ЛАБОРАТОРНАЯ РАБОТА № 2

**Тема: *«РЕЖИМЫ АДРЕСАЦИИ ОПЕРАНДОВ МП ФОН-НЕЙМАНОВСКОЙ АРХИТЕКТУРЫ»***

**Цель:**

* повторить структуру команды – основные поля и их назначение;
* определить формат команды – структуру ее полей

Команда состоит из операционной и адресной частей. Операционная часть команды содержит код операции (сложение, вычитание, передача и т.д.). Адресная часть содержит информацию об адресах операндов и результата, а иногда – и об адресе следующей команды (если используется принудительный порядок выборки команд, в этом случае команда должна быть четырёхадресной). Обычно применяется естественный порядок, когда после выполнения команды, расположенной в памяти по адресу К, выполняется команда по адресу К+1. Тогда отпадает необходимость указывать адрес следующей команды (для изменения естественного порядка используются специальные команды) и команда становится трехадресной. Если условиться, что результат всегда направляется по адресу одного из операндов, участвовавших в операции, то получим двухадресную команду. Если в качестве одного из операндов всегда используется содержимое определённого регистра (регистра результата, или аккумулятора) и туда же направляется результат, то можно не указывать этот адрес и, следовательно, работать с одним адресом в коде команды. Если использовать стек, то можно вообще построить безадресную команду.

Сокращение адресности удобно, т.к. позволяет уменьшить длину разрядной сетки в коде команды.

Следует различать **адресный код** **АК** , это информация об адресе операнда, записанная в адресной части команды, и **исполняемый адрес АИ** – это сам адрес, номер ячейки памяти, к которой производится фактическое обращение.

Выбор способов адресации, формирования исполнительного адреса и преобразования адресов является одним из важнейших вопросов разработки системы команд ЭВМ.

В ЭВМ используются следующие типы адресации:

**Прямая адресация**: **АК АИ**.

**Относительная адресация**: **АИ** = АБ + АК, где **АБ** – базовый адрес, хранящийся в определённом месте памяти. Такая система позволяет сократить длину кода **АК**, поскольку теперь можно задавать лишь некоторое смещение относительно длинного базового адреса. Эта система адресации, кроме того, обеспечивает простую реентерабельность (перемещаемость) программы в памяти, а также облегчает распределение памяти при составлении программ несколькими программистами.

**Косвенная адресация**: АК указывает адрес ячейки памяти, в которой хранится адрес операнда или команды («адресация адреса»). Она удобна в ЭВМ с коротким форматом команды.

**Непосредственная адресация**: вместо АК помещается сам операнд. Это сокращает время выполнения команды, однако разрядность операнда ограничена (меньше разрядности кода команды).

**Стековая адресация**. Стек реализует правило LIFO.

Любая операция выполняется в ЭВМ за несколько элементарных шагов, называемых ***микрооперациями*.** Каждая микрооперация выполняется за один шаг (такт) и приводится в действие одним управляющим сигналом.

В некоторые такты может поступать несколько управляющих сигналов, вызывая параллельное во времени выполнение соответствующих микроопераций. Такая совокупность микроопераций, выполняемая за один такт, называется ***микрокомандой***. Последовательность микрокоманд, обеспечивающих выполнение данной операции, называется ***микропрограммой*** данной операции.

Существует два различных принципа поиска операндов в памяти: ассоциативный и адресный.

**Ассоциативный поиск** операнда (*поиск по содержанию ячейки*) предполагает просмотр содержимого всех ячеек памяти для выявления кодов, содержащих заданный командой ассоциативный признак. Эти коды и выбираются из памяти в качестве искомых операндов.

**Адресный поиск** предполагает, что искомый операнд извлекается из ячейки, номер которой формируется на основе информации в адресном поле команды. Ниже мы будем рассматривать только реализацию адресного принципа поиска операнда.

Следует различать понятия исполнительного адреса и адресного кода.

**Адресный код** – это информация об адресе операнда, содержащегося в команде.

**Исполнительный адрес** – это номер ячейки памяти, к которой фактически производится обращение.

В современных ЭВМ адресный код, как правило, не совпадает с исполнительным адресом. Таким образом, ***СПОСОБ АДРЕСАЦИИИ* можно определить как способ формирования исполнительного адреса операнда Аи по адресному коду команды Ак.**

В системах команд современных ЭВМ часто предусматривается возможность использования нескольких способов адресации операндов для одной и той же операции. Для указания способа адресации в некоторых системах команд выделяется специальное поле в команде - «метод» (указатель адресации). В этом случае любая операция может выполняться с любым способом адресации, что значительно упрощает программирование.

Адресуемые в командах операнды хранятся в основной памяти (ОП) и регистровой памяти (РП), рисунок 2.



Рис. 1. Виды памяти для хранения адресуемых операндов

Рассмотрим способы адресации, применяемые в современных ЭВМ.

### Классификация способов адресации по наличию адресной информации в команде

По наличию адресной информации в команде различают явную и неявную адресацию.

При **явной адресации** операнда в команде есть поле адреса этого операнда, в котором задается адресный код **Ак**. Большинство методов адресации являются явными.

При **неявной адресации** адресное поле в команде отсутствует, адрес операнда подразумевается и фактически задается кодом операции. Например, в команде ИНКРЕМЕНТА второй операнд – приращение на 1 не адресуется и является подразумеваемым.

Метод неявной адресации операндов используется во всех процессорах. ***Основное его назначение - уменьшение длины команды за счет исключения части адресов.*** При этом методе код операции точно задает адрес операнда. Например, из команды исключается адрес приемника результата. При этом подразумевается, что результат в этой команде помещается на место второго операнда. В качестве примера – команды с использованием регистра аккумулятора.

### Классификация по способу формирования исполнительных адресов ячеек памяти

Способы формирования адресов ячеек памяти (Аи) можно разделить на абсолютные и относительные.

**Абсолютные способы формирования** предполагают, что двоичный код адреса ячейки памяти - Аи может быть извлечен целиком либо из адресного поля команды (в случае прямой адресации), или из какой-либо другой ячейки (в случае косвенной адресации), никаких преобразований кода адреса не производится.

**Относительные способы формирования** Аи предполагают, что двоичный код адреса ячейки памяти определяется суммой адресного кода и некоторого числа, называемого базовым адресом. Адресный код в этом случае играет роль смещения фактического исполнительного адреса относительно базового адреса. АК=АИ+АБ

### Классификация способов адресации по кратности обращения в память

Широко используются следующие методы адресации операнда с различной кратностью обращения (R) в память:

1. Непосредственная (R = 0).

2. Прямая (R = 1).

3. Косвенная (R > 2).

**Непосредственная адресация операнда**. При этом способе ***операнд располагается в адресном поле команды***. Обращение к регистровой памяти (РП) или оперативной памяти (ОП) не производится. Таким образом, уменьшается время выполнения операции, сокращается объем памяти. Непосредственная адресация удобна для задания констант, длина которых меньше или равна длине адресного поля команды. Когда операндом является число, оно обычно представляется в дополнительном коде. Этот способ адресации может применяться при выполнении арифметических операций, операций сравнения, а также для загрузки констант в регистры.

**Прямая адресация операндов** (абсолютный способ формирования). При этом способе (рис. 3.3) адресации обращение за операндом в РП или ОП производится по адресному коду в поле команды, т.е. ***исполнительный адрес операнда совпадает с адресным кодом команды (Аи = Ак).*** Фактическиадресный код команды указывает номер ячейки памяти, к которой производится обращение.

*Рис.2 Схема с прямой адресацией*

Обеспечивая простоту программирования, этот метод имеет существенные недостатки, так как для адресации к ячейкам памяти большой емкости (число адресов М велико) требуется «длинное» адресное поле в команде. Прямая адресация используется широко в сочетании с другими способами адресации.

**Регистровая адресация.** При использовании этого метода адресации в команде указывается регистр общего назначения и содержимое этого регистра интерпретируется процессором как операнд.

**Косвенная адресация операндов.** При этом способе ***адресный код команды указывает адрес ячейки памяти, в которой находится не сам операнд, а лишь адрес операнда, называемый указателем операнда.*** Другими словами, косвенная адресация может быть определена как «адресация адреса». В некоторых ЭВМ используется многоступенчатая косвенная адресация. Возможно, адресный код команды указывает на регистр микропроцессора, хранящий адрес операнда в оперативной памяти. Итак, адресация к операнду через цепочку указателей (косвенных адресов) называется косвенной.

Адрес указателя, задаваемый программой, остается неизменным, а ***косвенный адрес может изменяться в процессе выполнения программы***. Косвенная адресация, таким образом, обеспечивает переадресацию данных, т.е. упрощает обработку массивов и списковых структур данных, упрощает передачу параметров подпрограммам, но не обеспечивает перемещаемость программ в памяти (рис. 3.4).

**Регистровая косвенная адресация. *При адресации операнда используется содержимое указанного в команде какого-либо регистра процессора.*** Т.е. содержимое регистра интерпретируется как адрес операнда.

**Автоинкрементный метод адресации**

Содержимое указанного в команде регистра интерпретируется процессором как адрес ячейки памяти, в которой находится операнд (как в косвенно-регистровом методе), но после выборки операнда содержимое регистра (адрес операнда) увеличивается (инкрементируется) на 1 или 2, таким образом, чтобы указывать на адрес следующей по порядку ячейки памяти. (Пример – работа регистра счетчика команд).

**Косвенный-автоинкрементный метод адресации**

Содержимое указанного в команде регистра интерпретируется процессором как адрес ячейки памяти, в которой находится адрес операнда, и после выборки операнда содержимое регистра (адрес адреса) увеличивается (инкрементируется), таким образом, чтобы указывать на адрес следующей по порядку ячейки памяти.

**Автодекрементный метод адресации**

Содержимое указанного в команде регистра вначале уменьшается (декрементируется) на 1 или 2, после чего уменьшенное содержимое интерпретируется процессором как адрес ячейки памяти, в которой находится операнд.

**Косвенный-автодекрементный метод адресации**

Содержимое указанного в команде регистра вначале уменьшается, после чего интерпретируется процессором как адрес ячейки памяти, в которой находится адрес операнда.



Адрес указателя

*Рис. 3. Косвенная адресация*

а) косвенная адресация

б) регистровая косвенная адресация операндов – адресный код команды указывает на регистр микропроцессора.

#### Косвенная адресация со смещением

**Базирование способом суммирования.** В команде адресный код Ак разделяется на две составляющие: Аб - адрес регистра в регистровой памяти , в котором хранится база Б (базовый адрес); С - код смещения относительно базового адреса (рис. 3.5).

С помощью метода относительной адресации удается получить так называемый ***перемещаемый программный модуль***, который одинаково выполняется процессором независимо от адресов, в которых он расположен. Начальный адрес программного модуля (база) загружается, при входе в модуль, в базовый регистр. Все остальные адреса программного модуля формируются через смещение относительно начального адреса (базы) модуля. Таким образом, одна и та же программа может работать с данными, расположенными в любой области памяти, без перемещения данных и без изменения текста программы только за счет изменения содержания всего одного базового регистра. Однако время выполнения каждой операции при этом возрастает. (ХОРОШО РАЗОБРАТЬСЯ! ЭТО ОЧЕНЬ ВАЖНО!!!)

**Базирование способом совмещения составляющих.** Для увеличения емкости адресной ОП без увеличения длины адресного поля команды можно использовать для формирования исполнительного адреса совмещение (конкатенацию) кодов базы и смещения (рис. 3.6).

СМ – сумматор,

РАОП – регистр адреса ОП,

Б – база (базовый адрес),

С – смещение,

Аб – адрес регистра базы

Однако в данном случае начальные адреса массивов не могут быть реализованы произвольно, а должны иметь в младших разрядах n нулей, где n – длина поля смещения.

****

*Рис. 4 Схема формирования относительного адреса способом суммирования кодов базы и смещения.*

**Индексная адресация (косвенная адресация со смещением).** Для работы программ с массивами, требующими однотипных операций над элементами массива, удобно использовать индексную адресацию. Схема индексной адресации аналогична базированию путем суммирования (см. рис. 3.5).

В этом случае адрес i-гo операнда в массиве определяется как сумма начального адреса массива (задаваемого полем смещения С) и индекса И, записанного в одном из регистров РП, называемом теперь индексным регистром. Адрес индексного регистра задается в команде полем адреса индекса — Аин (аналогично Аб ).

В каждом i-м цикле содержимое индексного регистра изменяется на величину постоянную (часто равную 1). Использование индексной адресации значительно упрощает программирование циклических алгоритмов.

Для эффективной работы при относительной адресации применяется комбинированная индексация с базированием, при которой адрес операнда вычисляется как сумма трех величин (рис. 3.7): Аи = Б + И + С.

Принятая система адресации в команде предусматривала вычисление исполнительного адреса в виде суммы смещения, указанного непосредственно в коде команды и содержимого двух регистров общего назначения - базового и индексного

*Схема формирования исполнительного адреса*

**[Адрес в ОП] = [Смещение] +[Регистр базы] +[Регистр индекса]**

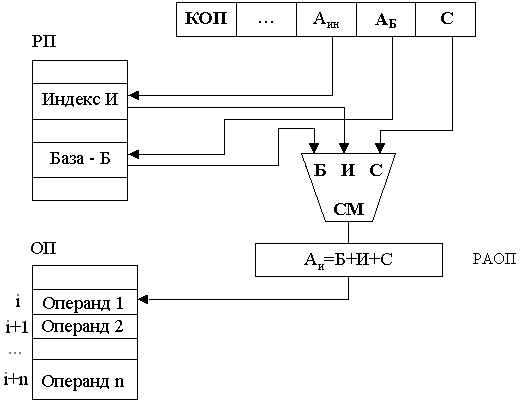
#### Относительный метод адресации (относительное формирование адреса). Во втором слове команды указывается относительный адрес операнда, т.е. величина смещения адреса операнда относительно адреса самой команды (текущего содержимого регистра-счетчика команд процессора).

#### Стековая адресация

Стековая память (стек) является эффективным элементом современных ЭВМ, реализует неявное задание адреса операнда. Хотя адрес обращения в стек отсутствует в команде, он формируется схемой управления автоматически по специальному правилу. (Рассмотренные автоинкрементная и автодекрементная адресации).



*Рис. 5 Схема формирования относительного адреса способом совмещения кодов базы и смещения.*



*Рис. 6 Схема формирования дополнительного адреса при индексной адресации и базировании: АИН - адрес индексного регистра.*

**КОНТРОЛЬНЫЕ ВОПРОСЫ.**

1. Типы команд ЭВМ.
2. Какая информация кодируется в команде ЭВМ?
3. Структура команды?
4. Формат команды?
5. Какие проблемы приходится решать при кодировании команд ЭВМ?
6. Назовите способы уменьшения длины команды.
7. Что такое адресация операндов, методы адресации?
8. Что такое адресный код, исполнительный адрес, чем отличаются эти понятия?
9. Для чего в ЭВМ необходимо наличие разнообразных методов адресации операндов?
10. Что такое подразумеваемый операнд, подразумеваемый адрес?
11. Непосредственная адресация.
12. Абсолютная адресация.
13. Относительная адресация.
14. Регистровая адресация.
15. Косвенная адресация.
16. Регистровый и косвенно-регистровый методы адресацию
17. Косвенная адресация со смещением.
18. Автоинкрементная адресация.
19. Косвенная автоинкрементная адресация.
20. Автодекрементная адресация.
21. Косвенная автодекрементная адресация.
22. Использование регистра-счетчика команд при реализации непосредственного, абсолютного и относительного методов адресации.
23. Какой метод адресации следует использовать в перемещаемой программе для адресации данных, расположенных в теле программы?
24. Назовите типы данных, используемых в ЭВМ.
25. Двоично-десятичные данные. Форматы их кодировки.

***Примеры трехадресных команд:***

***ADD R1,7,R1*** – содержимое регистра R1 (регистровая адресация) сложить с непосредственным операндом 7 (непосредственная адресация), результат занести в R1.

***ADD (R2),(R3),R1***- извлечь операнд 1 по адресу, хранящемуся в регистре R2 (регистровая косвенная адресация), извлечь операнд 2 по адресу, хранящемуся в регистре R3 (регистровая косвенная адресация), сложить их и результат занести в регистр R1.

***ADD (R1)+,R2,A*** – операнд 1 находится по адресу, хранящемуся в регистре R1, после извлечения операнда, содержимое регистра увеличится на 1 (операнды размером в 1 байт) – автоинкрементная адресация; операнд 2 находится в регистре R2 (регистровая адресация), сумма операндов заносится в регистр аккумулятор (регистровая адресация).

***ADD R5,−(R4),A*** – операнд 1 находится в регистре R5 (регистровая адресация), операнд 2 извлекается по адресу, хранящемуся в регистре R4, перед обращением к которому его содержимое уменьшается на 1 (операнды размером в 1 байт) – автодекрементная адресация; сумма операндов заносится в регистр аккумулятор (регистровая адресация).

***Примеры двухадресных команд:***

***ADD R1,R2*** – содержимоерегистра R1 сложить с содержимым регистра R2, результат записать в регистр R1 (регистровая адресация)

***ADD (R5),7 –*** содержимое ячейки памяти по адресу из регистра R5 (косвенная регистровая адресация) сложить с числом 7 (непосредственная адресация), результат занести на место первого операнда.

***ADD (RБ+25),A –*** содержимое ячейки памяти по адресу, определяемому как сумма содержимого регистра базы RБ и смещения 25 (косвенная базовая адресация), сложить с содержимым регистра аккумулятора (регистровая адресация), результат записать на место первого операнда***.***

***ADD R3,( RБ+RИ+8) –*** содержимое регистра R3 (регистровая адресация) сложить с содержимым ячейки памяти по адресу, определяемому как сумма содержимого регистра базы, регистра индекса и смещения RБ+RИ+8 (косвенная базово-индексная адресация со смещением), результат записать в регистр R3.

***ADD R1,PER –*** сложить содержимое регистра R1 (регистровая адресация) с содержимым ячейки памяти, поименованной в программе как PER (прямая абсолютная адресация), результат занести на место первого операнда.

***Примеры одноадресных команд***

***MUL R4 –*** содержимое регистраR4 умножить на содержимое регистра аккумулятора, результат записать в регистр аккумулятор (регистровая адресация).

***DIV (R1)*** – содержимое регистра аккумулятора разделить на содержимое ячейки памяти по адресу из регистра R1 (косвенная регистровая адресация), целую часть от деления записать в регистр аккумулятор.

***JA L1*** – осуществить условный переход «если больше» на команду, помеченную меткой L1. Команда хранит величину смещения, которую следует добавить к адресу, хранящемуся в регистре счетчике команд, для того, чтобы нарушить линейную структуру команды и вместо перехода к следующей за JA команде, перейти на команду с меткой L1 (прямая относительная адресация).

**Задание для выполнения:**

Привести аналогичные примеры (с полным описанием команд), для **всех** режимов адресации, используя команды SUB (вычитание), MOV (пересылка), OR и JMP.