

Основы компьютерной техники (Computer Organization. Basis)

БГТУ

кафедра ПИ

доцент Самаль Дмитрий Иванович т. 293-23-79, [dmitry\_samal@mail.ru,](mailto:dmitry_samal@mail.ru) a.510б-5

Лекция 6

«Способы адресации» 2020



План лекции

1. Способы адресации операндов
2. Порядок байт в памяти при адресации

***Слайд*** 2



# Количество адресов в команде

Время выполнения программы и адресность команд

Одноадресная команда быстрее, но для реализации трёхадресной команды нужно три одноадресных.

Определяющим при выборе является тип алгоритмов, на которые ориентирована ВМ:

* последовательные
* параллельные
* комбинированные

***Слайд*** 3



# Количество адресов в команде

Время выполнения программы и адресность команд

В последовательных программах – результат предыдущей операции используется для последующей. Выгодна одноадресная команда.

В параллельных – результат пересылается в память.

Трёхадресные команды будут эффективнее.

В комбинированных – лучше будут одноадресные и полутороадресные.

***Слайд*** 4



# Способы адресации операндов

Один из центральных вопросов при проектировании ВМ

Исполнительный адрес операнда (АИСП) – двоичный код номера ячейки памяти, служащей источником/приёмником операнда (адрес на ША или номер регистра).

Адресный код команды (АК) - двоичный код в адресном поле команды, из которого необходимо сформировать исполнительный адрес операнда.

В современных ВМ как правило АИСП и АК не совпадают, требуется соответствующее преобразование.

Способ адресации – это способ формирования исполнительного адреса по адресному коду команды.

***Слайд*** 5



# Способы адресации операндов

Один из центральных вопросов при проектировании ВМ

Исполнительный адрес операнда (АИСП) – двоичный код номера ячейки памяти, служащей источником/приёмником операнда (адрес на ША или номер регистра).

Адресный код команды (АК) - двоичный код в адресном поле команды, из которого необходимо сформировать исполнительный адрес операнда.

В современных ВМ как правило АИСП и АК не совпадают, требуется соответствующее преобразование.

Способ адресации – это способ формирования исполнительного адреса по адресному коду команды.

***Слайд*** 6



# Способы адресации операндов

Способы адресации (по Цилькеру и Орлову):

* Непосредственная
* Прямая
* Косвенная
* Регистровая
* Косвенная регистровая
* Адресация со смещением
  + Относительная
  + Базовая регистровая
  + Индексная
* Страничная
* Блочная
* Стековая

***Слайд*** 7

# Способы адресации операндов

Непосредственная (Immediate addressing) – в адресном поле вместо адреса сам операнд.

|  |  |  |
| --- | --- | --- |
| Код операции | СА | Непосредственный операнд |

Операции сравнения, загрузки констант в регистры, арифм. операции.

Операнд – число в (обычно) доп. коде.

Основная проблема – не каждый операнд можно передать непосредственно – адр. часть команды как правило меньше машинного слова. 50-60% - до 8 бит, 70-80% - до 16 бит.

Плюсы – малое время выполнения команды, экономия

памяти.



***Слайд*** 8

# Способы адресации операндов

Прямая (Direct addressing) – адресный код прямо указывает номер ячейки памяти операнда.

Память

|  |  |  |
| --- | --- | --- |
| Код операции | СА | АК |

|  |
| --- |
|  |
| Операнд |
|  |

Плюсы – простота.

АИСП = АК

Существенный недостаток – ограниченный размер адресного пространства.



Главный недостаток?

Адрес в команде не может быть изменён в процессе вычислений -> ограничение на размещение программы в ОЗУ.

***Слайд*** 9

# Способы адресации операндов

Косвенная (Indirect addressing) – ограниченное адресное поле указывает указывает адрес ячейки, содержащей полноразрядный адрес памяти операнда.

Память

|  |  |  |
| --- | --- | --- |
| Код операции | СА | АК |

АИСП =(АК)

|  |
| --- |
|  |
| Исп. адрес операнда |
|  |
| Операнд |
|  |

Плюсы – можно изменять

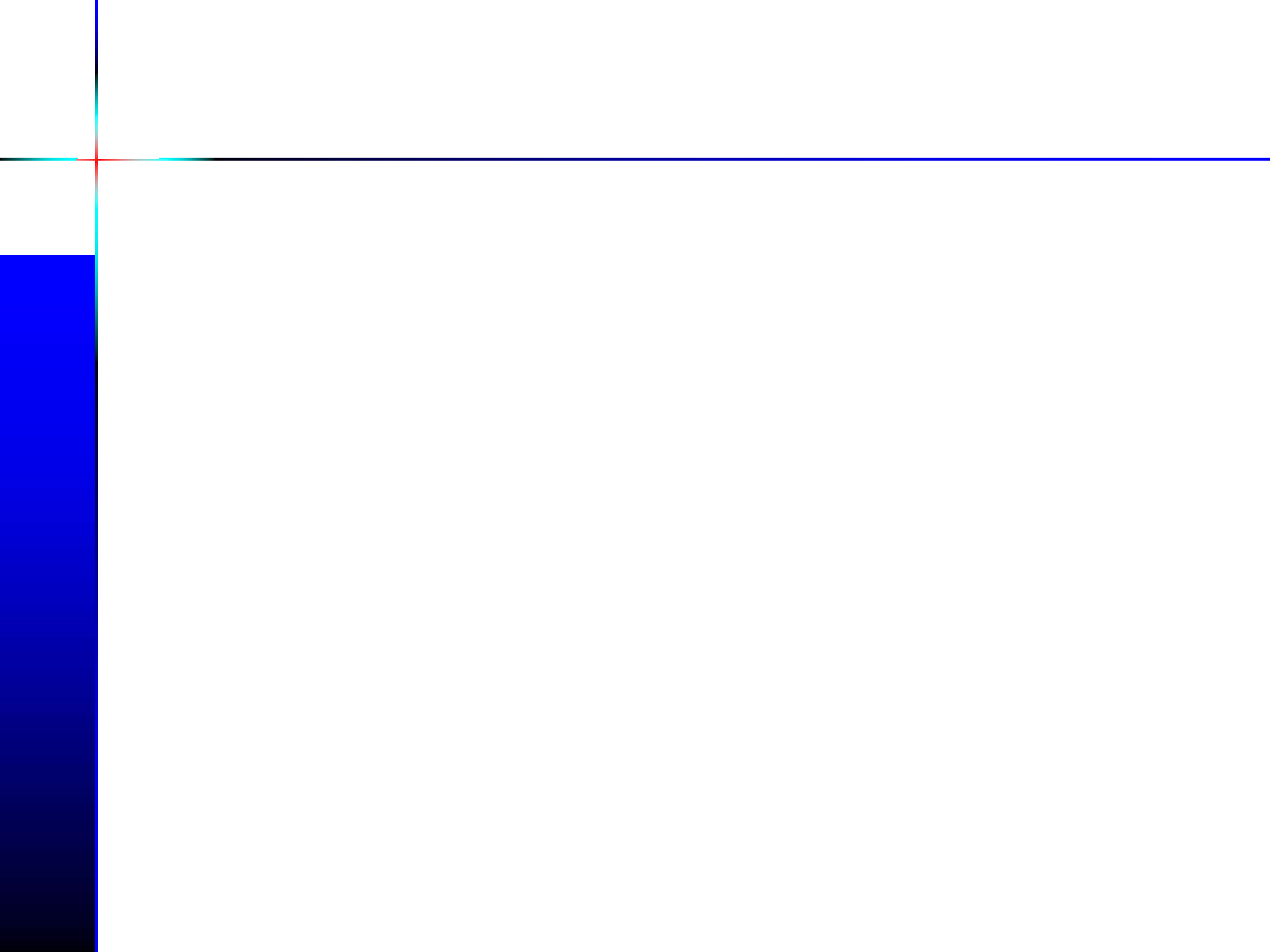
адреса операндов в процессе вычислений. Существенный недостаток – двухкратное обращение к памяти.

Многоуровневая или каскадная косвенная адресация – иногда удобна для обработки

многомерных массивов.



***Слайд*** 10



# Способы адресации операндов

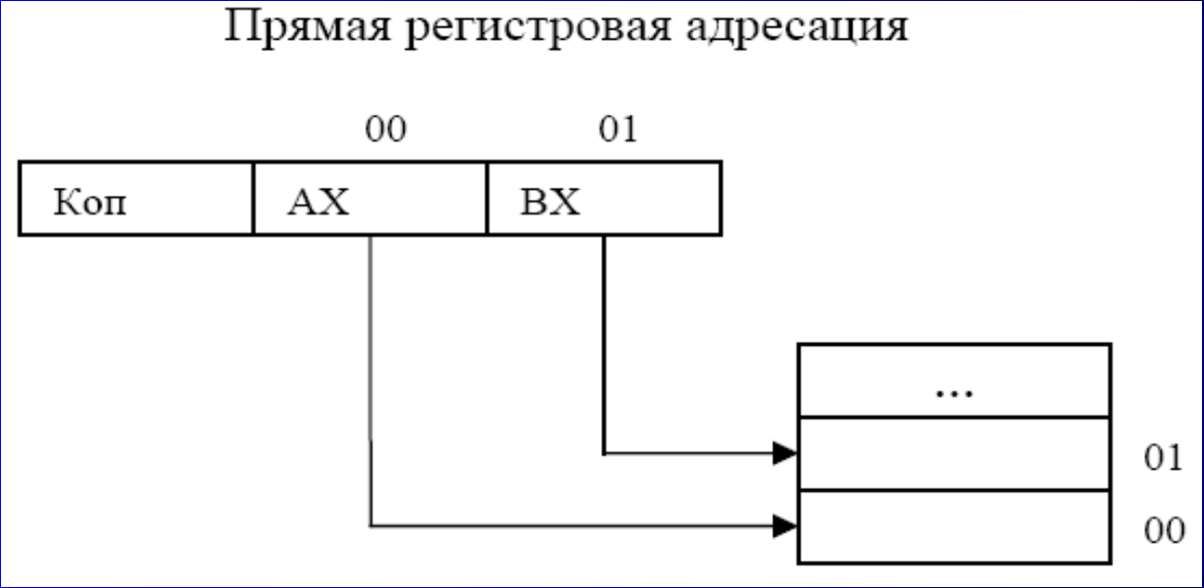
Регистровая (Register) – адресное поле указывает на номер регистра, содержащего операнд.

Регистры

|  |  |  |
| --- | --- | --- |
| Код операции | СА | R |

АИСП = R

|  |
| --- |
|  |
| Операнд |
|  |

По Добулевичу: Прямая регистровая

***Слайд*** 11



# Способы адресации операндов

Регистровая (Register) – адресное поле указывает на номер регистра, содержащего операнд.

Регистры

|  |
| --- |
|  |
| Операнд |
|  |

|  |  |  |
| --- | --- | --- |
| Код операции | СА | R |

АИСП = R

Плюсы – размер адресного поля 3-4 бита для 8-16 РОН. Исключение обращений к памяти.

Минусы – мало РОН.

***Слайд*** 12

# Способы адресации операндов

Косвенная регистровая (Register indirect) – адресное поле указывает на номер регистра, содержащего операнд.

Регистры

Память



|  |  |  |
| --- | --- | --- |
| Код операции | СА | R |

|  |
| --- |
|  |
| Исп. адрес операнда |
|  |

АИСП = (R)

|  |
| --- |
|  |
|  |
|  |
| Операнд |
|  |

Плюсы аналогичны обычной косвенной адресации – можно изменять адреса операндов в процессе вычислений.

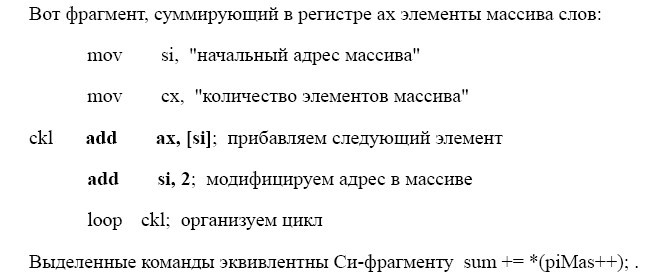
На одно обращение к памяти меньше.

***Слайд*** 13



# Способы адресации операндов

Косвенная регистровая (Register indirect) – адресное поле указывает на номер регистра, содержащего операнд.



***Слайд*** 14

# Способы адресации операндов

Адресация со смещением (Displacement) – исп. адрес формируется суммированием содержимого адресного поля команды с содержимым одного или нескольких регистров ЦП.

R2



Ак

|  |  |  |
| --- | --- | --- |
| Код операции | СА | R1 |

Регистры

Память

Составляющая адреса операнда

|  |
| --- |
|  |
|  |
|  |
| Операнд |
|  |

Составляющая адреса операнда

АИСП = Aк+(R1)+[(R2)]



***Слайд*** 15

# Способы адресации операндов

Адресная часть включает как минимум одно поле Ак и содержит константу. Смысл константы может меняться – базовый адрес, либо смещение.

Регистры могут быть как РОН, так и специализированные

– базовый и (или) индексный.

Спец регистры могут указываться по умолчанию, тогда только одно поле Ак.

Если РОН – то дополнительно R1 (R2, R3…). Дополнительный R (обычно РОН) может появляться для указания масштабного коэффициента.

Наиболее общий вариант – два поля Ак и R.



***Слайд*** 16

# Способы адресации операндов

Один из вариантов – вместо суммирования – конкатенация.

R2



Ак

|  |  |  |
| --- | --- | --- |
| Код операции | СА | R1 |

Регистры

Память

Составляющая адреса операнда

|  |
| --- |
|  |
|  |
|  |
| Операнд |
|  |

Составляющая адреса операнда

АИСП = (R1) || Aк +[(R2)]



***Слайд*** 17

# Способы адресации операндов

Адресная со смещением: относительная – для получения исполнительного адреса поле Ак складывается с содержимым счётчика команд.

|  |  |  |
| --- | --- | --- |
| Код операции | СА | Ак |

АИСП = Aк+(СК)



Счётчик команд (Program Counter) Адрес текущей команды

Память

Базируется на свойстве локальности – обращение к ячейкам в непосредственной близости от текущей. Плюсы – можно экономить на разрядности Ак, программа перемещаема в памяти.



Смещение = Aк

|  |
| --- |
|  |
| Текущая ком-да |
|  |
| Операнд |
|  |

***Слайд*** 18



# Способы адресации операндов

Адресная со смещением: базовая регистровая адресация

– регистр, называемый базовым, содержит полноразрядный адрес, поле Ас – смещение.

Память



Смещение

Базовый регистр (БР)

Базовый адрес

|  |  |  |
| --- | --- | --- |
| Код операции | СА | Ас |

|  |
| --- |
|  |
|  |
|  |
| Операнд |
|  |

АИСП = Aс+(БР)

Базовый регистр – может быть задан неявно (если специальный регистр).

***Слайд*** 19



# Способы адресации операндов

Адресная со смещением: базовая регистровая адресация Базовый регистр – может быть задан явно (если РОН).

Память

РОНы

|  |
| --- |
|  |
|  |
|  |
| Операнд |
|  |



Смещение

АИСП = Aс+(R)

Базовый адрес

|  |  |  |  |
| --- | --- | --- | --- |
| Код операции | СА | R | Ас |

Способ используется для доступа к элементам массива.

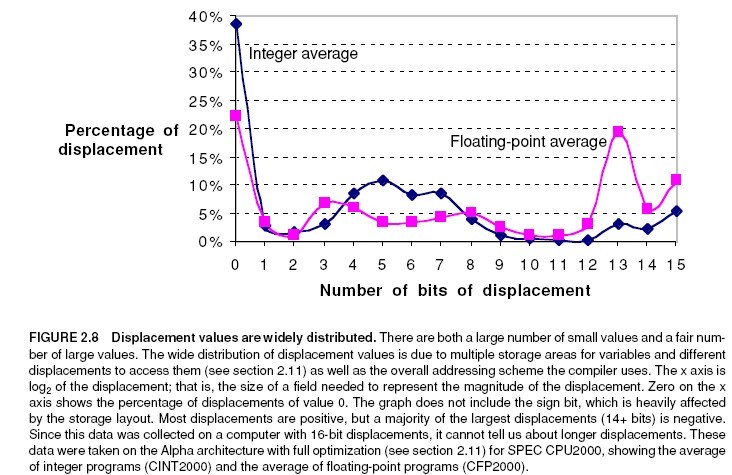
Смещение – меньшая длина, чем полный адрес -> команда короткая.

***Слайд*** 20



# Способы адресации операндов

Адресная со смещением: базовая регистровая адресация

***Слайд*** 21



# Способы адресации операндов

Адресная со смещением: индексная адресация – поле Ас содержит адрес памяти, регистр (явный/неявный) – смещение.

Память



Базовый адрес

Индексный регистр (ИР)

Индекс

|  |  |  |
| --- | --- | --- |
| Код операции | СА | Ас |

|  |
| --- |
|  |
|  |
|  |
| Операнд |
|  |

АИСП = Aс+(ИР)

Разрядность Ас – больше чем в предыдущем варианте. Вычисления аналогичны.

***Слайд*** 22



# Способы адресации операндов

Адресная со смещением: индексная адресация – поле Ас содержит адрес памяти, регистр (явный) – смещение.

Память

РОНы

|  |
| --- |
|  |
|  |
|  |
| Операнд |
|  |



Базовый адрес

АИСП = Aс+(R)

Индекс

|  |  |  |  |
| --- | --- | --- | --- |
| Код операции | СА | R | Ас |

Удобный механизм для организации итеративных вычислений.

***Слайд*** 23

# Способы адресации операндов

Адресная со смещением: индексная адресация

Пример: Дан массив чисел (последовательно в памяти), начиная с адреса N. Необходимо увеличить на 1 все элементы массива.

Выборка всех элементов, инкрементация и возврат обратно в память. Адреса – N, N+1, N+2…

Значение N – из Ас, а индексный регистр – 0, +1,+2 …

Типичный случай – индексный регистр изменяется автоматически как часть машинного цикла (автоиндексирование).



Почему нельзя сделать то же самое с помощью базовой регистровой?

***Слайд*** 24



# Способы адресации операндов

Адресная со смещением: индексная адресация Если индексный регистр – то неявное автоиндексирование называется автоинкрементная

адресация (autoincrement).

АИСП = Aс+(R), R <- (R)+1 R <- (R) +1 , АИСП = Aс+(R),

Постинкрементное автоиндексирование Преинкрементное автоиндексирование

Автодекрементная адресация (autodecrement).

АИСП = Aс+(R), R <- (R) -1 R <- (R) -1 , АИСП = Aс+(R),

Постдекрементное автоиндексирование Предекрементное автоиндексирование

Индексная адресация с масштабированием и смещением – содержимое инд. регистра умножается на масш. коэф. и суммируется с Ас. Коэф. = 1,2,4,8 (Intel).

***Слайд*** 25

# Способы адресации операндов

Страничная адресация

Разбиение адресного пространства на страницы. Начальный адрес страницы – база.

Старшая часть адреса страницы – в спец. регистре (РАС) Адресный код – смещение внутри страницы.

Регистр адреса страницы (РАС)

|  |  |  |
| --- | --- | --- |
| Код операции | СА | Ас |

Смещение от начала адреса страницы

AEL0

Ас

АEHi

Старшая часть адреса страницы

АEHi xxx (xxx= 000)

Память



АE = AEHi || AEL0

xxx = AEL0

|  |
| --- |
|  |
|  |
|  |
| Операнд |
|  |
|  |

***Слайд*** 26



# Способы адресации операндов

Блочная адресация

Используется в командах, обрабатывающих блок данных, расположенный в последовательных ячейках памяти.

Блок обычно описывается – адрес первой или последней ячейки памяти + количество элементов блока (в байтах либо ячейках).

Может быть так же использован специальный признак

«конец блока», идущий за последним элементом блока.

Удобен для работы с внешними ЗУ, операциях с векторами.

***Слайд*** 27



# Способы адресации операндов

Стековая адресация

Используется в ВМ со стековой архитектурой.

Стековый доступ к памяти происходит в большинстве процессоров в следующих ситуациях:

* 1. При выполнении команд стекового доступа: "поместить в стек" **push** или "извлечь из стека" **pop**.
  2. При выполнении команды вызова подпрограммы и при возврате из нее.
  3. При входе в прерывание и возврате из него.

***Слайд*** 28

# Способы адресации операндов

Положение стека в адресуемой памяти определяется содержимым указателя стека.

Указатель стека программно доступен, программист может задать его значение обычной командой пересылки: **mov sp,#1000h -** задает положение стека, начиная с адреса 1000h.

Содержимое указателя стека используется как адрес операнда-приемника при записи или как адрес операнда- источника при считывании из стека.

При обращении к стеку автоматически модифицируется и содержимое указателя стека, чтобы обеспечить следующее обращение к очередной ячейке стека.

Обращение к стеку - косвенно-регистровая адресация

через *указательстека*с автоиндексацией.



***Слайд*** 29

# Способы адресации операндов

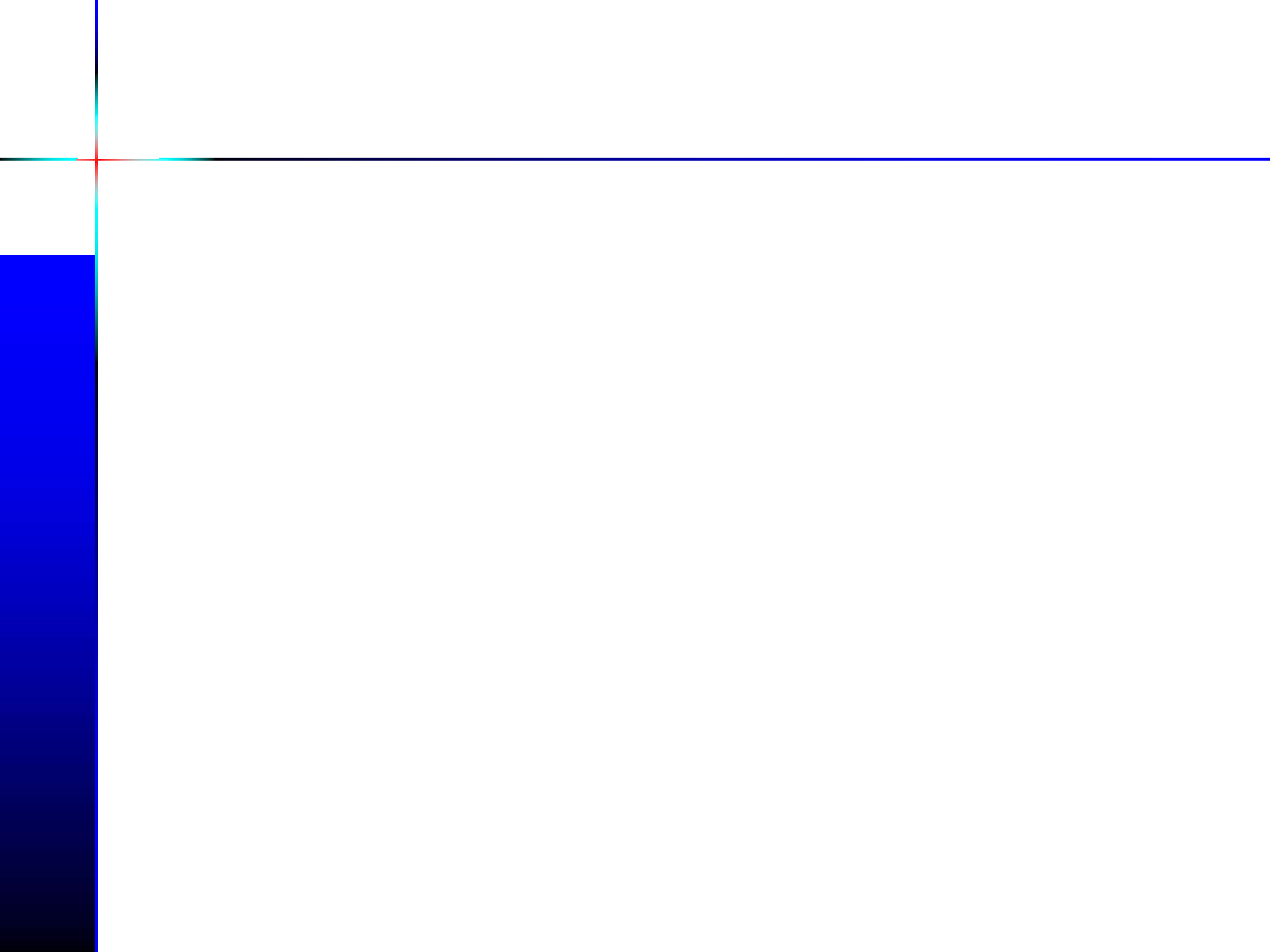
Стек может быть организован по разному:

1. Направление роста стека. Если при записи в стек, содержимое указателя стека автоматически увеличивается (и, соответственно, при считывании автоматически уменьшается), то говорят, что стек растет в сторону увеличения адресов. В противоположном случае говорят, что стек растет в сторону уменьшения адресов.
2. Если модификация указателя стека выполняется до записи и соответственно после считывания, то указатель стека всегда указывает на последнюю занятую ячейку стека. Наоборот, если модификация производится после записи и до считывания, указатель стека всегда указывает на первую

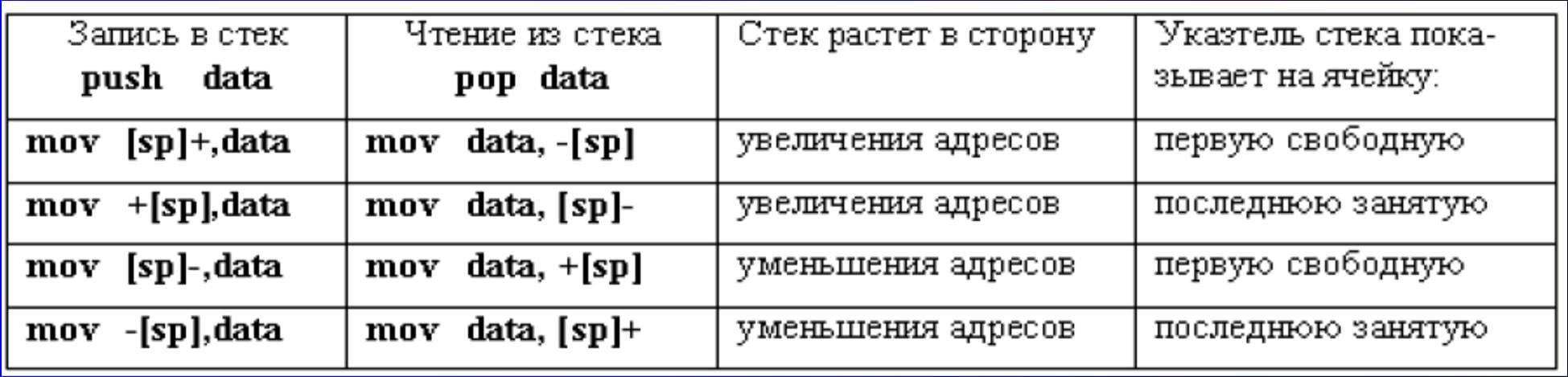
свободную ячейку стека.



***Слайд*** 30



# Способы адресации операндов

Стек может быть организован по разному:

***Слайд*** 31

# Способы адресации операндов

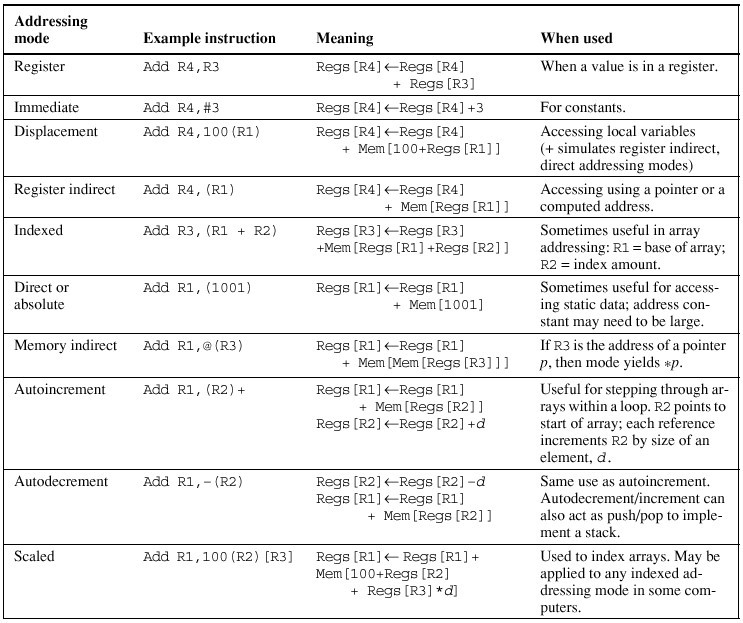


***Слайд*** 32



# Способы адресации операндов

***Слайд*** 33



**Регистровая**

**Непосредственая**

**Со смещением**

**Регистровая косвенная Индексная**

**Прямая**

**Ковсенная**

**Автоинкрементная**

**Автодекрементная**

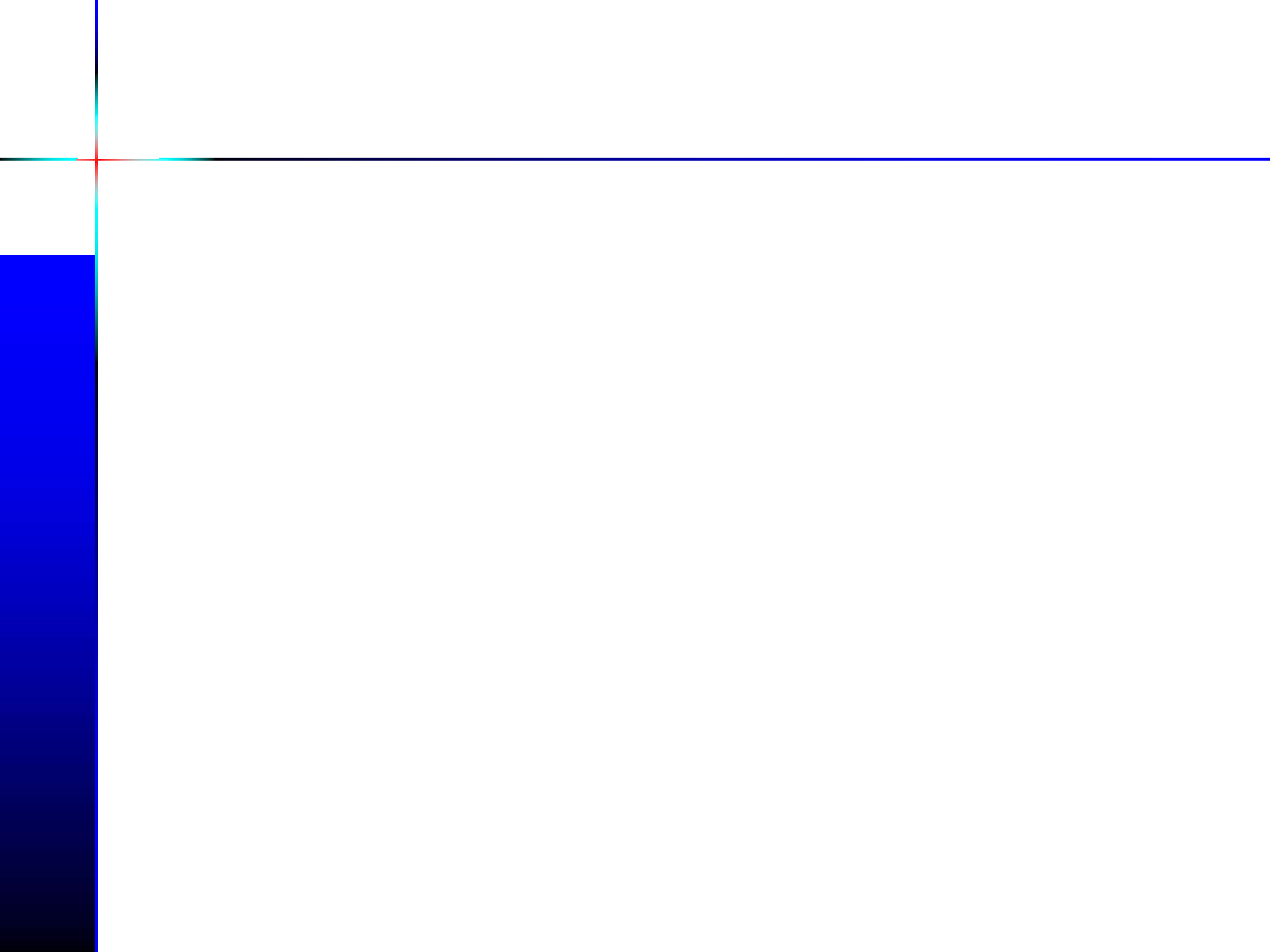
**Индексная со смещением и масштабированием**



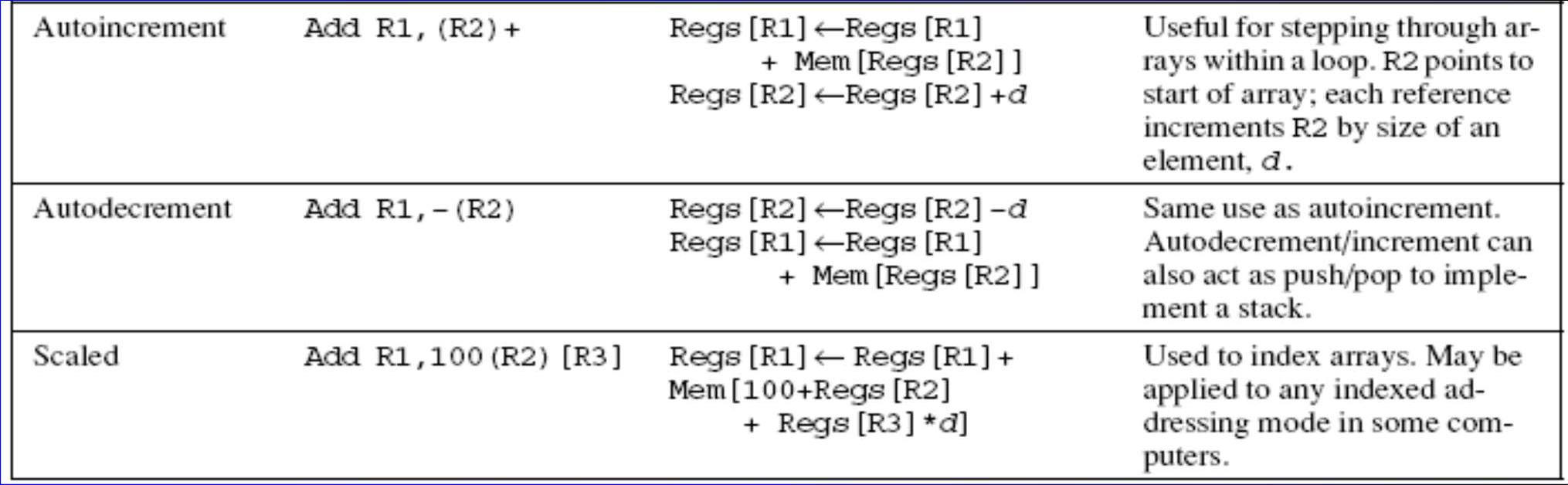
# Способы адресации операндов

34

|  |  |  |
| --- | --- | --- |
| **Метод адресации** | **Пример команды** | **Смысл команды метод** и**спользования** |
| Регистровая | Add R4,R3 | R4<-R4+R3 Требуемое значение в регистре |
| Непосредственная или литеральная | Add R4,#3 | R4 <- R4+3 Для задания констант |
| Базовая со смещением | Add R4,100(R1) | R4 <- R4+M[100+R1]  Для обращения к локальным переменным |
| Косвенная регистровая | Add R4,(R1) | R4 <- R4+M[R1]  Для обращения по указателю или вычисленному адр. |
| Индексная | Add  R3,(R1+R2) | R3 <- R3+M[R1+R2] Иногда полезна при работе с массивами: R1 - база, R3 - индекс |
| Прямая или абсолютная | Add R1,(1000) | R1 <- R1+M[1000]  Иногда полезна для обращения к стат. данным |
| Косвенная | Add R1,@(R3) | R1 <- R1+M[M[R3]] Если R3-адрес указателя p, то выбирается значение по этому указателю |
| Автоинкрементная | Add R1,(R2)+ | R1 <- R1+M[R2]  R2 <- R2+d Полезна для прохода в цикле по массиву с шагом: R2 - начало массива В каждом цикле R2 получает приращение d |
| Автодекрементная | Add R1,-(R2) | R2 <- R2-d  R1 <- R1+M[R2] Аналогична предыдущей  Обе могут использоваться для реализации стека |
| Индексная со смещением и масштабированием | Add R1, 100(R2)[R3] | R1<-R1+M[100]+R2+R3\*d ***Слайд***  Для индексации массивов |



# Способы адресации операндов



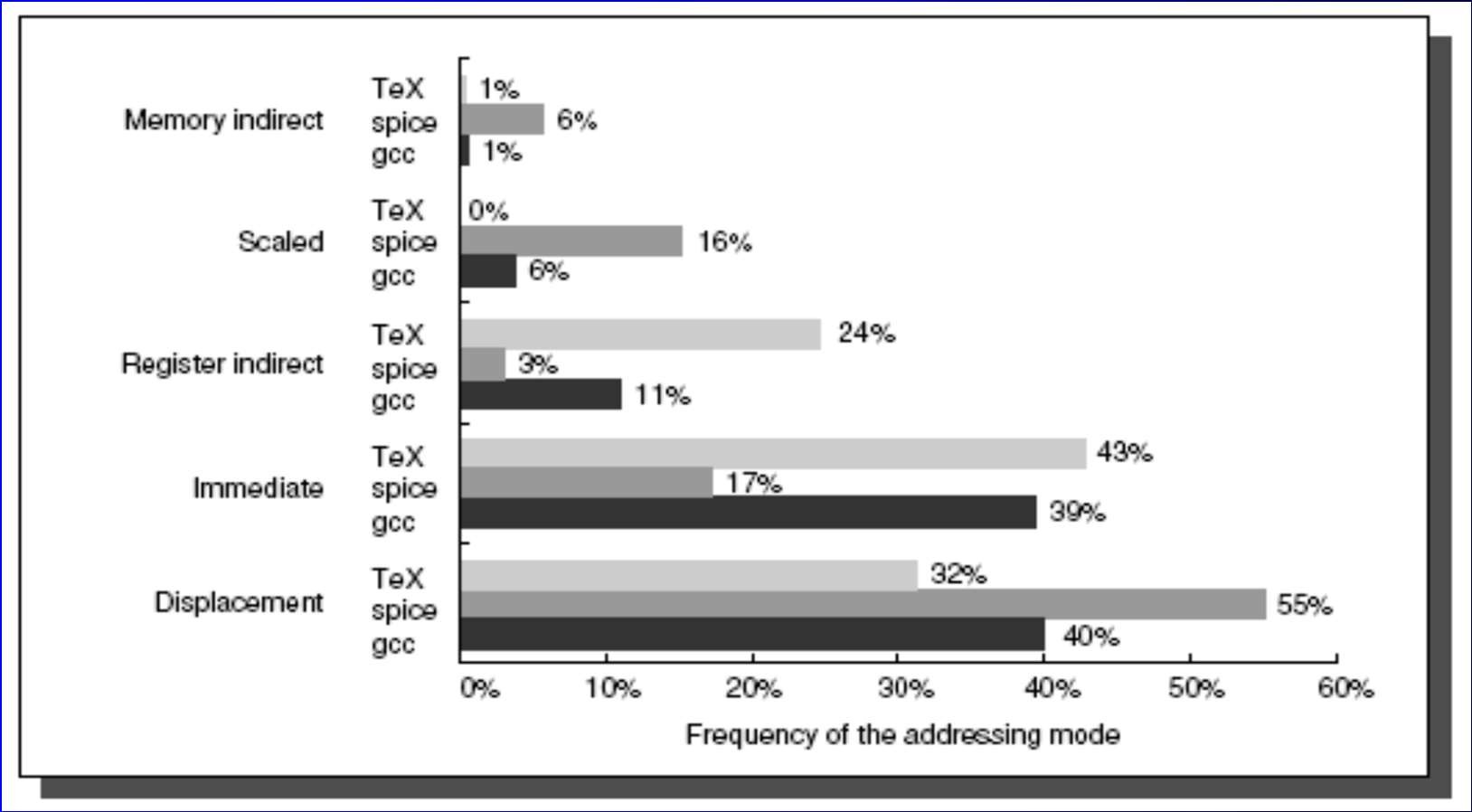
d- варьируется от размера данных (1,2,4,8 байт), useful только при последовательном размещении в памяти.

RISC процессоры используют адресацию со смещением для эмуляции регистровой косвенной (0 в поле адреса) и эмулируют прямую адресацию (с 0 в базовом регистре).

***Слайд*** 35

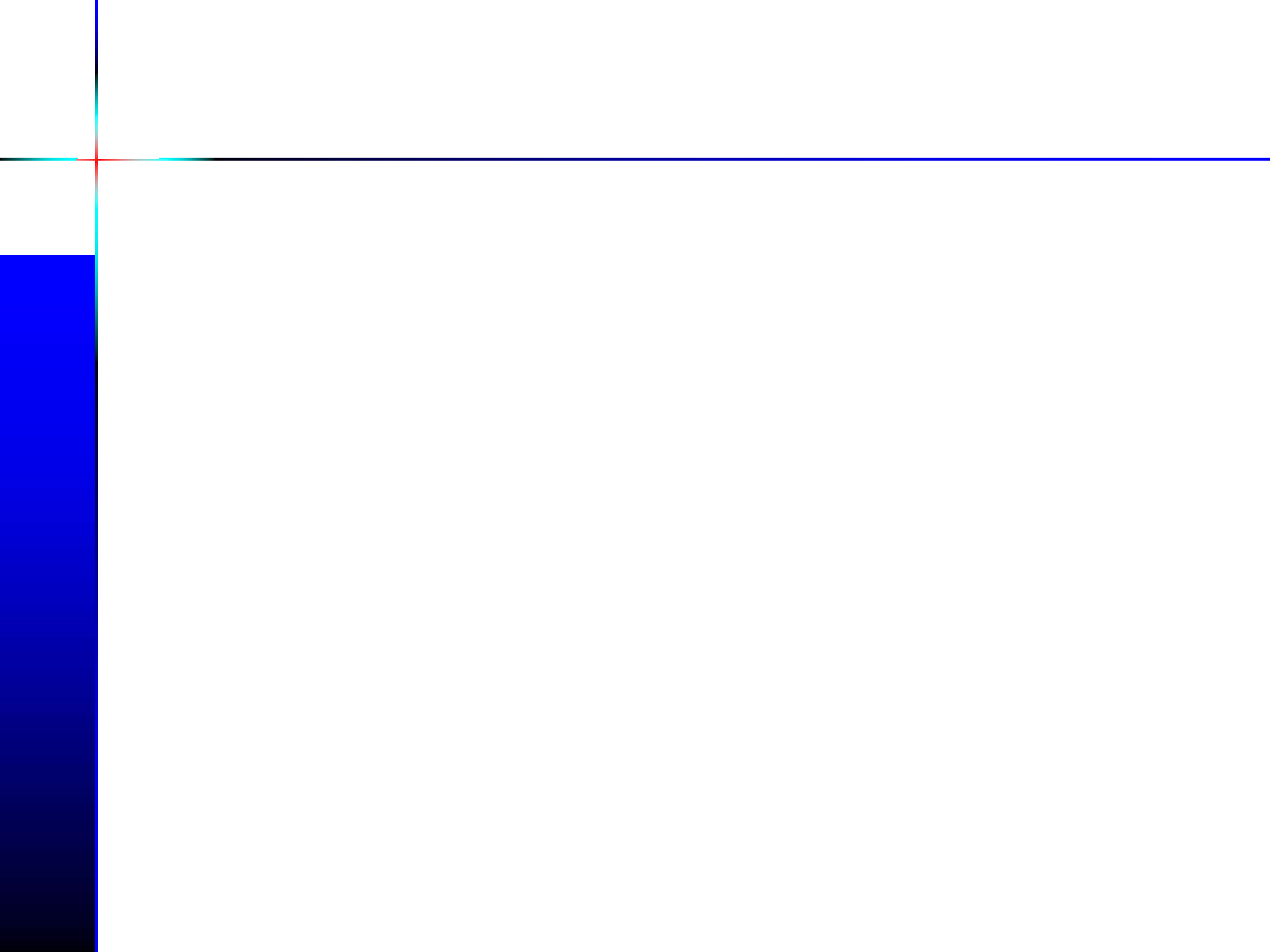
# Способы адресации операндов

Данные для DEC VAX:



RISC архитектура - преимущественный способ адресации: регистровая. Для аккумуляторной архитектуры главные

способы адресации - прямая и непосредственная.

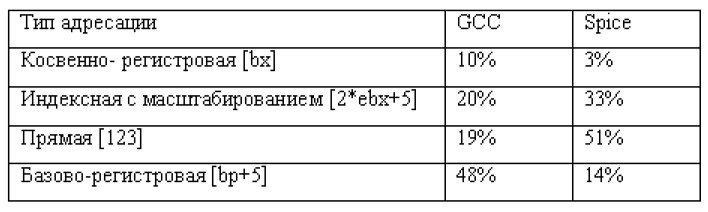


***Слайд*** 36



# Способы адресации операндов

Данные для Intel процессоров:



Наиболее активно используется прямая и базово- регистровая адресации, хотя интенсивность использования различных способов тесно связана с решаемой задачей.

***Слайд*** 37

Big-endian vs Little endian

* Little endian - младший байт по младшему адресу

Intel, DEC

(e.g., Base Address+0 Byte0, Base Address+1 Byte1, Base Address+2 Byte2, Base Address+3 Byte3)

* Big endian – старший байт по младшему адресу (E.g.,

Base Address+0 Byte3, Base Address+1 Byte2, Base Address+2 Byte1, Base Address+3 Byte0)

Motorola,

IBM

* Что лучше?
  + Printing, Binary-to-decimal conversion, sign-testing, multiple precision math routines
* Различные форматы файлов используют различный порядок:
  + Big Endian – Photoshop, TCP/IP sockets, JPEG, etc.

– Little Endian – BMP, GIF, RTF, etc.



***Слайд*** 38