC	2
0054DD	NEG	2	0055DD	ADC	2	0056DD	SBC	2
0057DD	TST	2	0060DD	ROL	2	0061DD	ROR	2
0062DD	ASR	2	0063DD	ASL	2	0064NN	MARK	3
0067DD	SXT	2	01SSDD	MOV	2	02SSDD	CMP	2
03SSDD	BIT	2	04SSDD	BIC	2	05SSDD	BIS	2
06SSDD	ADD	2	070RSS	MUL	2	071RSS	DIV	2
072RSS	ASH	2	073RSS	ASHC	2	074RDD	XOR	2
077RNN	SOB	2	100[0..3]XX	BPL	4	100[4..7]XX	BMI	4
101[0..3]XX	BHI	4	101[4..7]XX	BLOS	4	102[0..3]XX	BVC	4
102[4..7]XX	BVS	4	103[0..3]XX	BCC	4	103[4..7]XX	BCS	4
104[0..3]XX	EMT	3	104[4..7]XX	TRAP	3	1050DD	CLRB	2
1051DD	COMB	2	1052DD	INCB	2	1053DD	DECB	2
1054DD	NEGB	2	1055DD	ADCB	2	1056DD	SBCB	2
1057DD	TSTB	2	1060DD	ROLB	2	1061DD	RORB	2
1062DD	ASRB	2	1063DD	ASLB	2	1064SS	MTPS	2
1067DD	MFPS	2	11SSDD	MOVB	2	12SSDD	CMPB	2
13SSDD	BITB	2	14SSDD	BICB	2	15SSDD	BIS	2
16SSDD	SUB	2						

Пример дизассемблирования фрагмента машинной программы приведен в таблице 8.

В этом фрагменте директива размещения слов данных .WORD (см. описаниение директивы в 2.6) сформирована в результате анализа смысла последовательностей машинных команд. Поскольку команды с адреса 1006 по адрес 1014 представляют собой цикл суммирования массива чисел, адресуемый указателем R1, загрузка значения 1022 в данный регистр перед входом в цикл позво-

ляет сделать вывод, что это базовый адрес массива. Проверка адреса конца массива в коменде CMP и команда перехода по адресу 1034 свидетельствуют, что обрабатываемые данные размещены именно с адреса 1022 по адрес 1032.

Таблица 8. Пример дизассемблирования фрагмента программы

Адрес	Код	Мнемокод
1000	005000	CLR R0
1002	012701	MOV #1022, R1
1004	001022	
1006	062100	ADD (R1)+, R0
1010	022701	CMP #1032, R1
1012	001032	
1014	100374	BPL 1006
1016	000167	JMP 1034
1020	000012	
1022	000001	.WORD 1, 12, 15, 3, 5
1024	000012	
1026	000015	
1030	000003	
1032	000005	
1034	016767	MOV 1022, 1024
1036	177762	
1040	177762	

2.5. Ввод-вывод и прерывания

В PDP-11 нет специальных команд ввода-вывода. Объекты внешних устройств (ВУ) включаются в единое адресное пространство ЭВМ. Наиболее часто такими объектами являются регистры ввода-вывода (РВВ). Устройство сопряжения (интерфейсный модуль) с внешним устройством обычно включает в себя несколько РВВ и селектор адреса, который преобразует адрес обращения к РВВ в соответствующие сигналы управления ими. Иногда термин "регистр" применяют к объектам, которые на самом деле регистрами не являются. Например, информация о достижении режущим инструментом границы допустимой зоны перемещения, получаемая при замыкании контактных пластин (так называемый "концевой выключатель"), представляет из себя один бит, но этот один бит может выдаваться в процессор при обращении к РВВ, то есть представлять целое слово. Так же этот бит может быть включен в группу данных о состоянии устройства, например: кнопка "Стоп" инженерного пульта; двухразрядный регистр, значение в котором содержит код направления перемещения инструмента в плоскости и т.д., представляя собой слово состояния устройства, которое выдается на шину при обращении к одному адресу РВВ. Допустим адрес этого объекта равен 165000 и состояние концевого выключателя выдается на старший разряд системной шины. При этом ветвление в программе в зависимости от того, замкнут ли этот контакт, может выполняться например так:

```
TST    @#165000
BMI    ENDTBL    ; переход по срабатыванию концевого выключателя
; продолжение программы при разомкнутом контакте.
```

Если РВВ доступен как по чтению, так и по записи, то содержимое этого регистра может модифицироваться по командам обработки данных.

Например:

```
; Зажечь светодиод, подключенный к выходу 5-го разряда РВВ
; с адресом 165002
LD = 165002
BIS    #40,  @#LD

; Погасить светодиод, подключенный к выходу 0-го разряда
BIC    #1,   @#LD
```

Для организации ввода-вывода часто один из РВВ блока сопряжения с ВУ играет роль регистра состояния-управления. Чтение данных из этого регистра позволяет узнать состояние процесса обмена данными и ВУ. Запись в этот РВВ обеспечивает задание режимов и инициирование операций. Для многих регистров состояния используются общие соглашения, согласно которым в 15-м разряде (старший разряд слова) представлен признак ошибки операции ввода-вывода, в 7-м разряде (старший разряд младшего байта РВВ) - признак готовности ВУ к обмену данными, в 6-м разряде - разрешение прерывания.

Для системных внешних устройств PDP-11 существуют соглашения по адресации РВВ. Простейшие УВВ имеет следующие адреса:

177550, 177552 - регистры состояния и данных фотосчитывателя
177554, 177556 - регистры состояния и данных перфоратора
177560, 177562 - регистры состояния и данных клавиатуры консоли
177564, 177566 - регистры состояния и данных вывода на консоль
177514, 177516 - регистры состояния и данных принтера

Фрагмент программы вывода на принтер строки STR, завершающейся байтом 0, может быть таким:

```
    MOV     #STR,       R0 ; установить указатель R0 в начало строки
    ; Опрашивать регистр состояния в ожидании готовности
    ; или появления ошибки (конец бумаги, выключение принтера)
L:  BIT     #100200,    @#177514    ; Бит 15 - ошибка, бит 7 - готовность
    BEQ     L ; Ожидать, если нет ни ошибки, ни готовности
    ; Проверить наличие ошибки
    TST     @#177514    ; Признак в знаковом разряде
    BMI     ERROR       ; переход к реакции на ошибку
    ; Вывести символ
    MOVB    (R0)+,      @#177516
    BNE                 L           ; Продолжить, если не завершающий 0
    ; Продолжение после вывода строки
```

Прерывания в PDP-11 в зависимости от инициатора делят на программные и аппаратные. Прерывания по инициативе программы возникают, когда процессор выполняет одну из команд EMT, TRAP, IOT, BPT. Аппаратное прерывание возникает, когда процессор перед началом выборки следующей команды обнаружил такой запрос на прерывание, уровень которого выше текущего приоритета программы (PSW[7..5]). Уровень запроса определяется номером физической линии требования прерывания, к которой подключено устройство. В PDP-11 имеется 4 линии требования с номерами 4..7. Если текущий приоритет PSW[7..5] == 7, то запрещены все прерывания, если PSW[7..5] == 6, то разрешены только прерывания, требования которых поступают по линии с номером 7. Если PSW[7..5] < 4, то разрешены любые прерывания. В некоторых ЭВМ с архитектурой PDP-11 имеется только линия требования прерывания с номером 4. При этом разряд PSW[7] имеет смысл признака запрета прерывания.

При входе в прерывание выполняются действия:

- сохранение PSW в стеке (*--SP = PSW);
- сохранение адреса возврата в стеке (*--SP = PC);
- занесение стартового адреса программы обработки прерывания в PC из первого слова вектора прерывания (PC = *AV);
- занесение нового значения PSW из второго слова вектора прерывания(PC = *(AV+2)).

Выход из подпрограммы обработки прерывания происходит по команде RTI, выполнение которой заключается в восстановлении из стека адреса возврата (PC = *SP++) и старого значения PSW (PSW = *SP++).

Командам вызова программных прерываний соответствуют вполне определенные адреса двухсловных векторов прерываний. Адрес вектора аппаратного прерывания передается в процессор блоком сопряжения с ВУ, когда тот получает от процессора подтверждение, что его требование прерывания удовлетворяется. Вектора в PDP-11 занимают пространство памяти с адресами в диапазоне 0..376. Для ЭВМ этого семейства приняты следующие соглашения по адресации основных векторов прерывания: 60 - клавиатура консоли, 64 - вывод на консоль, 100 - системный таймер, 200 - принтер.

2.6. Диспетчер памяти

В большинстве модификаций PDP-11 имеется диспетчер памяти, который позволяет использовать до 4 мегабайт памяти и организовать многозадачную работу программного обеспечения таким образом, чтобы данные одной задачи могли быть защищены от вмещательства команд другой задачи. Диспетчер памяти выполняет преобразование эффективного адреса EA, вырабатываемого процессором при обработке адресных полей команды, в физический адрес памяти по схеме, представленной на рис. 2.

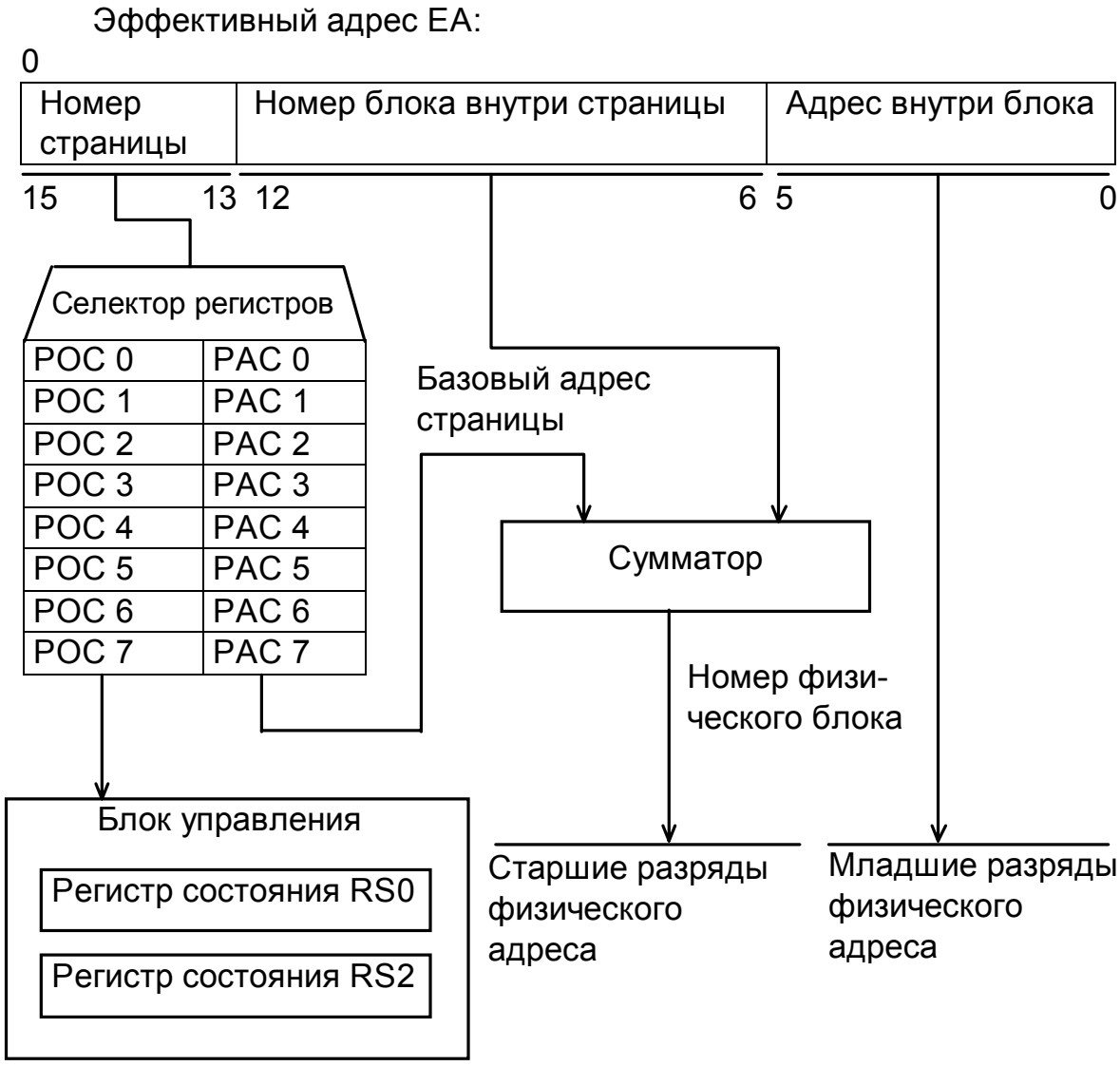


Рис. 2. Схема формирования физического адреса в диспетчере памяти PDP-11

При включенном режиме диспетчеризации памяти адресное пространство эффективного адреса разбивается на страницы. Номером страницы служат три старших разряда EA. Этот номер адресует регистр адреса страниц PAC[i], из которого извлекается базовый адрес страницы. Базовый адрес всегда указывает на начало 64-байтного блока физической памяти, поэтому в младших разрядах имеет 6 нулей. Естественно, что эти нули в PAC не хранятся и не участвуют в суммировании базового адреса с номером блока внутри страницы (разряды EA[12..6]). В моделях ЭВМ с 4 Мб памяти PAC имеет разрядность 16. Сумма базового адреса с номером блока образует 16-разрядный номер блока физической памяти. Этот номер образует старшие 16 разрядов физического адреса, т.е. FA[21..6]. В качестве 6 младших разрядов используется EA[5..0], т.е. FA[21..0] = (PAC + EA[12..6])•EA[5..0].

Для поддержки эффективных механизмов управления памятью в многозадачной среде каждая страница сопровождается специальным описанием, хранящемся в регистре описания страницы РОС. Этот регистр имеет формат:

РОС[0] = 0;

РОС[2..1] - поле контроля доступа:

00,10 - запрет обращения;

01 - страница доступна только для чтения;

11 - страница доступна для чтиния и записи;

РОС[3] - направление расширения страницы:

0 - расширение в сторону увеличения адресов;

1 - расширение в сторону уменьшение адресов(стек);

РОС[5..4] = 00;

РОС[6] - контроль записи: 0 - записи не было, 1 - запись в страницу была; это поле позволяет избежать вывода во внешний накопитель тех страниц, которые играют роль буфера ввода-вывода и немодифицированы;

РОС[7] = 0;

РОС[14..8] - размер страницы в блоках по 64 байта.

При работе диспетчера памяти выделяются 2 режима работы: системный (код 00) и пользовательский (код 11). В диспетчере памяти имеются 2 комплекта РАС и РОС: один для системного, другой для пользовательского режима. Благодаря этому переключение в системный режим, часто используемое прикладной программой для запуска функций операционной системы (ввод, вывод, служба времени, запрос дополнительной памяти и т.п.), выполняется без дополнительных затрат времени на модификацию содержимого РАС и РОС. Проблема обмена данными между ОС и прикладными программами решается с помощью двух специальных команд и включения в состав процессора дополнительного указателя стека R12. Команда MFPI (код 0065SS) пересылает 16-разрядный операнд-источник, адресуемый полями DD, в стек текущего режима, адресуемый через R12. Команда MTPI (код 0066DD) пересылает слово из стека текущего режима, адресуемого через R12, по адресу операнда-приемника предыдущего режима.

В формате PSW ЭВМ, поддерживающей эти режимы, используется старший байт, 4 старших бита которых хранят коды текущего (PSW[15..14]) и предыдущего режимов (PSW[13..12]).

Регистр состояния RS0 обеспечивает включение диспетчера (RS[0] = 1) со стороны операционной системы и анализ причины прерывания:

RS0[3..1] - номер РАС, при обращении по которому возникло прерывание;

RS0[13] - прерывание возникло при попытке записи в страницу, досупную только для чтения;

RS0[14] - прерывание возникло при нарушении длины страницы;

RS0[15] - прерывание возникло при обращении к неактивной странице.

За диспетчером памяти закреплен вектор прерывания с адресом 250. В регистре состояния RS2 содержится EA, во время обработки которого возникло это прерывание.

2.7 Основные директивы ассемблера и макрокоманды ОС RT-11

Ассемблер MACRO-11, используемый для машинно-ориентированного программирования на ЭВМ PDP-11, выполняет построчное сканирование текста ассемблер-программы, анализирует этот текст и генерирует объектный код машинной программы. Каждая строка исходного модуля содержит один оператор языка ассемблера. Общий формат оператора имеет вид:

<метка>**:** <мнемокод оператора> <операнды> **;** <комментарий>

Не все поля обязательны. Разделители '**:**' и '**;**' обязательно должны присутствовать там, где нужно завершить поле метки ('**:**') или начать комментарий ('**;**'). Операнды разделяются запятой. В качестве операндов могут быть числовые, символьные и строковые константы, имена и выражения. По умолчанию числовые константы представлены в восьмеричном виде. Символьные константы начинается с кавычки (например: 'A, '3). Ассемблер транслирует такую константу в один байт со значением в соответствии с кодировкой символов ЭВМ (ASCII, КОИ-7). Строковые константы могут фигурировать только в директивах размещения строк текста, которые будут описаны ниже. Имя может содержать цифры, прописные латинские буквы и символ '¤'. Если значение имени должно быть доступно из других исходных модулей (когда испольуется компоновка одного загрузочного модуля из нескольких объектных), то такое имя должно быть объявлено как глобальное. У глобальных имен значащими считаются 6 первых символов, т.е. такие имена должны быть уникальными по первым шести символам.

RS0[3..1] - номер PAC, при обращении по которому возникло прерывание;

Метка записывается в начале оператора как имя с описанным выше синтаксисом. Кроме имени метка имеет значение. Во время трансляции ассемблер постоянно строит план размещения команд и данных в памяти. Для этого используется так называемый счетчик размещения (LC - location counter). По мере продвижения по программе этот счетчик постоянно увеличивается на длину команд и данных, представленных в операторах исходного модуля. Например, при трансляции оператора с однословной командой ветвления счетчик размещения до начала команды содержит адрес, по которому транслируется команда (местоположение в памяти на момент ее выполнения, если загрузочный модуль компонуется только на основе этого исходного модуля). Во время трансляции этого оператора счетчик размещения увеличивается на 2 и после трансляции указывает местоположение следующей команды. В выражениях ассемблера значение счетчика размещения представляется символом '**.**', причем это значение счетчика на момент начала трансляции оператора, содержащего выражение с "точкой". Например:

```
A:   MOV   #.,     R0     ; загрузка адреса данной команды в RO.
```

Оператор исходного текста может задавать либо машинную команду, либо директиву ассемблера. Список и форматы основных директив языка ассемблера MACRO-11 приводится ниже:

. = <выражение> ; установить счетчик размещения в новое значение

<имя> **=** <выражение> ; назначить значение для имени
.BYTE <список выражений> ; размещение байтов
.WORD <список выражений > ; размещение слов
.DWORD <список выражений> ; размещение слов двойной длины.
.ASCII <символ-ограничитель><текст><символ-ограничитель>
; размещение строки
; символ-ограничитель не должен входить в текст строки
.ASCIZ <символ-ограничитель><текст><символ-ограничитель>
; размещение строки с завершающим байтом 0 в конце
.BLKB <выражение> ; резервирование заданного числа байтов
.BLKW <выражение> ; резервирование заданного числа слов
.GLOBAL <список имен> ; объявление имен глобальными
.EVEN ; обеспечение четности значения счетчика размещения
.MCALL <список имен> ; перечисление макрокоманд, определения
; которых должны быть взяты из системной
; или пользовательской макробиблиотек
.END ; конец исходного модуля

Выражения в операторах ассемблера могут содержать знаки операций, константы, имена и значение счетчика размещения. Числовые константы по умолчанию восьмеричные. Символ '**.**' в конце числовой константы заставляет ассемблер интерпретировать ее как десятичную. Имется ряд префиксов, заставляющих изменять трактовку константы:

^B - следующее число трактуется как двоичное;

^D - следующее число трактуется как десятичное;

^O - следующее число трактуется как восьмеричное;

^C - следующее выражение вычисляется ассемблером и полученное значение поразрядно инвертируется.

Для задания символьных констант используются конструкции:

'<символ> - генерация слова, в котором младший байт равен коду заданного символа, а старший байт равен 0;

"<символ1><символ2> - генерация слова, в котором младший байт равен коду первого символа, а старший - коду младшего.

MACRO-11 поддерживает формирование и использование макрокоманд, причем ассемблер изначально поставляется с развитой системной макробиблиотекой, основными макрокомандами которой являются:

.EXIT ;возврат из программы в операционную систему

.TTYIN ; ввод кода символа с терминала в младший байт R0

.TTYOUT ; вывод на терминал символа из младшего байта R0

.TTYOUT <выражение> ;вывод на терминал значения символьного выражения;

.PRINT ;вывод на терминал строки, адресуемой регистром R0 и завершающейся байтом 0;

.PRINT <выражение> ;вывод строки, адрес которой задан выражением.

Ниже приводится простейшая программа, которая выполняет действия:

- ввод в двоичном виде двух числовых векторов длиной по 20 элементов в массивы VECT1 и VECT2; разрядность чисел - не более 16;
- формирование по месту VECT1 суммы векторов по правилам
  VECT1[i] += VECT[i];
- вывод полученного массива в двоичном виде, размещая на каждой строке по одному числу и нумеруя элементы дисятичными числами.

Считается, что входные данные вводятся правильно и их контроль не требуется. Текст примера набран прописными буквами в соответствии с тем, что терминалы российских аналогов PDP-11 обычно поддерживают только прописные символы.

```
            .MCALL      .EXIT, .TTYIN, .TTYOUT, .PRINT
            ; ПАМЯТЬ ДЛЯ ВЕКТОРОВ
VECT1:      .BLKW       20.
VECT2:      .BLKW       20.
            ; ТЕКСТЫ ЗАПРОСОВ
QVECT:      .ASCII /ВВОДИТЕ ВЕКТОР/<200>
QELEM:      .ASCII /ЭЛЕМЕНТ  /<200>
HRESULT:    .ASCIZ /РЕЗУЛЬТАТ/
NEWLINE:    .ASCIZ //       ; ПУСТАЯ СТРОКА ДЛЯ ПЕРЕХОДА НА НОВУЮ СТРОКУ
                            ; С ПОМОЩЬЮ  .PRINT
CR          = 15            ; КОД СИМВОЛА  “ВОЗВРАТ КАРЕТКИ”
LF          = 12            ; КОД СИМВОЛА  “ПЕРЕВОД СТРОКИ”
            .EVEN           ;  ВЫРАВНИВАЕМ ПО ЧЕТНОМУ АДРЕСУ

; СЧИТАЕМ, ЧТО В ЛЮБОЙ  ПОДПРОГРАММЕ  R0  И  R1  МОГУТ
; БЫТЬ ИСПОЛЬЗОВАНЫ БЕЗ СОХРАНЕНИЯ
            ; ОСНОВНАЯ ПРОГРАММА
START:      ; ВВОД ВЕКТОРОВ
            ; НОМЕР ВЕКТОРА И АДРЕС ПЕРЕДАЮТСЯ В GETVECT ЧЕРЕЗ СТЕК
            MOV         #1,         -(SP)
            MOV         #VECT1,     -(SP)
            JSR         PC,         GETVECT
            CMP         (SP)+,      (SP)+     ; SP += 4 ДЛЯ ОСВОБОЖДЕНИЯ
                                                ; ПАМЯТИ ПАРАМЕТРОВ ПП
            MOV         #2,         -(SP)
            MOV         #VECT2,     -(SP)
            JSR         PC,         GETVECT
            CMP         (SP)+,      (SP)+
            ; ПОЛУЧЕНИЕ СУММЫ
            MOV         #VECT1,     R0  ; БАЗОВЫЙ АДРЕС
            MOV         #VECT2,     R1
            MOV         #20.,       R2
L1:         ADD         (R1)+,      (R0)+
            SOB         R2,         L1
            ; ВЫВОД РЕЗУЛЬТАТА
            .PRINT      #HRESULT
```

```
                ; НАЧАТЬ С ПЕРВОГО ЭЛЕМЕНТА
                MOV     #1,     R4      ; ИНИЦИАЛИЗАЦИЯ СЧЕТЧИКА
                MOV     #VECT1, R3      ; ИНИЦИАЛИЗАЦИЯ УКАЗАТЕЛЯ
L2:             ; ВЫВОД  НОМЕРА  ЭЛЕМЕНТА
                MOV     R4,     R0
                JSR     PC,     PUTNUM
                ; ВЫВОД  ЗНАЧЕНИЯ  ЭЛЕМЕНТА
                MOV     (R3)+,  R0
                JSR     PC,     PUTBIN
                INC     R4              ; МОДИФИКАЦИЯ НОМЕРА ЭЛЕМЕНТА
                CMP     R4,     #20     ; ВЫВЕДЕН НЕ ПОСЛЕДНИЙ ЭЛЕМЕНТ?
                BLE     L2              ; ЕСЛИ НЕТ, ТО ПРОДОЛЖИТЬ
                .EXIT                   ; ВЫЙТИ В ОС

; ПП ВВОДА ВЕКТОРА ПО АДРЕСУ R0.  В R1 НОМЕР ВЕКТОРА
GETVECT:
                ; СОХРАНЕНИЕ РЕГИСТРОВ ДЛЯ ИСПОЛЬЗОВАНИЯ
                ; В КАЧЕСТВЕ ЛОКАЛЬНЫХ ПЕРЕМЕННЫХ
                MOV     R2,     -(SP)
                MOV     R3,     -(SP)
                ; ТЕПЕРЬ:  6(SP) - АДРЕС ВЕКТОРА, 8.(SP) - НОМЕР ВЕКТОРА
                ; ЗАПРОС ВЕКТОРА
                .PRINT  #QVECT
                ; ФОРМИРОВАНИЕ И ВЫВОД СИМВОЛА - НОМЕРА ВЕКТОРА
                MOVB    #'0,    R0
                ADD     8.(SP), R0 ; СДЕЛАТЬ СИМВОЛ ИЗ ЧИСЛА
                .TTYOUT
                .PRINT  #NEWLINE
                MOV     #1,     R2 ; ИНИЦИАЛИЗАЦИЯ СЧЕТЧИКА ЭЛЕМЕНТОВ
                MOV     6(SP),  R3 ; ИНИЦИАЛИЗАЦИЯ УКАЗАТЕЛЯ ВЕКТОРА
LGVEC:          .PRINT  #QELEM
                MOV     R2,     R0
                JSR     PC,     PUTNUM
                JSR     PC,     GETBIN
                MOV     R0,     (R3)+
                CMP     R2,     #20.
                BLE     LGVEC
                MOV     (SP)+,  R3
                MOV     (SP)+,  R2
                RTS     PC

; ПП ВВОДА ДВОИЧНОГО ЧИСЛА В R0
GETBIN:         CLR     R1  ; ОЧИСТИТЬ МЕСТО ФОРМИРОВАНИЯ  ЧИСЛА
                ; ЦИКЛ ВВОДА ЦИФР ЧИСЛА (БЕЗ КОНТРОЛЯ)
L3:             .TTYIN
                CMPB    R0,     #CR     ; КОНЕЦ СТРОКИ?
                BEQ     L4              ; ДА, ИДТИ НА ЗАВЕРШЕНИЕ
                ROR     R0              ; ВЫДВИНУТЬ 0/1 ИЗ БАЙТА СИМВОЛА
                ROL     R1              ; ДОПИСАТЬ БИТ СЛЕВА К ЧИСЛУ В  R1
                BR      L3              ; ПРОДОЛЖИТЬ
```

```
L4:     .TTYIN                          ; ДОЧИТАТЬ LF В КОНЦЕ СТРОКИ
ВВОДА
        MOV     R1,     R0      ;  ПОЛУЧИТЬ РЕЗУЛЬТАТ В R0
        RTS     PC              ;  ВЫЙТИ ИЗ ПОДПРОГРАММЫ

; ПП ВЫВОДА ЦИФРЫ НА ТЕРМИНАЛ
;ЧИСЛОВОЕ ЗНАЧЕНИЕ ЦИФРЫ НАХОДИТСЯ В R0
PUTDIG: ADD     #'O,    R0
        .TTYOUT
        RTS     PC

; ПОДПРОГРАММА ВЫДАЧИ НОМЕРА ЭЛЕМЕНТА
; НОМЕР ПЕРЕДАЕТСЯ ЧЕРЕЗ R0
PUTNUM: MOV     R3,     -(SP)   ; СОХРАНИТЬ R3 И R4
        MOV     R4,     -(SP)
        .PRINT  #QELEM          ; ВЫВОД СЛОВА "ЭЛЕМЕНТ "
        ; ПОЛУЧИТЬ ЦИФРУ ДЕСЯТКОВ В R3 И ЦИФРУ ЕДИНИЦ В R4
        MOV     R0,     R3
        CLR     R4
        DIV     #10.,   R3
        ; ТЕПЕРЬ В R0 ЧИСЛО ДЕСЯТКОВ В НОМЕРЕ,
        ; А В R1 ЧИСЛО ЕДИНИЦ
        ; ВЫВЕСТИ ДВЕ ЦИФРЫ
        JSR     PC,     PUTDIG
        MOV     R1,     R0
        JSR     PC,     PUTDIG
        ; ВОССТАНОВЛЕНИЕ
        MOV     (SP)+,  R4
        MOV     (SP)+,  R3
        RTS     PC

; ПОДПРОГРАММА ВЫДАЧИ ЧИСЛА В ДВОИЧНОМ  ВИДЕ
; ЧИСЛО ПЕРЕДАЕТСЯ ЧЕРЕЗ R0
PUTBIN: MOV     R3,     -(SP)   ; СОХРАНИТЬ R3 И R4
        MOV     R4,     -(SP)
        MOV     R0,     R3
        MOV     #16,    R4
L5:     CLR     R0
        ASL     R3
        ROL     RO
        ; ТЕПЕРЬ В R0 СЛЕДУЮЩАЯ ДВОИЧНАЯ ЦИФРА В ЧИСЛЕ
        JSR     PC,     PUTDIG
        SOB     R4,     L5
        ; ВОССТАНОВЛЕНИЕ
        MOV     (SP)+,  R4
        MOV     (SP)+,  R3
        .PRINT  NEWLINE
        RTS     PC
        .END    START ;  КОНЕЦ ТЕКСТА ПРОГРАММЫ
```

Список литературы

1. Уокерли Дж. Архитектура и программирование микро-ЭВМ: В 2-х книгах. - М.: Мир, 1984. - Кн. 2.

2. Кичев Г.Г., Некрасов Л.П. Архитектура и аппаратные средства мини-ЭВМ СМ-1600: Учеб. пособие для специалистов по эксплуатации аппаратных средств в области ВТ и АСУ. - М.: Машиностроение, 1988.

3. Финогенов К.Г. Программирование измерительных систем реального времени. - М.: Энергоатомиздат, 1990.

4. Управляющие и вычислительные устройства роботизированных комплексов на базе микроЭВМ: Учеб.пособие для техн.вузов/В.С.Медведев, Г.А.Орлов, Ю.И.Рассадин и др.; Под ред. В.С.Медведева. - М.: Высш. школа, 1990.