# Лекция 6.
# Способы адресации в микропроцессорных системах

# Режимы адресации

Для взаимодействия с различными модулями в ЭВМ должны быть средства идентификации ячеек внешней памяти, ячеек внутренней памяти, регистров МП и регистров устройств ввода/вывода. Поэтому каждой из запоминающих ячеек присваивается адрес, т.е. однозначная комбинация бит. Количество бит определяет число идентифицируемых ячеек. Обычно ЭВМ имеет различные адресные пространства памяти и регистров МП, а иногда - отдельные адресные пространства регистров устройств ввода/вывода и внутренней памяти. Кроме того, память хранит как данные, так и команды. Поэтому для ЭВМ разработано множество способов обращения к памяти, называемых режимами адресации.

Режим адресации памяти - это процедура или схема преобразования адресной информации об операнде в его исполнительный адрес.

# Режимы адресации

Все способы адресации памяти можно разделить на:

1)**прямой**, когда исполнительный адрес берется непосредственно из команды или вычисляется с использованием значения, указанного в команде, и содержимого какого-либо регистра (прямая адресация, регистровая, базовая, индексная и т.д.);

2) **косвенный**, который предполагает, что в команде содержится значение косвенного адреса, т.е. адреса ячейки памяти, в которой находится окончательный исполнительный адрес (косвенная адресация).

В каждой микроЭВМ реализованы только некоторые режимы адресации, использование которых, как правило, определяется архитектурой МП.

# Способы адресации

Двоичный *n*-разрядный номер ячейки памяти, к которой нужно обратиться в ходе выполнения вычислительного процесса, принадлежащий этой и только этой ячейке – называется полным ***физическим*** или ***исполнительным адресом***.

Процедура вычисления исполнительного адреса определяется способами адресации и расположением самих операндов, которые могут находиться:

- в самой команде;
- в регистрах микропроцессора;
- в памяти, т.е. в ОЗУ;
- в устройствах ввода-вывода.

# Способы адресации

Следует различать понятия ***исполнительный адрес*** и ***адресный код*** в формате команды.

Исполнительный адрес — это номер ячейки памяти, к которой производится фактическое обращение.

Адресный код — это информация об адресе операнда, содержащаяся в команде. В современных микросистемах, адресный код часто не совпадает с исполнительным адресом. Выбор способов адресации, формирования исполнительного адреса и преобразования адресов является одним из важнейших вопросов разработки ЭВМ.

# Способы адресации

Команды разных микропроцессорных систем могут использовать различные способы адресации, которые применяются как сами по себе, так и совместно друг с другом. Рассмотрим некоторые способы адресации, широко используемые в современных ЭВМ:

***Неявная адресация***. В команде не содержится явных указаний об адресе участвующего в операции операнда или адреса, по которому помещается результат операции, но этот адрес, так или иначе, подразумевается. И в конечном итоге жестко привязан к выполняемой операции.

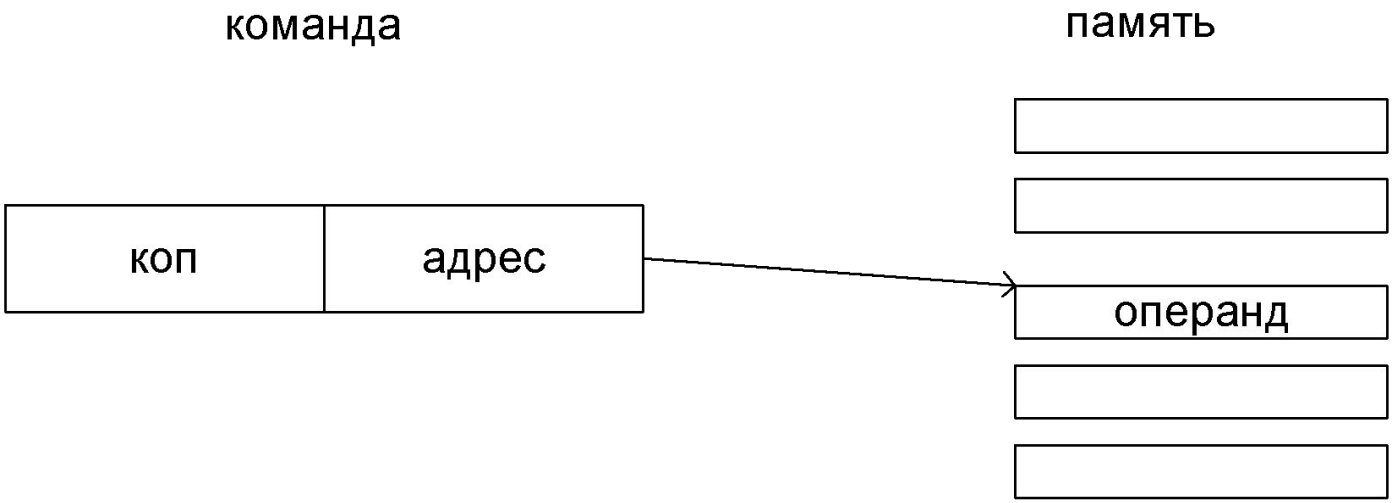
```
cbw

mul al
```

# Способы адресации

**Прямая адресация**. Исполнительный адрес совпадает с адресной частью команды. Этот способ адресации был общепринятым в первых вычислительных машинах и продолжает применяться в настоящее время в комбинации с другими способами.

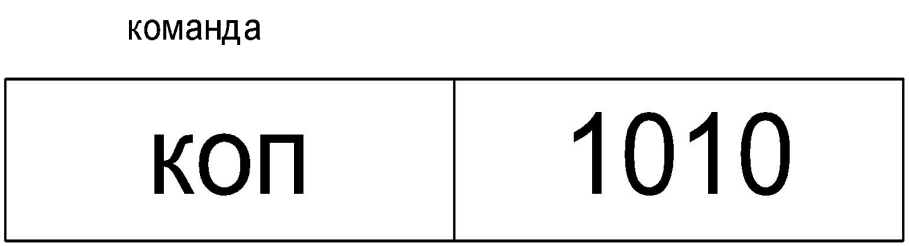
Недостатком является, то что при работе с большим объемом памяти требуется длинное адресное поле и при работе программы адрес не может быть изменен.



# Способы адресации

***Непосредственная адресация***. В команде содержится не адрес операнда, а непосредственно сам операнд. Такая адресация удобна для хранения различного рода констант. (Целочисленное значение операнда записывается в поле команды в виде дополнительного кода.) Используется для арифметических операций, сравнения и записи в регистры.

```
mov al,5
add bx,1234h
mov dx,a
```




Операнд : константа 10 в двоичном коде

# Способы адресации

- ***Относительная адресация или базирование***. *Исполнительный адрес* определяется суммой адресного кода команды $A_К$ и некоторого числа $A_Б$ называемого ***базовым адресом***:
  - $A_И = A_Б + A_К$

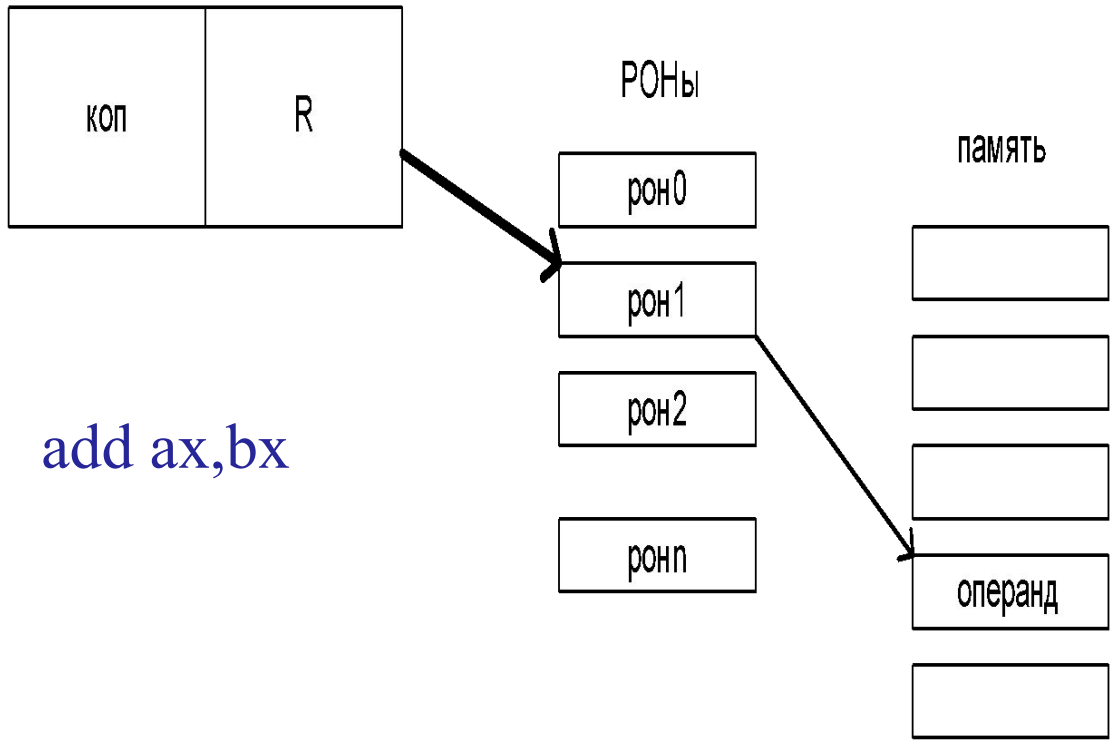
Относительная адресация позволяет при меньшей длине адресного кода команды обеспечить доступ к любой ячейке памяти. Для этого число разрядов в базовом адресе выбирают таким, чтобы можно было адресовать любую ячейку ОЗУ, а *адресный код* ***АК*** самой команды используют для представления лишь сравнительно короткого *«**смещения**»*.

```
metka:
...
loop metka
```

# Способы адресации

***Регистровая адресация***. Применяется, когда промежуточные результаты хранятся в одном из рабочих регистров центрального процессора (регистрах общего назначения (РОН)). Поскольку регистров значительно меньше чем ячеек памяти, то небольшого адресного поля может хватить для адресации.

Применение регистровой адресации наряду с сокращением длины адресов операндов, позволяет увеличить скорость выполнения операций, так как уменьшается число обращений к ОЗУ через системную магистраль.



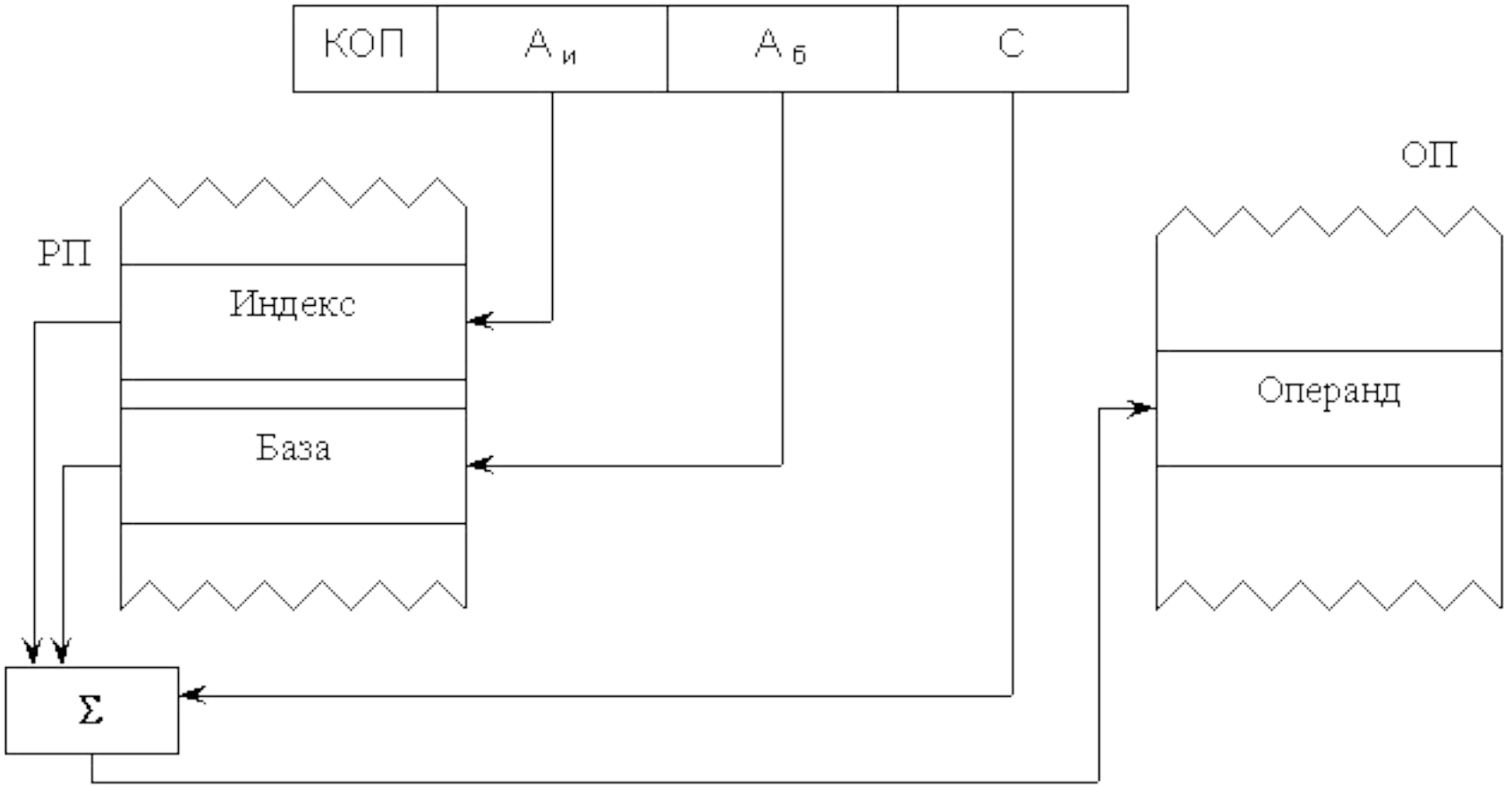
# Способы адресации

- **Индексная адресация** содержит адрес ячейки памяти, и индексный регистр (SI, DI, указанный явно или неявно) , содержащий смещение относительно этого адреса.

```
mov AL, [DI]
```

При адресации через регистры BX, SI или DI в качестве сегментного регистра подразумевается DS, при адресации через BP - регистр SS.

# Индексная адресация



# Сегментные регистры

Всего шесть сегментных регистров: cs, ss, ds, es, gs, fs. Они предназначены для указания сегмента программы к которому она имеет доступ в конкретный момент. В этих регистрах содержатся адреса памяти, с которых начинаются соответствующие сегменты. Микропроцессор поддерживает следующие типы сегментов.

- 1. Сегмент кода. Содержит команды программы. Для доступа к этому сегменту служит регистр cs (code segment register) – сегментный регистр кода. Он содержит адрес сегмента с машинными командами, к которому имеет доступ микропроцессор (т. е. эти команды загружаются в конвейер микропроцессора).

- 2. Сегмент данных. Содержит обрабатываемые программой данные. Для доступа к этому сегменту служит регистр ds (data segment register) – сегментный регистр данных, который хранит адрес сегмента данных текущей программы.

- 3. Сегмент стека. Этот сегмент представляет собой область памяти, называемую стеком. Работу со стеком микропроцессор организует по следующему принципу: последний записанный в эту область элемент выбирается первым. Для доступа к этому сегменту служит регистр ss (stack segment register) – сегментный регистр стека, содержащий адрес сегмента стека.

- 4. Дополнительный сегмент данных ds. Если программе недостаточно одного сегмента данных, то она имеет возможность использовать еще три дополнительных сегмента данных. Но в отличие от основного сегмента данных, адрес которого содержится в сегментном регистре ds, при использовании дополнительных сегментов данных их адреса требуется указывать явно с помощью специальных префиксов переопределения сегментов в команде. Адреса дополнительных сегментов данных должны содержаться в регистрах es, gs, fs (extension data segment registers).

## Способы адресации

**Базово-индексная адресация памяти**

Относительный адрес операнда определяется суммой содержимого базового и индексного регистров. Допускается использование следующих пар:

[BX][SI], [BX][DI], [BP][SI], [BP][DI].

```
mov  BX,[BP][SI] ;
```

В BX засылается слово из стека (сегментный адрес в SS), а смещение вычисляется как сумма содержимого BP и SI

.

# Способы адресации

- ***Косвенная адресация***. Адресный код команды указывает адрес ячейки памяти, в которой находится адрес операнда или команды. С помощью ограниченного адресного поля команды указывается адрес ячейки, в свою очередь, содержащей полноразрядный адрес операнда.

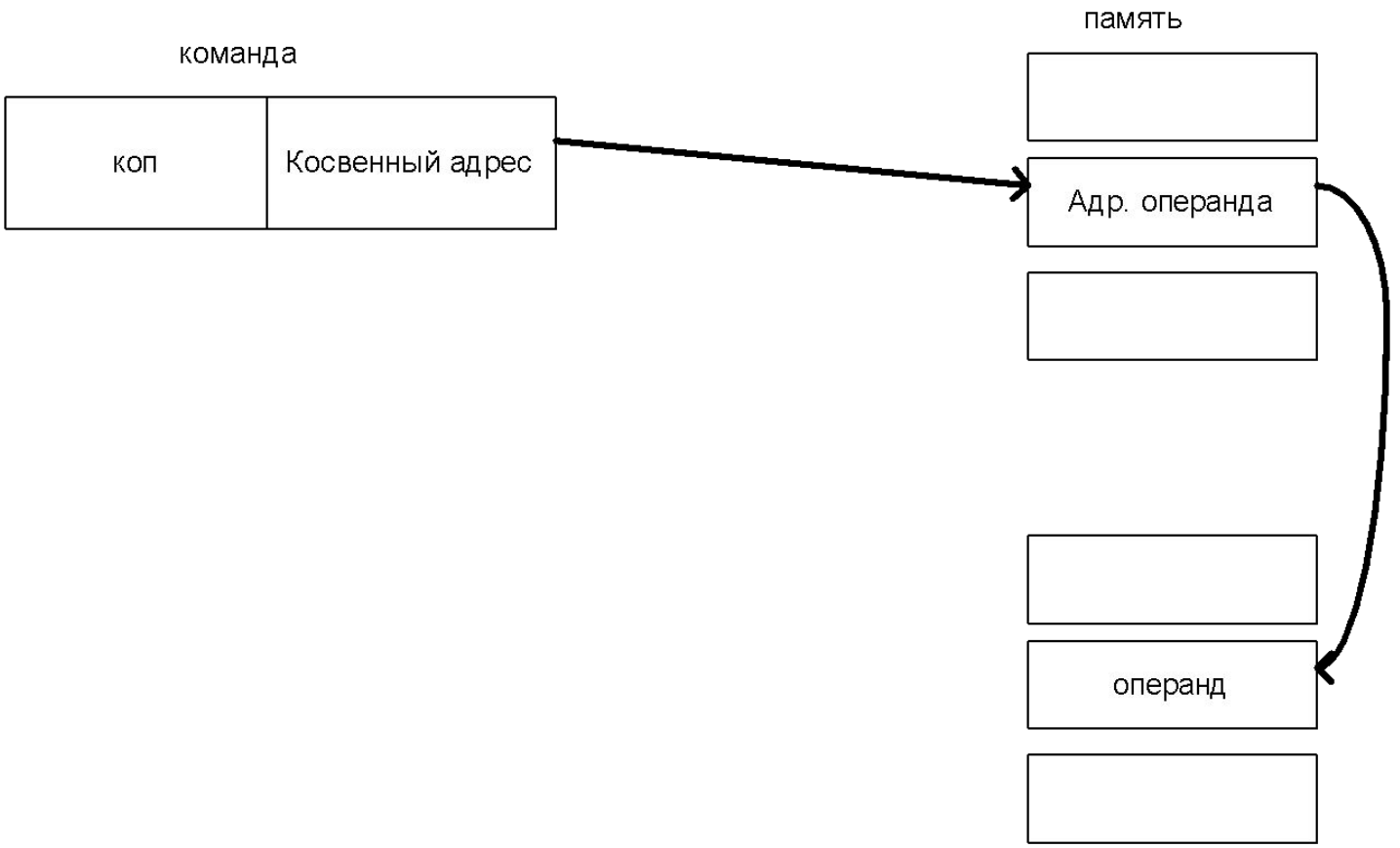
**Достоинства** содержимое адресного поля команды остается неизменно в то время как косвенный адрес в процессе выполнения программы можно изменять.

**Недостатки**:

- двукратное обращение к  памяти;
- задействуется лишняя ячейка памяти для хранения исполнительного адреса операнда.

# Способы адресации

## Косвенная адресация



# Способы адресации

- **Косвенная регистровая адресация** - исполнительный адрес операнда хранится не в ячейке основной памяти, а в регистре процессора. Соответственно, адресное поле команды указывает не на ячейку памяти, а на регистр.

Адрес операнда должен находится в одном из регистров BX, BP, SI или DI:

```
add ax,[bx]
mov dl,[si]
```

- **Косвенная регистровая (базовая) адресация со смещением**

Адрес операнда вычисляется как сумма содержимого регистра BX, BP, SI или DI и 8- или 16-разрядного смещения

```
add ax,[bx+2]
mov dx,[array1+si]
```

# Способы адресации

**Косвенная базовая индексная адресация**

Адрес операнда вычисляется как сумма содержимого одного из базовых регистров BX или BP и одного из индексных регистров SI или DI.

```
add ax,[bx+di]
```

Например, в одном из регистров может находиться адрес начала массива в памяти, а в другом — смещение какого-то элемента относительно начала.

**Косвенная базовая индексная адресация со смещением**

Адрес операнда вычисляется как сумма содержимого одного из базовых регистров BX или BP, одного из индексных регистров SI или DI и 8- или 16-разрядного смещения.

```
mov al,[bp+di+5]
mov bl,[array2+bx+si]
```

# Способы адресации

***Автоинкрементная и автодекрементная адресации***. Обеспечивает эффективную работу с массивами данных, за счет формирования адреса следующего элемента массива путем автоматического приращения или уменьшения адреса текущего обрабатываемого элемента массива, при его косвенной адресации.

# Способы адресации

**Страничная адресация -** предполагает разбиение адресного пространства на страницы. Страница определяется своим начальным адресом, выступающим в качестве базы. Старшая часть этого адреса хранится в специальном регистре - регистре адреса страницы (РАС). В адресном коде команды указывается смещение внутри страницы, рассматриваемое как младшая часть исполнительного адреса.

Исполнительный адрес образуется конкатенацией смещения к РАС

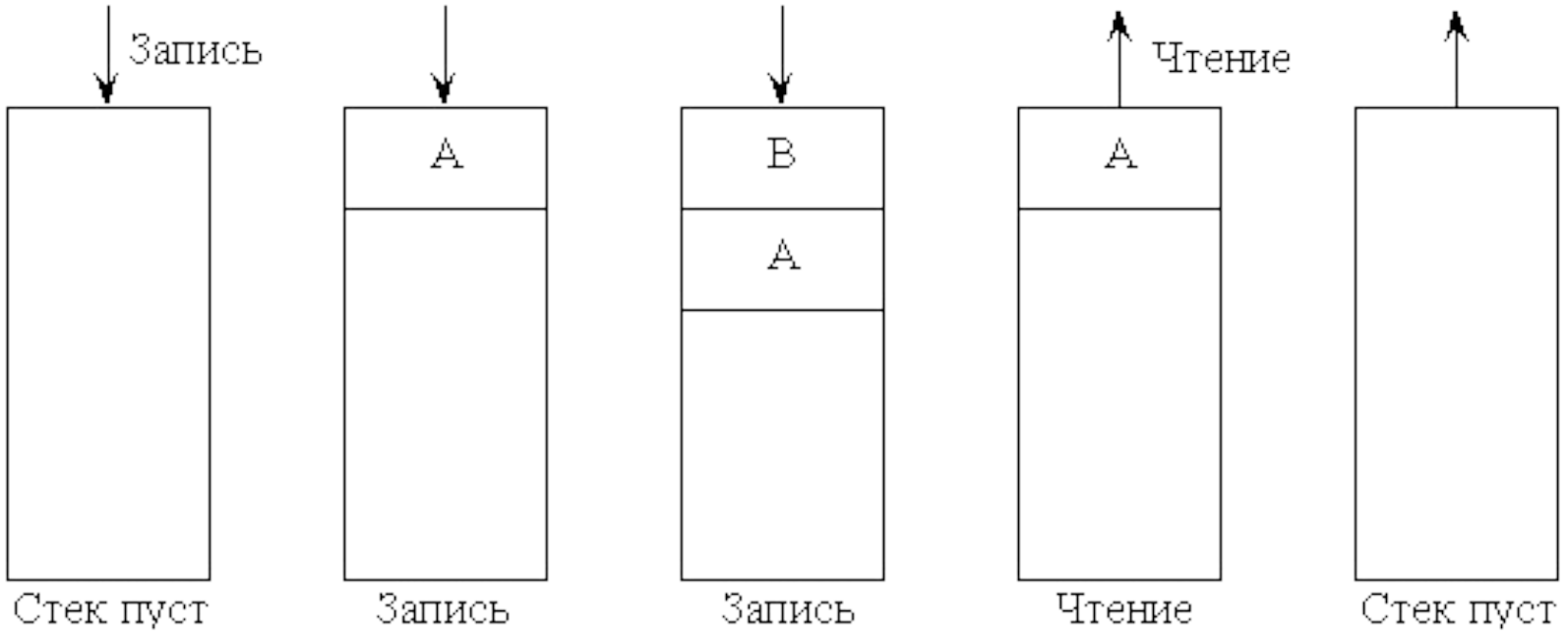
# Способы адресации

**Стековая адресация**

Стек образует множество логически взаимосвязанных ячеек, взаимодействующих по принципу LIFO.

Стековая память широко используется в современных ЭВМ. Хотя адрес обращения в стек отсутствует в команде, он формируется схемой управления:

# Способы адресации



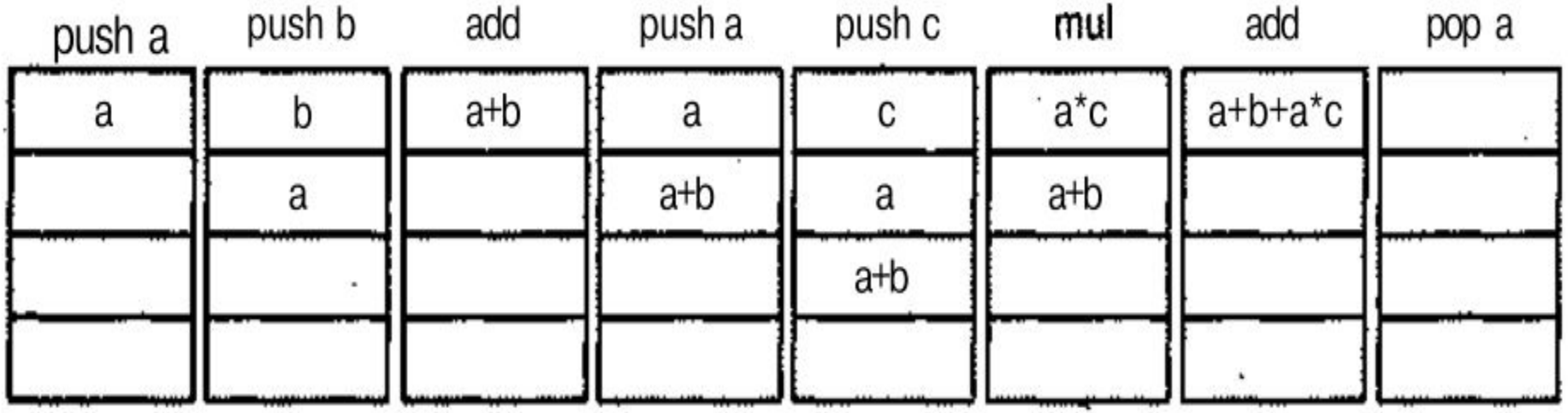
# Способы адресации

- Для чтения и записи доступна только вершина стека. Этот способ адресации используется, в частности, системой прерывания программ при вложенных вызовах подпрограмм.
- Стековая память реализуется на основе обычной памяти с использованием указателя стека и автоиндексной адресации.
- Запись в стек производится с использованием автодекрементной адресации, а чтение - с использованием автоинкрементной адресации.

# Способы адресации

- операнды перед обработкой помещаются в две верхних ячейки стековой памяти. Результат операции заносится в стек. Принцип действия стековой машины поясним на примере вычисления выражения:

$$a = a + b + a*c.$$



# Реализация и применение способов адресации

Использование всего набора способов адресации и их правильный выбор для каждого конкретного случая – позволяет:

- обеспечить доступ к структурированным данным;

- обеспечить перемещаемость программ и данных без изменения их кодов на этапе загрузки;

- сократить длину программного кода и число обращений к магистрали;

- адресовать большой объем памяти в условиях малой разрядности микропроцессора.

# Форматы команд и способы адресации

Обработка информации в микропроцессорной системе осуществляется автоматически, путем программного управления. Программа представляет собой алгоритм обработки данных, записанный в виде последовательности команд, которые должны быть выполнены системой для получения требуемого результата.

*Команда* представляет собой код, определяющий операцию обработки информации и данные, участвующие в этой операции.

По характеру выполняемых операций все возможные команды условно делят на несколько основных групп:

а) команды арифметической обработки;

б) команды логической обработки;

в) команды передачи (пересылки) кодов;

г) команды ввода-вывода;

д) команды передачи управления;

е) команды управления режимами работы микропроцессора и др.

# Форматы команды

Код команды можно представить состоящим из нескольких частей или полей, имеющих определенное функциональное назначение. В общем случае, команда состоит из *кода операции* (***КОП***) и адресной части (***Поля адресов***).


***Код операции*** – задает действие (сложение, умножение, передача), определяемое командой. А ***поле адресов*** – содержит информацию о расположении подлежащих обработке операндов, расположении результата, а в некоторых случаях, и о месте расположения следующей команды. Иногда в коде команды могут содержаться и сами данные, тогда для них выделяется отдельное поле – ***поле данных***.

***Формат команды*** – это совокупность сведений о длине, составе, назначении и взаимном расположении частей, или информационных полей в составе команды.