Адресность ЭВМ

Трёхадресная машина.

КОП	A1	A2	A3	= 10 байт
8 разрядов	24 разряда	24 разряда	24 разряда	

R1 := <A2>; R2 := <A3>; S := R1 ⊗ R2;
<A1> := S; {⊗ – операция}

Двухадресная машина.

КОП	A1	A2	= 7 байт
8 разрядов	24 разряда	24 разряда	

R1 : = <A1>; R2 := <A2>; S := R1 ⊗ R2;
<A1> := S;

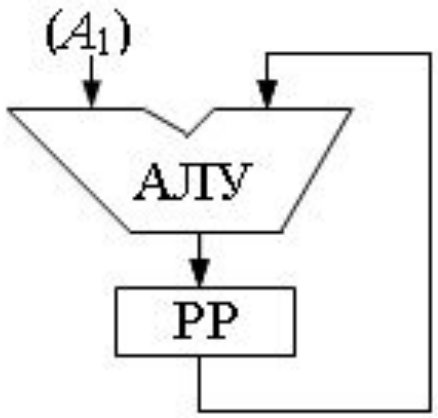
Результат операции по умолчанию помещается на место первого операнда, уничтожая его.

Одноадресная машина.

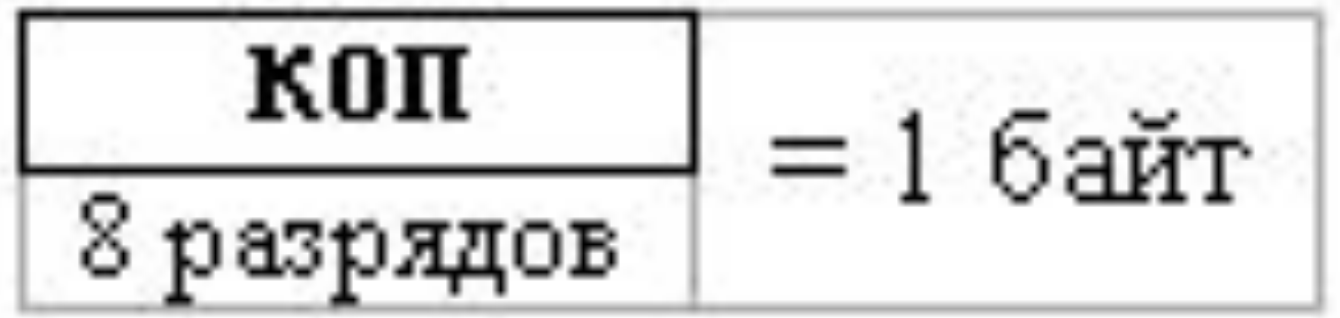
КОП	A1	= 4 байта
8 разрядов	24 разряда	

R1 := <A1>; S := S ⊗ R1;

Нужны команды :
СЧ A1
S := <A1>
и
ЗП A1
<A1> := S



Безадресная машина.



команды:

ВСТЕК A1

R1 := <A1>;

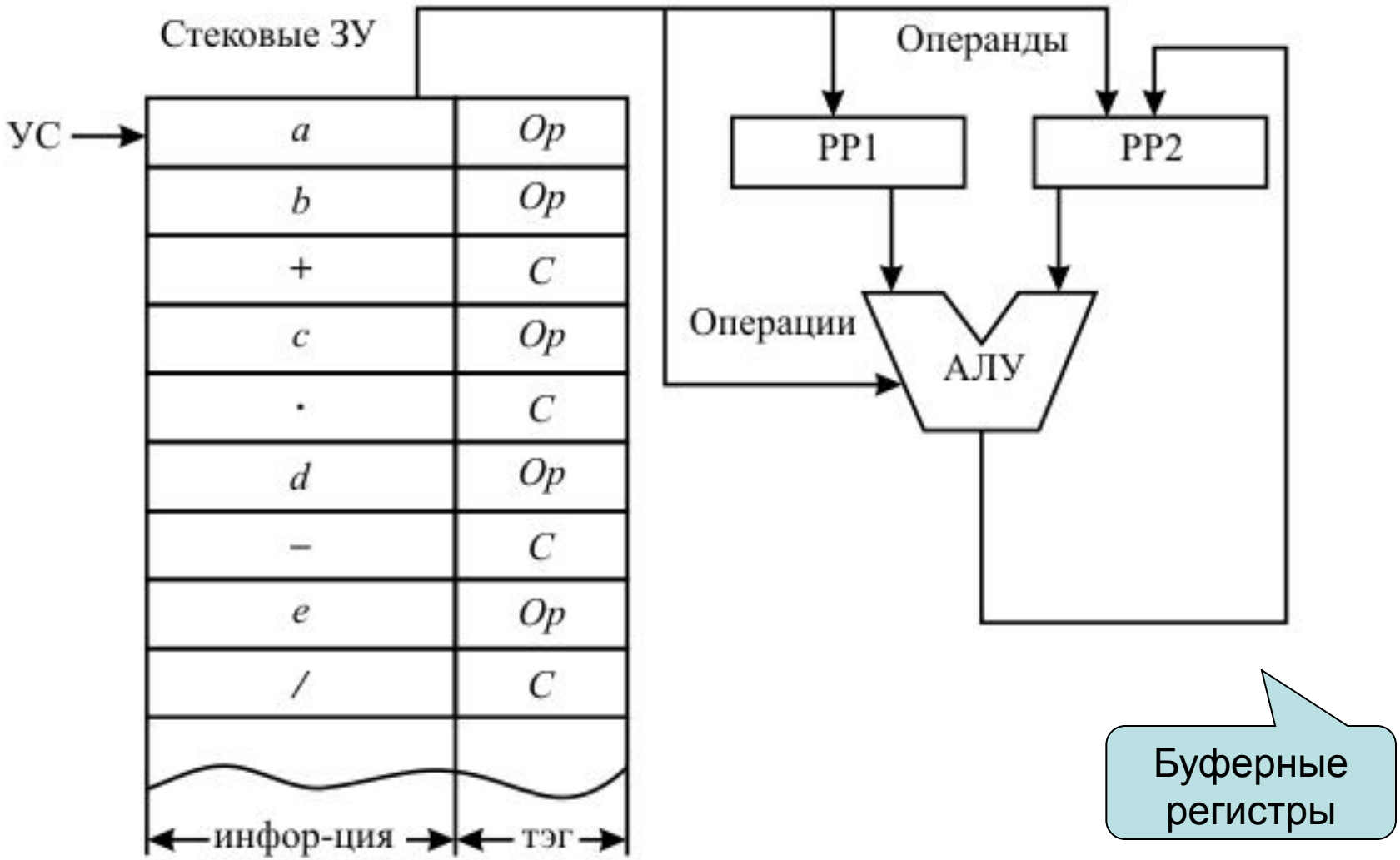
ИЗСТЕКА A1

<A1> := R1

Стековые ЭВМ

R1 := ИЗСТЕКА; R2 := ИЗСТЕКА; S := R1 $\otimes$ R2; ВСТЕК(S)

Безадресное *кодирование команд* ((a + b) * c - d) / e



Четырёхадресные, в четвёртом адресе которых дополнительно хранится ещё и адрес следующей выполняемой команды. Собственно, адресов может быть и больше, с помощью таких команд можно, например, реализовать функции от многих переменных

Четырехадресная система *кодирования* практического применения не получила. Основной причиной этого является существенное увеличение размера каждой команды и, соответственно, увеличение объема ЗУ, необходимого для размещения программы.

VLIW – very large instruction word

- Наличие в команде нескольких *кодов операций*

Указанные команды могут реализовывать оператор присваивания вида z:=k*(x+y) по схеме:

- R1 := <x>; R2 := <y>; S := R1+R2;
- R1 := <k>; S := S*R1; <z> := S

VLIW – very large instruction word

В компьютерах с такой архитектурой команда содержит два кода операции и четыре адреса аргументов:

КОП1	КОП2	A1	A2	A3	A4

Такие команды могут выполняться, например, по схеме:

```
R1 := <A2>; R2 := <A3>; S := R1 КОП1 R2;
R1 := <A4>; S := S КОП2 R1; <A1> := S
```

Сравнительный анализ ЭВМ различной адресности

Какая архитектура лучше ?

Пусть необходимо вычислить

$$x := a/(a+b)^2$$

- **Трёхадресная машина.**

СЛ	x	a	B	X := a+b
УМН	x	x	X	X := (a+b)^2
ДЕЛ	x	a	x	X := a/(a+b)^2

Длина программы: 3*10 = 30 байт.

- **Двухадресная машина.**

ПЕР	R	a	R := a
СЛ	R	b	R := a+b
УМН	R	R	R := (a+b)^2
ПЕР	X	a	x := a;
ДЕЛ	X	R	x := a/(a+b)^2

Длина программы: 5*7 = 35 байт.

Одноадресная машина.

СЧ	A	S := a
СЛ	B	S := a+b
ЗП	X	x := a+b
УМН	X	x := $(a+b)^2$
ЗП	X	
СЧ	A	S := a/$(a+b)^2$
ДЕЛ	X	
ЗП	X	

Длина программы: 8*4 = 32 байта.

Безадресная машина.

ВСТЕК	A	Поместить a в стек
ВСТЕК		Дублировать вершину стека
ВСТЕК	B	Теперь в стеке 3 числа: b, a, a
СЛ		В стеке два числа: b+a, a
ВСТЕК		Дублировать вершину стека, в стеке b+a, b+a, a
УМН		В стеке $(a+b)^2$, a
ОБМЕН		Поменять местами два верхних элемента стека
ДЕЛ		В стеке a/$(a+b)^2$
ИЗСТЕКА	X	Запись результата из стека в x

В данной программе использовались команды разной длины (безадресные и одноадресные). Длина программы: 3*4 + 6*1 = 18 байт.

Дробно-адресная архитектура

Память состоит из двух частей, каждая со своей независимой нумерацией ячеек:

адресуемая регистровая память и *основная(оперативная) память*.

- Адреса регистров - R1 и R2
- Адрес основной памяти A1 или A2.
- Первый вид команд будем называть командами *формата* ***регистр-регистр*** (обозначается RR),
- а вторые – ***формата регистр-память*** (обозначается RX).

Получим команды двух форматов длины 2 и 4 байта соответственно:

КОП	R1	R2	= 2 байта
1 байт	1 байт		

КОП	R1	A2	= 4 байта
8 бит	4 бита	20 бит	

$x:=a/(a+b)^2$

СЧ R1,a; R1 := a

СЧ R2,b; R2 := b

СЛ R2,R1; R2 := b+a=a+b

УМН R2,R2; R2 := (a+b)2

ДЕЛ R1,R2; R1 := a/(a+b)2

ЗП x,R1; x := R1= a/(a+b)2

...

Длина этого фрагмента программы равна
3*4+3*2 = 18 байт

Многообразием форматов команд

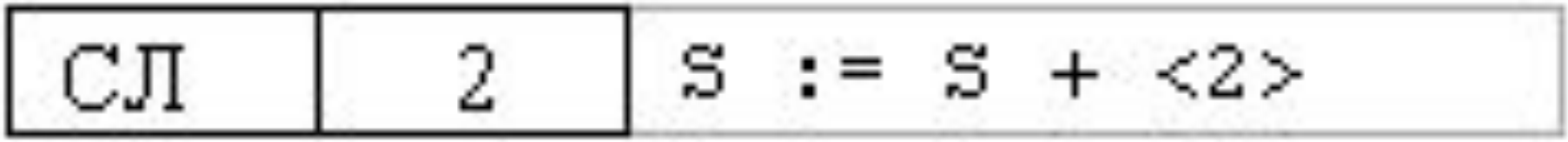
Современные ЭВМ обладают *многообразием форматов команд*. Например, на тех компьютерах, на которых Вы сейчас выполняете свои практические работы, реализованы около десяти форматов, а длина команд составляет от 1 до 6 байт.

Способы адресации

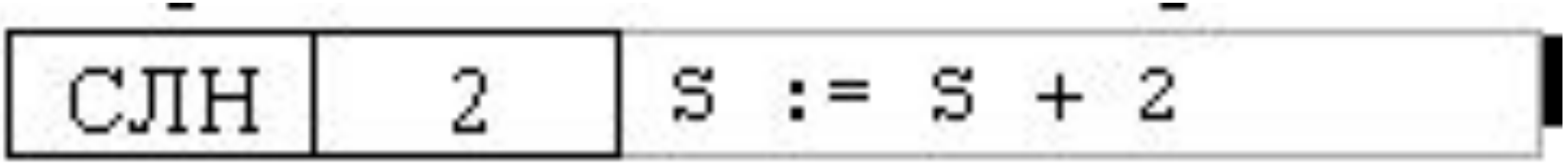
Способ адресации – это способ задания операндов внутри машинной команды.

Мнемоника кодов операций будет указывать на способ адресации

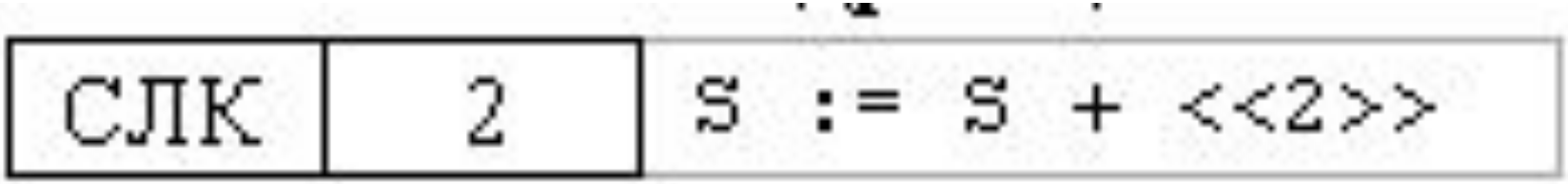
Прямой способ адресации.



Непосредственный способ адресации.



Косвенный способ адресации.



Пример с разными способами адресации

			Адрес	Значение
СЧ	2;	S:=<2>=3	000	1
СЛ	2;	S:=S+<2>=6	001	2
СЛН	2;	S:=S+2=8	002	3
СЛК	2;	S:=S+<<2>>=S+4=12	003	4

Значение регистра сумматора после выполнения одноадресных команд сложения с различными способами адресации.

Многообразие форматов данных

- регистр – регистр (RR);
- регистр – память, память – регистр (RX);
- регистр – непосредственный операнд в команде (RI);
- память – непосредственный операнд в команде (SI);
- память – память, т.е. оба операнда в основной памяти (SS)

Базирование адресов

Пусть в программе на одноадресной машине необходимо реализовать арифметический оператор присваивания $X:=(A+B)^2$....

СЧ A; S:=A

СЛ B; S:=A+B

ЗП R; R:=A+B – запись в рабочую переменную

УМ R; $S:=(A+B)^2$

ЗП X; $X:=(A+B)^2$

Базирование адресов

Пусть, например, наши переменные располагаются соответственно в следующих ячейках памяти:

A – в ячейке с адресом 10 000 000

B – в ячейке с адресом 10 000 001

X – в ячейке с адресом 10 000 002

R – в ячейке с адресом 10 000 003

Базирование адресов

Тогда приведённый выше фрагмент программы будут выглядеть следующим образом:

...

СЧ 10 000 000; S:=A

СЛ 10 000 001; S:=A+B

ЗП 10 000 003; R:=A+B

УМ 10 000 003; $S:=(A+B)^2$

ЗП 10 000 002; $X:=(A+B)^2$

- ...

Базирование адресов

Большинство адресов в нашей программе имеют вид B+Δ, где число B назовём ***базовым адресом*** программы или просто ***базой*** (в нашем случае B=10 000 000), а Δ – ***смещением*** адреса относительно этой базы.

команда *загрузки базы* (длина этой команды 4 байта):

ЗГБ	А1
8 бит	24 бита

Тогда наш фрагмент программы будет иметь такой вид:

```
    ...
ЗГБ 10 000 000
    ...
СЧ  000; S:=A
СЛ  001; S:=A+B
ЗП  003; R:=A+B
УМ  003; S:=(A+B)²
ЗП  002; X:=(A+B)²
    ...
```

Осталось выбрать длину смещения Δ. Вернёмся к рассмотрению дробноадресной ЭВМ, для которой реализовано базирование адресов. Например, пусть под запись смещения выделим в команде поле длиной в 12 бит. Тогда все команды, которые обращаются за операндом в основную память, будут в нашей дробноадресной ЭВМ более короткими:

Схема выполнения такой команды для формата регистр-память:

<R1> := <R1> ⊗ <B+A2>

или для формата память-регистр:

<B+A2> := <B+A2> ⊗ <R1>

Область, в которой находятся вычисляемые относительно базы ячейки основной памяти, обычно называется ***сегментом*** памяти – это сплошной участок памяти, начало которого задаётся в некотором регистре, называемом ***базовым***, или ***сегментным***. Такой приём – ***сегментирование*** памяти

Практически все современные ЭВМ производят сегментирование памяти

Современные ЭВМ обеспечивают одновременную работу с несколькими сегментами памяти и, соответственно, имеют несколько сегментных регистров.

Во многих архитектурах используются *специализированные* регистры, т.е. определённые регистры являются сегментными, на других могут производиться операции и т.д.