IX. Принципы построения и архитектура ЭВМ

Общие принципы построения современных ЭВМ

Принципы Фон-Неймана

-Двоичное кодирование информации

-Программное управление

-Адресность памяти

-*Однородность памяти*

ОКОД, SISD

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Команды | Устройство | |  |  |
|  |  |  |  |
|  | Адреса | управления | |  |  |
|  |  |  |  |  |
| Память | команд и |  | Управление | |  |
| данных |  |  |
|  |  |  |  |  |
|  | Данные, |  |  |  |  |
|  | Обрабатывающее | |  |  |
|  | результаты |  |  |
|  | устройство | |  |  |
|  |  |  |  |
|  |  |  |  |  |  |



-Гарвардская архитектура

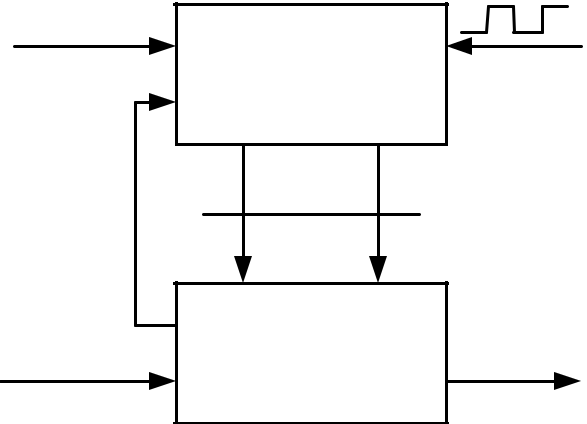
(ОП для хранения команд и ОП для хранения данных)

Принстонская архитектура

(ОП для хранения команд и данных)

Принципы микропрограммного управления

|  |  |  |  |
| --- | --- | --- | --- |
| КОП |  | C |  |
|  | УУ |  |  |
| Состояние | ... | Микрокоманда V |  |
|  |  |
| ОУ |  |  |
|  |  |  |
| Операнды | ОУ | Операнды |  |



Любое цифровое устройство можно рассматривать, как совокупность операционного и управляющего блока.

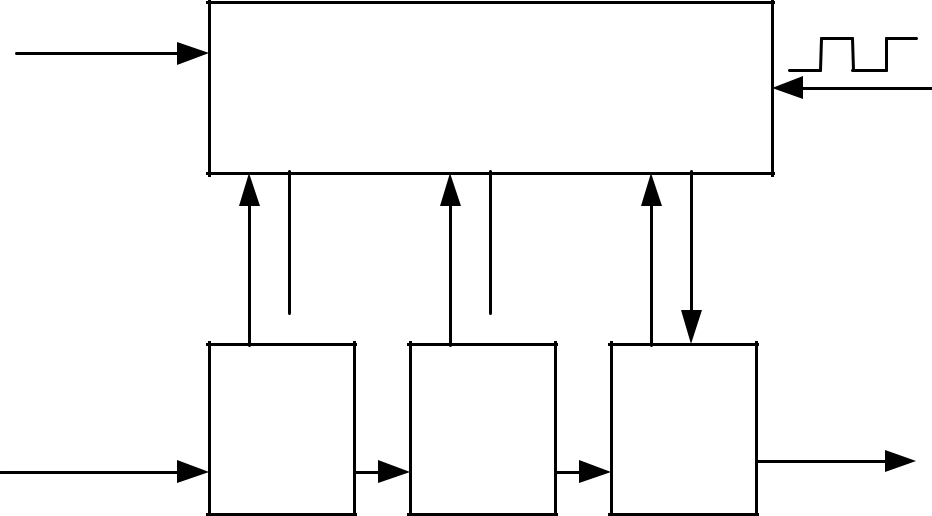
Любая команда или последовательность команд реализуется в операционном блоке за несколько тактов

Последовательность сигналов управления должна выдаваться устройством управления в соответствии с поступающей на вход командой и текущим состоянием операционного блока

Состояние линий управления в каждом такте задает микрокоманду. Совокупность микрокоманд, необходимых для реализации команды называется микропрограммой.

Принцип конвейерной обработки

КОП



|  |  |  |
| --- | --- | --- |
| УУ | C |  |
|  |  |

...

Состояние 

ОУ1..ОУn

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Операнды | ОУ 1 | ОУ 2 | ОУ n | Операнды |

Конвейерная обработка представляет собой процесс, при котором сложные действия разделяются на более короткие стадии. Их параллельное выполнение для последовательности действий позволяет более полно использовать обрабатывающие ресурсы конвейера.

Структура современных ЭВМ

-Центральное процессорное устройство (ЦПУ).

* Арифметико-логическое устройство (АЛУ) -Устройство управления (УУ)

-Регистры общего назначения (РОН)

-Основная память

-Система ввода-вывода

-Управление внешними устройствами

-Внешняя память

-Система передачи информации

-Система синхронизации

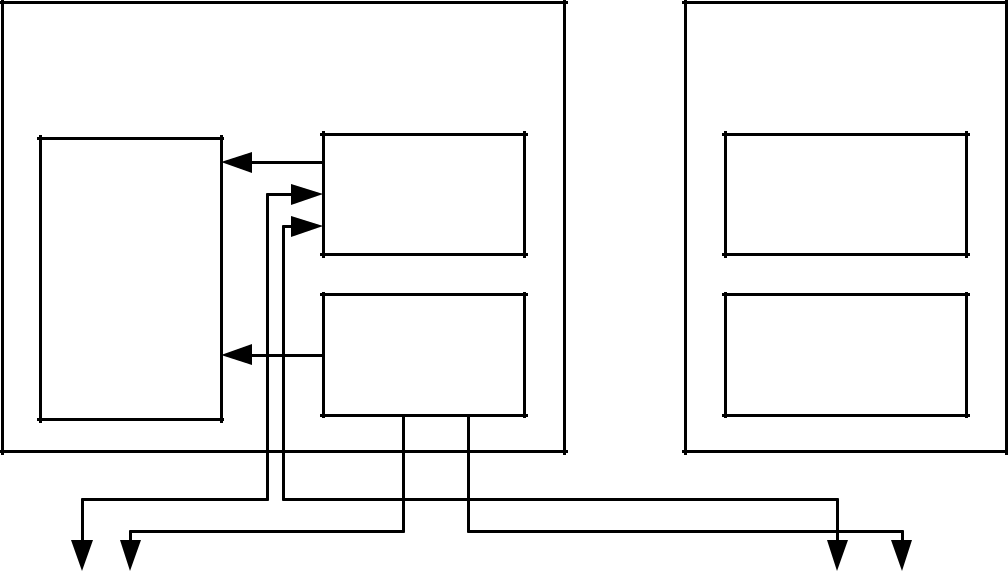
-Система прерываний

-Система прямого доступа к памяти

-Система подвода питания/земли и система энергосбережения Система контроля и повышения отказоустойчивости

ЭВМ с непосредственными связями

ЦПУ ОП

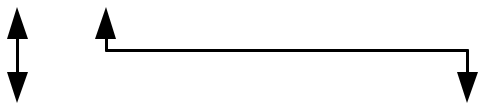


РОН  Данные

АЛУ

УУ  Команды

|  |  |  |
| --- | --- | --- |
| СВВ |  | ВП |
|  |  |  |



|  |  |  |
| --- | --- | --- |
| УВВ | ... | УВВ |
|  |  |  |

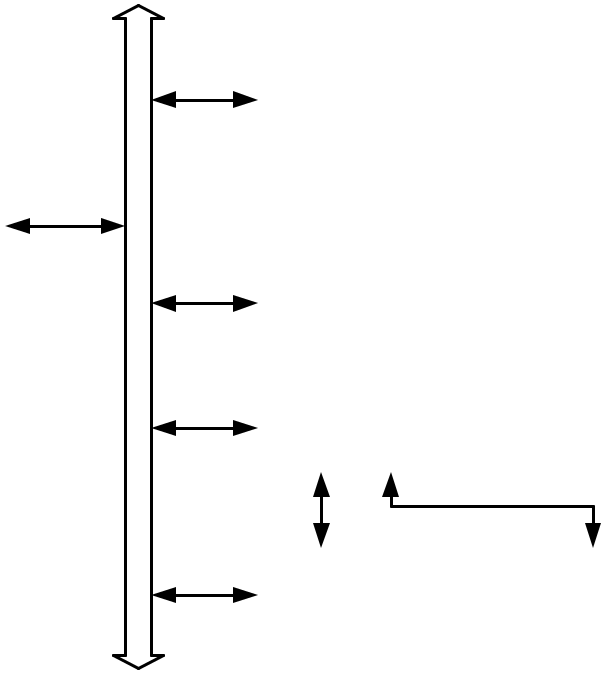
-(+) При построении оптимальных линий связи вычислительная машина обладает максимальным быстродействием.

(-) Ограничение на количество выводов микросхем не позволяет организовать широкие шины.

(-) Реконфигурация системы требует изменения характеристик линий связи.

ЭВМ с магистральной структурой

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | ОП | |  |
|  |  |  |  |  |  |  |  |
|  | ЦПУ | |  |  | Данные |  |  |
|  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  | АЛУ |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  | Команды |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  | УУ |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  | ВП | |  |
|  |  |  |  |  |  |
|  |  |  |  |
|  |  | |  |  |  |  |  |
|  | РОН |  |  |  |  |  |  |
|  |  |  |  | СВВ | |  |
|  |  |  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |



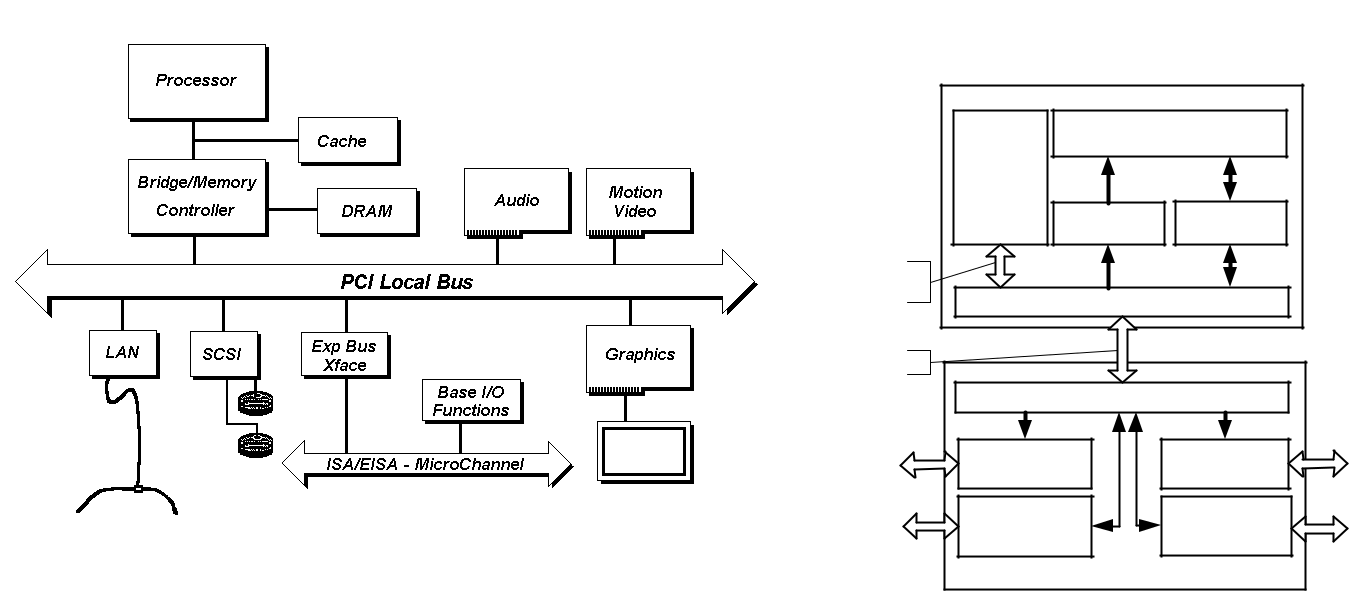
|  |  |  |
| --- | --- | --- |
| УВВ | ... | УВВ |
|  |  |  |

* Общая шина позволяет легко реконфигурировать систему.

(-) Шина является узким местом.

* Шина, используемая всеми устройствами системы для передачи данных называется системной.
* Для разгрузки системной шины используют иерархию шин.
* По назначению, разделяют шины адреса, шины данных и шины управления.

Примеры построения ЭВМ с иерархией шин

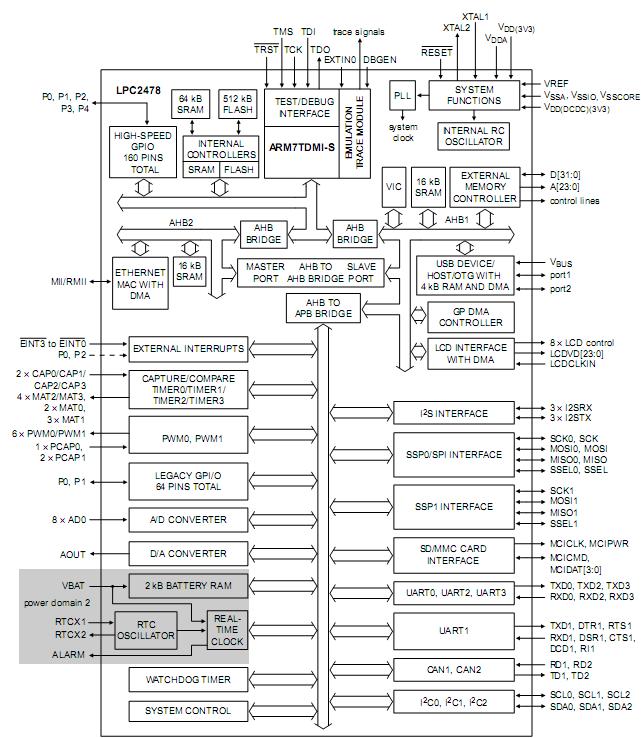


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | Процессор |  |
|  |  | Исполнительные | |  |
|  | Кэш | устройства | |  |
|  | память |  |  |  |
|  | второго |  |  |  |
|  | уровня | Конвейер | Конвейер |  |
|  |  | команд | данных |  |
| Шина второго |  |  |  |  |
| плана | Устройство шинного интерфейса | | |  |
|  |  |
| Системная шина |  |  |  |  |
|  | Устройство шинного интерфейса | | |  |

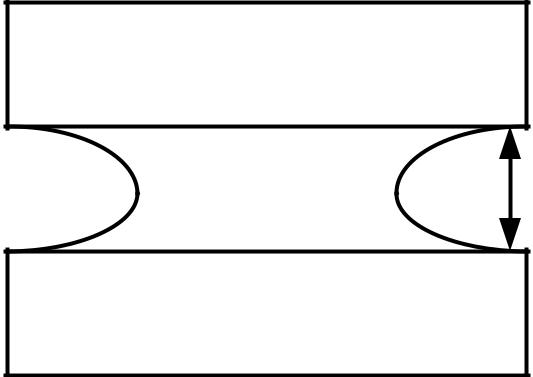
|  |  |  |  |
| --- | --- | --- | --- |
| Шина AGP | Интерфейс | Контроллер |  |
|  |  |
|  | AGP | памяти |  |
| Шина PCI | Контроллер | Контроллер |  |
|  |  |
|  | PCI | IDE |  |
|  |  | Концентратор |  |

Шина памяти

Шина IDE



Основные тенденции развития ЭВМ



* Повышение степени интеграции элементной базы

-Увеличение набора команд -Увеличение степени аппаратной поддержки.

* Обратная совместимость
* Наличие семантического разрыва

|  |  |
| --- | --- |
| Программное обеспечение |  |
| Система | Семантический |
| команд | разрыв |

Аппаратное обеспечение

Проблема семантического разрыва

Технология программирования непрерывно развивается, что позволяет увеличивать функциональность программ и сокращать время их разработки. Создание проблемно-ориентированных языков высокого уровня усугубляет принципиальное отличие языка машинных команд, реализуемого компьютером, от языков, используемых при написании программ. Данная проблема носит название "семантического разрыва" и выражается в неоправданном падении производительности вычислительной системы.

Архитектура системы команд

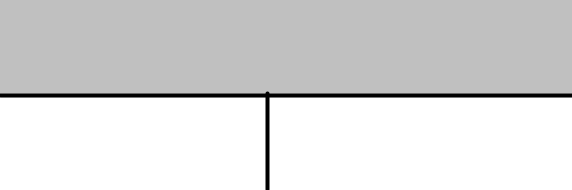
* команде указывается, какую операцию выполнять (КОП), над какими операндами выполнять операцию, а также куда поместить операнд.

Классификация архитектур системы команд



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Аккумуляторная | | |  | Стековая архитектура | |  |
|  | архитектура | | |  |  |
|  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Регистровая архитектура | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

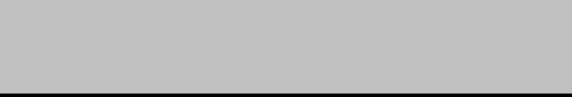
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Архитектура с полным |  | Архитектура с выделенным | |  |  | Архитектура с безоперандным |
| набором команд CISC |  | доступом к памяти | |  |  | набором команд (ROSC) |
|  |  |  |  |  |  |  |



Архитектура с сокращенным

набором команд (RISC)

Архитектура с командами



сверхбольшой длины (VLIW)

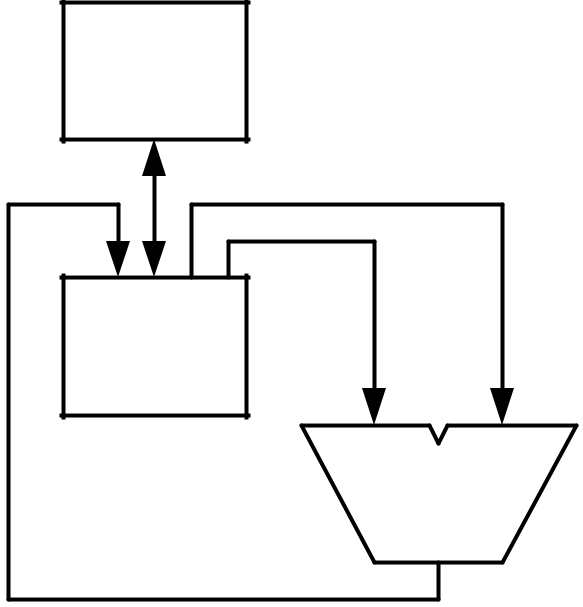
RISC – Reduced Instruction Set Computer; CISC – Complex Instruction Set Computer; VLIW – Very Long Instruction Word; ROSC - Removed Operand Set Computer

Сравнение CISC, RISC и VLIW архитектур СК

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Характеристика** | **CISC** | **RISC** | **VLIW** |  |
|  |  |  |  |  |
| **Длина команды** | Различная | Одинаковая | Одинаковая |  |
|  |  |  |  |  |
| **Расположение** | Различное | Одинаковое | Одинаковое |  |
| **полей в** |  |
| **командах** |  |  |  |  |
|  |  |  |  |  |
| **Количество** | Малое. Регистры | Большое. | Большое. |  |
| специализи- | Регистры | Регистры |  |
| **регистров** | рованные | универсальные | универсальные |  |
|  |  |
|  |  |  |  |  |
|  | Кодируется в | Выполняется по | Выполняется по |  |
|  | команде. |  |
| **Доступ к памяти** | специальной | специальной |  |
| Выполняется по |  |
|  | команде | команде |  |
|  | микрокоманде |  |
|  |  |  |  |
|  |  |  |  |  |
| **Длительность** |  | Одинаковая (для | Различная |  |
| **выполнения** | Различная | большинства |  |
|  |  |
| **команд** |  | команд) |  |  |
|  |  |  |  |  |

Стековая архитектура СК

* При размещении операндов в стековой памяти (LIFO) архитектура команд упрощается (большое количество действий выполняется аппаратно)



ОП

СТЕК

ALU

Операции:

* занесение в стек (PUSH);
* извлечение из стека (POP);
* выполнение действий на стеком

(извлечение операндов из вершины стека, выполнение действий, помещение результата в вершину стека)

Для выполнение арифметических операций их преобразуют к постфиксной форме (Польской записи).

Пример: a = a + b \* (c -d); Постфиксная форма: abcd-\*+;

Действия: PUSH a; PUSH b; PUSH c; PUSH d; SUB; MUL; ADD; POP a.

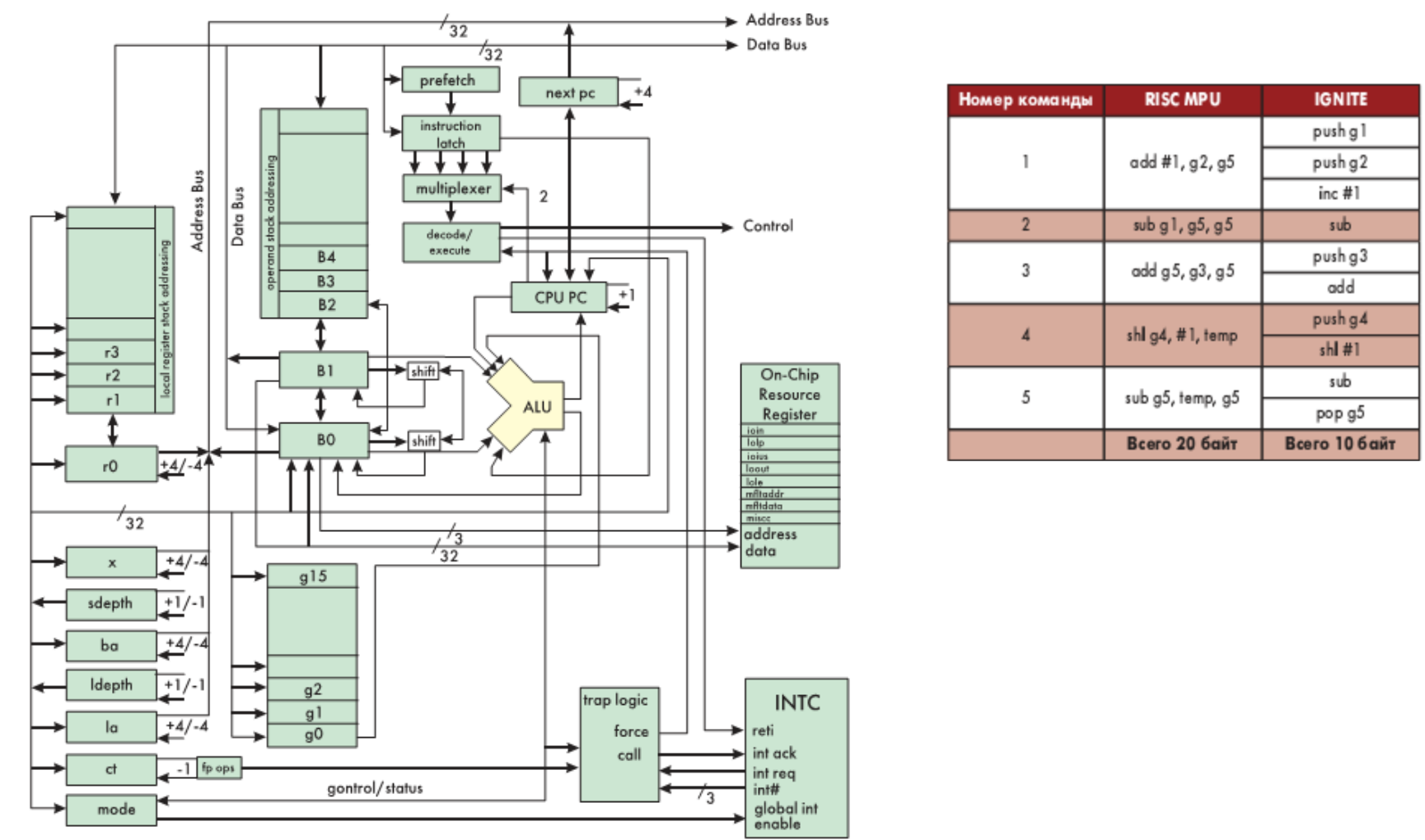
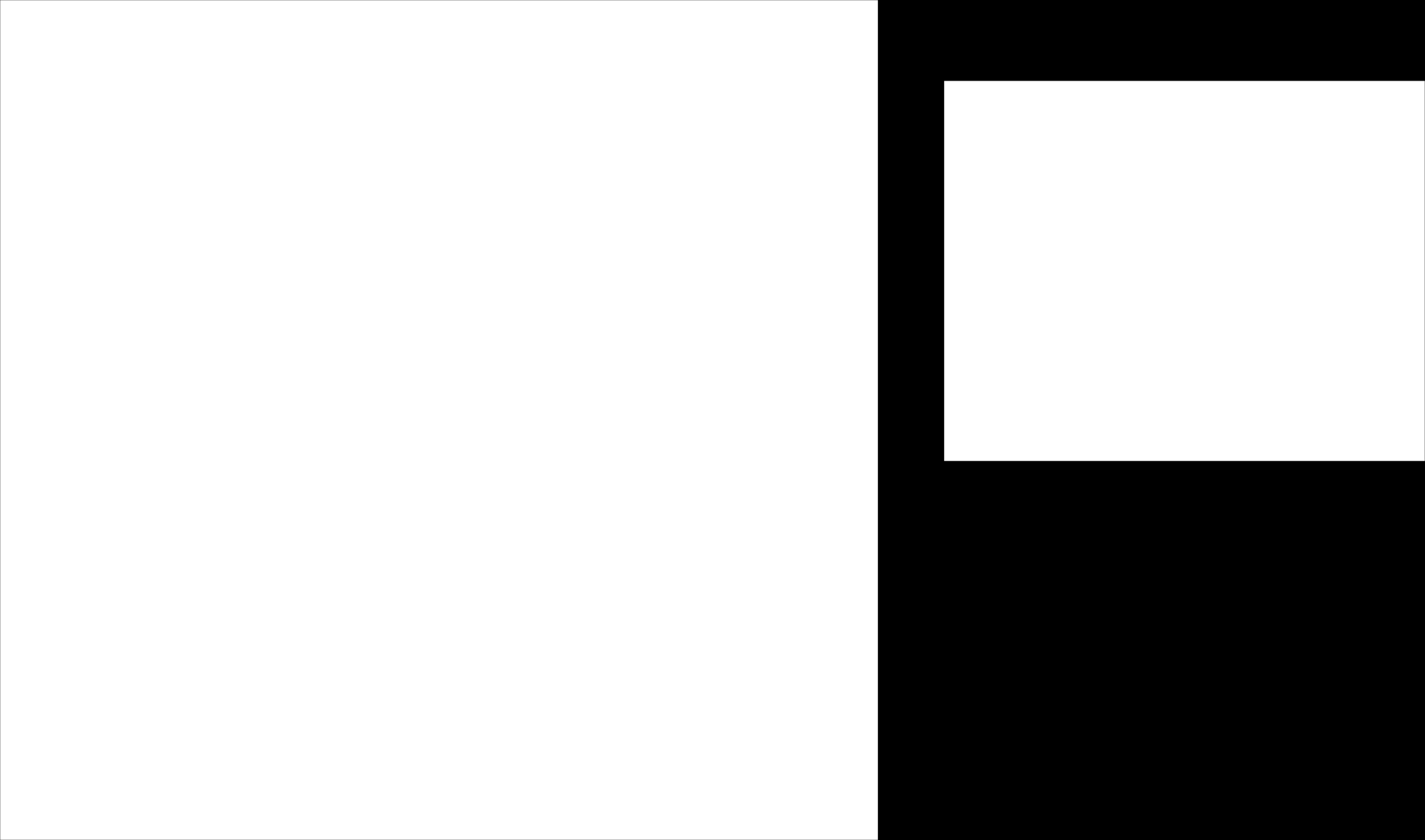
(-) Отсутствие прямого доступа к памяти ограничивает область применения.

(-) Сложность организации параллельной обработки.

Стековые процессоры

(Форт-процессоры)

|  |  |  |
| --- | --- | --- |
| Блок-схема микропроцессора IGNITE | Сравнение выполнения программы на |  |
| RISC-процессоре и на стековом |  |
|  |  |
|  | микропроцессоре |  |



|  |
| --- |
| ПРИМЕР |

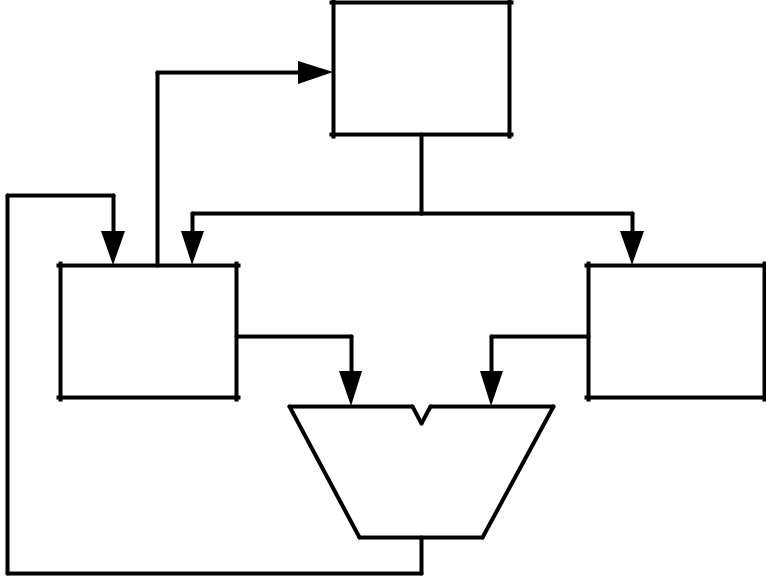
Набор микросхем **TDS9092 FORTHCHIPS**



Использованыматериалыстатьи: ИосифКаршенбойм, Стековые микропроцессоры, или новое - это хорошо забытое новое, Компонентыи технологии, № 9, 2008

Аккумуляторная архитектура СК

Один из операндов должен обязательно находиться в специальном регистре-аккумуляторе. Результат также сохраняется в аккумулятору.



ОП

|  |  |
| --- | --- |
| ACU | TMP |

ALU

Операции:

* занесение в аккумулятор (LOAD);
* извлечение из аккумулятора (STORE);
* выполнение действий над операндами (извлечение первого операнда из аккумулятора, извлечение второго операнда из ОП и помещение во временный теневой регистр TMP, выполнение действий, помещение результата в аккумулятор).

Пример: a = a + b \* (c -d); Определение троек: T1=c-d; T2=b\*T1; T3=a+T2;

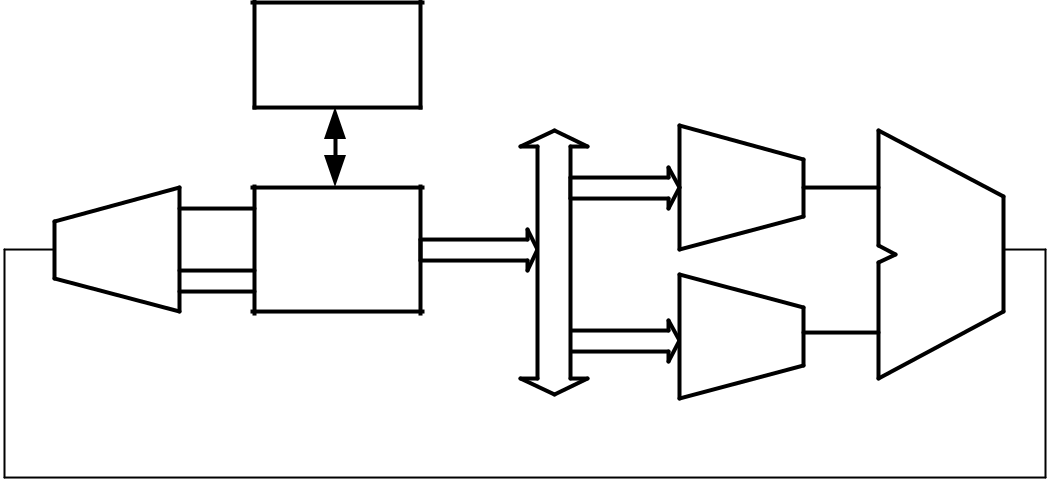
Действия:LOAD c; SUB D; MUL b; ADD a; STORE a.

* В команде необходимо указывать только адрес второго операнда.
* Ускоряются длинные вычисления (a\*b/c+d-e).

(-) Наличие одного аккумулятора является узким местом, т.к. временно ненужный результат необходимо перезаписывать в другой регистр или ОП.

Регистровая архитектура СК

В состав процессора входит большое количество однотипных регистров. В команде необходимо указать номера регистров, хранящих операнды, а также номер регистра операнда.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | ОП |  |  |
|  |  |  | MUX1 |  |
| DMUX | ... | РОН | ALU |  |
|  |  |
|  |  |  |  |
|  |  |  | MUX2 |  |

Для данной архитектуры возможны варианты размещения операндов: оба операнда в памяти; один операнд в памяти и один в РОН; оба операнда в РОН.

Для уменьшения размерности команд и для упрощения декодирования накладывают ограничения на размещение операндов.

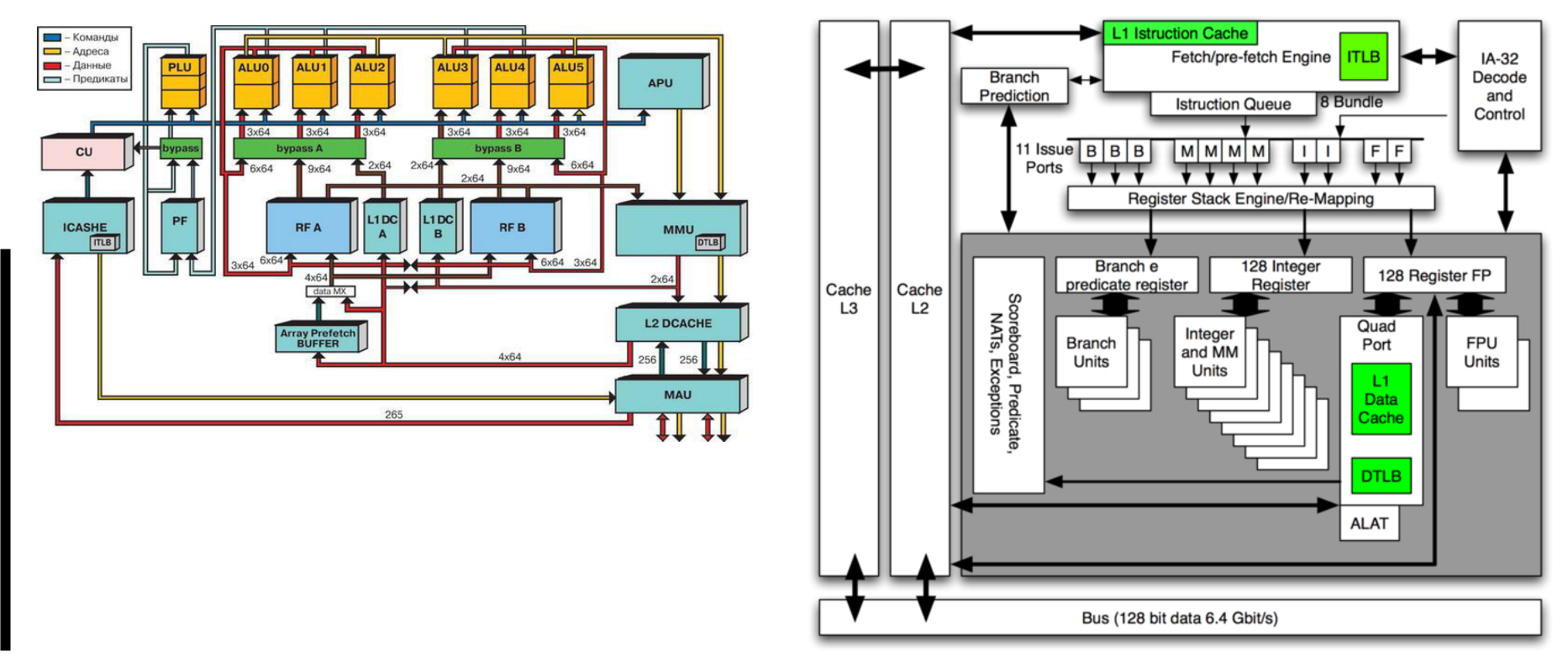
Архитектура VLIW – Very Long Instruction Word

* процессорах VLIW задача распределения решается во время компиляции и в инструкциях явно указано, какое вычислительное устройство должно выполнять какую команду.

Эльбрус-3 и его микропроцессорное исполнение Эльбрус 2000 (E2K) также являютcя VLIW процессорами.

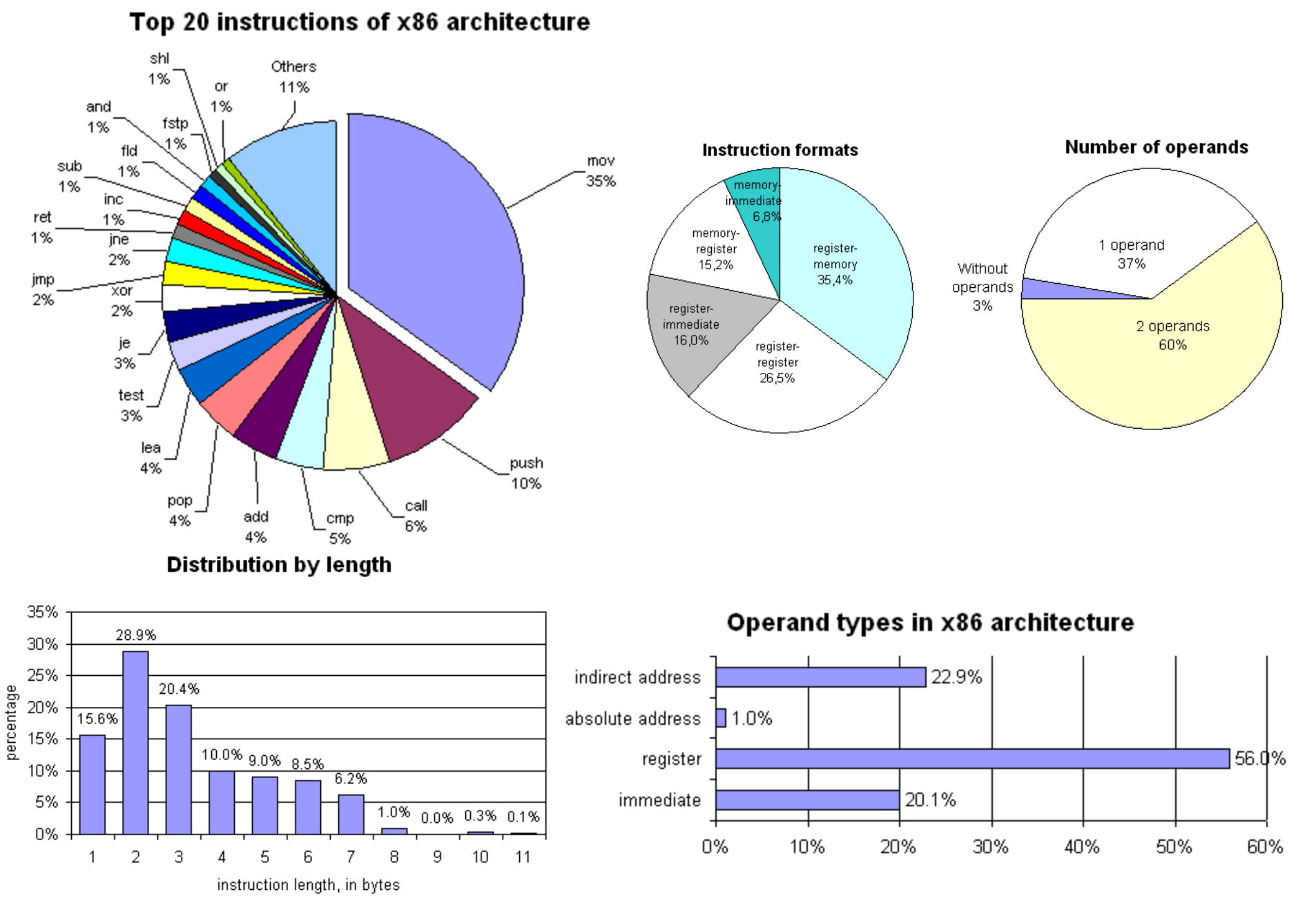
Микропроцессор Intel Itanium имеет как традиционную систему команд IA-32, так и систему команд «с явным параллелизмом» (англ. Explicitly Parallel Instruction Computing, EPIC), исполняемую VLIW-ядром

|  |
| --- |
| ПРИМЕР |



|  |  |  |
| --- | --- | --- |
| **Вариант** | **(+)** | **(-)** |
|  |  |  |
| Оба операнда | Простота аппаратной | Избыточность в команде |
| находятся в | реализации. Простота | из-за сложности |
| регистрах | параллельной обработки. | кодирования с кратностью |
|  |  | 8 бит |
|  |  |  |
| Один операнд | Код компактен. Данные | Наличие адреса в команде |
| находится в | поступают в ALU без | усложняет дешифрацию и |
| регистре, а один в | промежуточного хранения | сокращает возможное кол- |
| памяти | в РОН | во РОН, адресуемых в |
|  |  | команде. |
|  |  |  |
| Оба операнда | Код наиболее компактен. | Команды имеют |
| находятся в | Возможность выполнения | максимальную длину. Из-за |
| памяти | простых действий | наличия коротких и |
|  | наиболее быстро без | длинных команд трудно |
|  | занесения в РОН | оптимизировать тракты |
|  |  | передачи данных и |
|  |  | декодеры инструкций |
|  |  |  |

Статистические данные для x86 команд



Типы команд.

* Команды пересылки данных.
  + регистр-регистр
  + регистр-память
  + память-память
* Команды арифметической и логической обработки (сложение, вычитание, умножение, деление, инкремент, декремент, сравнение, операции над ЧПЗ, логические операции, операции сдвига).

Сдвиг: логический, арифметический, циклический, циклический через дополнительным разряд.

* Команды работы со строками (могут быть реализованы набором других команд, однако удобны при работе с символьной информацией).
* Команды векторной обработки (позволяет выполнять однотипные

действия над большим количеством однородных данных). Пример арифметики с насыщением:

1011 0111 1010

* 0001 1001 1000

1100 1111 1111

* Команды преобразования: служат для табличного преобразования данных из одной системы кодов в другую (2-10 <-> 2)

•Команды ввода/вывода. Служат для управления, проверки состояния и обмена данными с периферийными устройствами.

* + Команды вывода в порт
* Команды ввода из порта.

•Команды управления потоком команд. Данные команды служат для указания очередности выполняемых команд.

Вычисление адреса очередной команды может выполняться несколькими способами:

* увеличением адреса на длину исполненной (естественный порядок).
* изменением адреса на длину следующей (перешагивание)

- изменением адреса на значение, указанное в текущей команде (короткий переход).

- непосредственное указание следующей команды (длинный переход). Перечисленные команды могут выполняться лишь по некоторому условию (уловные переходы).

Команды условного перехода составляют 80% команд управления.

Команды безусловного перехода: вызовы и возвраты из процедур, и.т.д.

Форматы команд.

|  |  |
| --- | --- |
| Операционная часть | Адресная часть |
|  |  |

1. Четырехадресная команда.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| КОП | 1 операнд | 2 операнд | результат | Адр след ком. |
|  |  |  |  |  |

1. Трехадресная команда

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| КОП |  | 1 операнд | 2 операнд | | результат |  |  |
|  |  |  |  |  |  |  |  |
| 3. Двухадресная команда. | | |  |  | Характерна для CISC- | |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| КОП |  | 1 операнд | 2 оп-д/результат | | архитектуры | |  |
|  |  |  |  |
|  | |  |  |  |  |  |  |
| 4. Аккумуляторная архитектура | | |  | Второй операнд хранится в аккумуляторе. | | |  |
|  |  |  |  |  |
| КОП |  | 1 операнд |  | Данный формат команд характерен для RISC- | | |  |
|  |  | архитектур. |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. Нульоперандная команда.

КОП

Способы адресации

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | Адресная часть | | |  |  |  |
|  |  |  | |  | |  | |  |  |  |
|  |  | Способ адресации (СА) | | | | |  | Адрес/операнд |  |  |
|  |  | | |  |  |  |  |  |  |  |
| Непосредственная | | | |  |  |  |  |  |  |  |
| КОП |  |  | СА | Непосредственный операнд | |  |
| адресация | |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

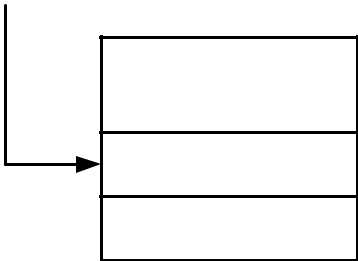
Вместо адреса команда содержит непосредственно операнд.

* команда выполняется быстро

(-) непосредственный операнд может не войти в команду

Прямая адресация

|  |  |  |
| --- | --- | --- |
| КОП | СА | Адрес операнда |
|  |  |  |



ОП

Операнд

Адрес в команде является адресом операнда

* если операнд находится в памяти, то это самый быстрый способ указать на него (-) заранее определенный адрес влияет на переносимость программы.

(-) Адрес занимает много места

|  |  |  |  |
| --- | --- | --- | --- |
| Неявная адресация |  | КОП | СА |
|  |  |  |  |

Операнд подразумевается (следует из КОП).

* Команда занимает мало места

(-) только такие командах нельзя использовать для построение всей системы команд.

Регистровая адресация

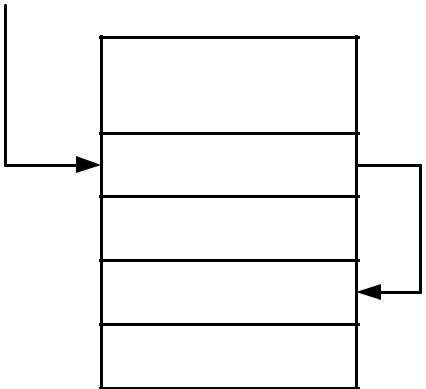
Адрес в команде указывает не на ячейку ОП, а на регистр.

* Быстрее прямой адресации

(-) Количество регистров ограничено

Косвенная адресация

|  |  |  |
| --- | --- | --- |
| КОП | СА | Адрес операнда |
|  |  |  |



Адрес в команде указывает на ячейку памяти, в которой находится адрес операнда.

* удобна для обработки структурных типов данных. (-) приходится осуществлять много обращений к ОП.

ОП

Адрес

Операнд

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Косвенная |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | КОП |  | СА |  |  | Адрес операнда | | | | | |  |
| регистровая | | | |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| адресация | |  |  |  |  |  |  | |  |  |  |  |  |  |  |
|  | |  |  |  | Регистры | |  |  |  |  |  | ОП |  |
| В команде содержится номер регистра, | | | | | |  |  |  |  |  |  |  |  |  |  |
|  |  | Адрес |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| в котором содержится адрес операнда. | | | | | |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | Операнд |  |
| (+) быстрее косвенной адресации | | | | | |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| (-) для перемещения по ОП нужно | | | | | |  |  |  |  |  |  |  |  |  |  |
| менять содержимое регистра | | | | | |  |  |  |  |  |  |  |  |  |  |



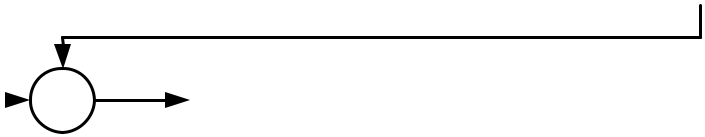
Относительная

адресация

Адрес вычисляется относительно счётчика команд

* Код переносим, команды занимают мало места (-) Может понадобиться длинный адрес

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | ОП |  |
|  |  |  |  |  |  |  |  |  |
| PC | | |  | КОП | СА |  | Адрес |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | |  |
|  |  | + | |  |  | Операнд | |  |
|  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |



Базовая регистровая

адресация

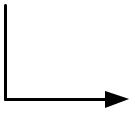
Адрес в команде представляет собой смещение, которое складывается со значением в базовом регистре для получения адреса операнда

* Удобна для работы со структурами данных, размещаемых динамически. (-) Переносимость меньше, чем у относительной адресации

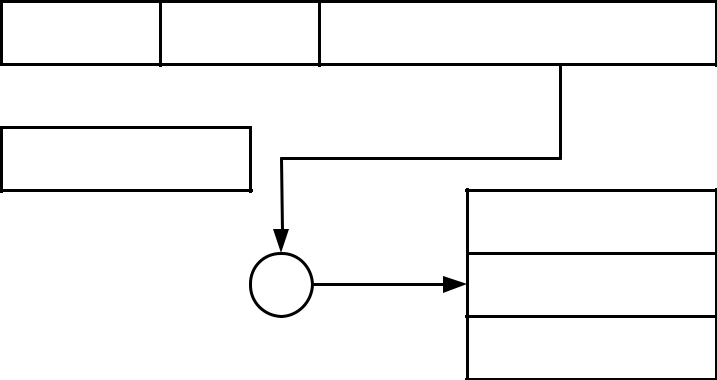
КОП СА

Базовый регистр

Адрес

 +

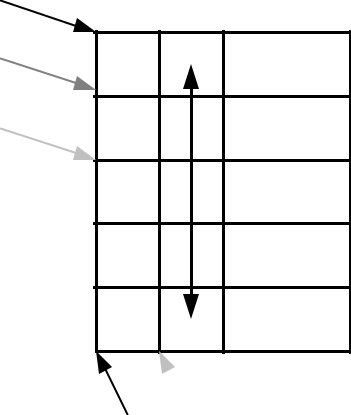
Смещение



ОП

Операнд

База

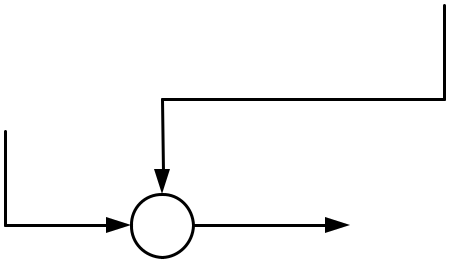


Индексная

регистровая

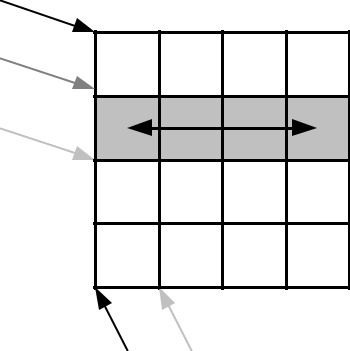
адресация

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | КОП | СА | | Базовый адрес | |
|  |  |  |  |  |  |
| Индексный регистр | | | |  |  |
|  |  | | |  |  |
|  | Смещение | |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  | ОП |
|  |  |  |  |  |  |
|  |  | + | |  | Операнд |
|  |  |  |  |  |  |
|  |  |  |  |  |  |



 Смещение

База



* поле адреса команды содержится базовый адрес, складываемый со значением смещения в индексном регистре.

(+) Удобна для работы со структурами данных, размещаемых динамически.

(-) Переносимость меньше, чем у относительной адресации

Индекс

Автоинкрементная/автодекрементная

адресация

Разновидность регистровой индексной или базовой адресации. До или после выполнения команды значение базового или индексного регистра увеличивается/уменьшается на единицу или масштаб.

* Способ адресации удобен для команд обработки строк.

(-) Автоматическое изменение часто требуется выполнять на величину, большую единицы.

Индексная адресация с

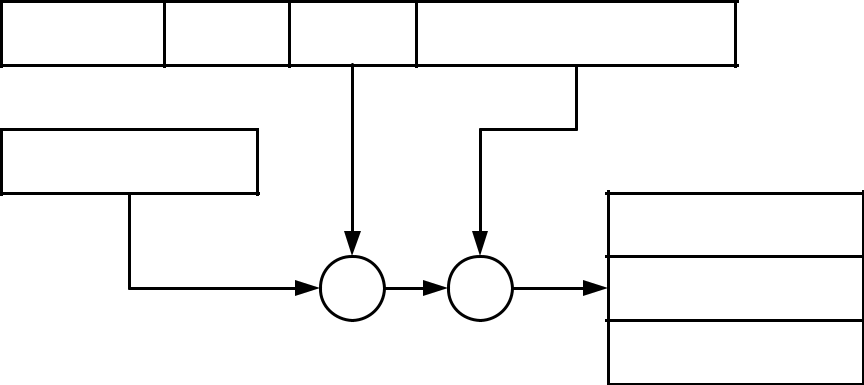
масштабированием

Индексный регистр умножается на масштаб M и суммируется с базовым адресом из команды.

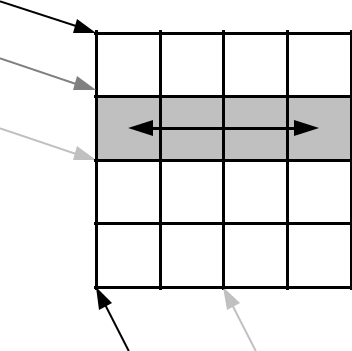
* Удобен для модификации адреса на величину M.

(-) Вычисление адреса замедляется, т.к. требуется выполнять умножение.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| КОП | СА | M | Базовый адрес | |
| Индексный регистр | |  |  |  |
| Смещение | |  |  |  |
|  |  |  |  | ОП |
|  |  | x | + | Операнд |



База



Индекс

(М=2)

Базовая индексная адресация с

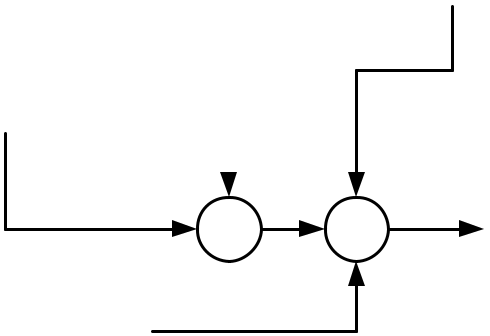
масштабированием

Адрес определяется по формуле Адрес=Индекс\*Масштаб+База+Смещение.

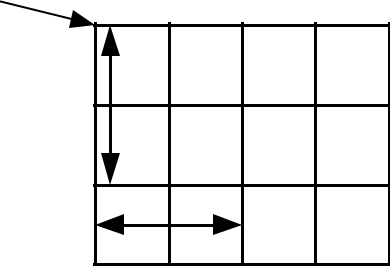
* Базовая индексная адресация с масштабированием часто используется при обращении к системным таблицам, находящимся в ОП (таблица дескрипторов, таблицы страниц, таблица векторов прерываний и т.д.)

(-) Ограниченное на величину M (M=1,2,4,8).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | КОП | СА | | M | | Смещение | |  |  |
|  |  |  |  |  |  |  |  |  |  |
| Индексный регистр | | | |  |  |  |  |  |  |
|  |  | | |  |  |  |  |  |  |
|  | Столбец | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | ОП | |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | x | | + | Операнд | |  |
|  | Базовый регистр | | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  | | |  |  |  |  |  |  |
|  | Строка(база) | |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |



Смещение



База

Индекс\*M