Организация ЭВМ и ВС
Лекции 6–8. Центральный процессор

Макаров П. А.

10, 24 ноября и 8 декабря 2026 г.

Содержание

1 Общее представление о регистрах и системе команд 2
1.1 Регистры процессора . . . 3
1.2 Система команд . . . 5
1.3 Классификация архитектур систем команд по типам памяти . . . 9
1.4 Порядок байт и выравнивание информации . . . 10
1.5 Режимы адресации . . . 11

2 Архитектура и микроархитектура процессора 12

3 Основные классы архитектур процессоров 13
3.1 CISC . . . 13
3.2 RISC . . . 14
3.3 MIPS . . . 15
3.4 VLIW . . . 15
3.5 MISC . . . 16
3.6 EPIC . . . 17
3.7 Комбинированные системы команд . . . 17

4 Процессоры семейства x86 18
4.1 Предшественники x86 — первые модели восьмиразрядных микропроцессоров . . . 18
4.1.1 Intel 8008 . . . 18
4.1.2 Zilog Z80 . . . 18
4.1.3 Motorola MC6800 . . . 19
4.1.4 Intel 8080 . . . 19
4.2 Первое поколение x86 . . . 21

4.2.1 Intel 8086 . . . 21
4.2.2 Intel 8088 . . . 22
4.3 Архитектура и микроархитектура базового процессора x86 . . . 22
4.4 Архитектура IA-32 (x86-32) . . . 26

5 Технологии повышения производительности процессоров 28
5.1 Операции над вещественными числами (с плавающей точкой) . . . 28
5.2 Конвейерная обработка команд . . . 29
5.2.1 Конфликты в конвейерах . . . 31
5.2.2 Суперскалярные процессоры . . . 32
5.3 Увеличение разрядности систем . . . 34
5.3.1 IA-64 . . . 35
5.3.2 AMD64 . . . 35
5.3.3 EM64T или Intel 64 . . . 35
5.4 Векторная обработка (SIMD-команды) . . . 36
5.4.1 MMX . . . 36
5.4.2 3DNow! . . . 38
5.4.3 SSE . . . 39
5.4.4 SSE2 . . . 40
5.4.5 SSE3 . . . 40
5.5 Динамическое исполнение . . . 41
5.5.1 Предсказание ветвлений . . . 41
5.5.2 Внеочередное выполнение . . . 41
5.5.3 Переименование (ротация) регистров (register rename) . . . 41
5.5.4 Выполнение по предположению . . . 41
5.6 Многократное декодирование команд . . . 44
5.6.1 Декодирование команд CISC/RISC в VLIW . . . 45
5.6.2 Декодирование команд CISC/VLIW в RISC . . . 45
5.6.3 Макрослияние (macrofusion) . . . 46
5.6.4 Микрослияние (micro-op fusion) . . . 46
5.7 Технология Hyper-Threading (HT) . . . 46
5.8 Многоядерные процессоры . . . 47

1 Общее представление о регистрах и системе команд

В предыдущих лекциях было сказано, что центральное устройство компьютера образуют процессор и оперативная память (см. рис. 1). Сейчас сосредоточимся на более подробном изучении процессора ВМ.



Рис. 1: Структура простейшего центрального устройства ВМ

Определение 1. ***Процессором*** *является функционально полная совокупность устройств, которая регулирует, управляет и контролирует рабочий процесс компьютера (процесс обработки данных).*

1.1 Регистры процессора

Регистры — устройства, предназначенные для временного хранения данных ограниченного размера. Иногда говорят о РЗУ — регистровом запоминающем устройстве. Важным свойством регистра является высокая скорость записи и чтения данных. Регистр состоит из двоичных разрядов, в которых хранится то или иное слово: машинная инструкция, непосредственный операнд и т. д. Обычно регистр имеет ту же разрядность, что и машинное слово.

Регистр, обладающий способностью перемещать содержимое своих разрядов, называют сдвиговым. В этих регистрах за один такт хранимое слово поразрядно сдвигается на одну позицию.

Рассмотрим основные типы регистров по их функциям.

- Регистры общего назначения (GPR — General Purpose Registers) или регистры сверхоперативной памяти (либо регистровый файл) — это общее название для регистров, которые временно содержат данные, передаваемые в память или принимаемые из неё.

- Регистр команды (IR — Instruction Register) служит для размещения машинной команды, которая находится в нём в течение текущего цикла процессора.

- Регистр или счётчик адреса команды (IP — Instruction Pointer) — регистр, содержащий адрес текущей команды.

- Регистр адреса — содержит адрес одного из операндов выполняемой команды (регистров может быть несколько).
- Регистр числа — содержит операнд выполняемой команды (этих регистров также несколько).
- Регистр результата предназначается для хранения результата выполнения команды.
- Сумматор — регистр, осуществляющий операции сложения (логического и арифметического двоичного) чисел или битовых строк, представленных в прямом или обратном коде. Регистр, хранящий промежуточные данные, часто именуют аккумулятором.
- Регистр состояния (SR — Status Register) или регистр флагов (FLAGS). Его содержимым является информация об особых результатах завершения команды (ноль, переполнение, деление на ноль, перенос и пр.). Устройство управления (УУ) использует информацию из SR для исполнения условных переходов.

Цикл выполнения короткой команды может выглядеть следующим образом.

1. В соответствии с содержимым IP УУ извлекает из оперативной памяти очередную команду и помещает её в IR.
2. Осуществляется расшифровка (декодирование) команды.
3. Адреса A1, A2 и пр. помещаются в регистры адреса.
4. Если в инструкции указаны индексный или базовый регистры, то их содержимое используется для модификации регистра адреса — фактически выбираются данные, смещённые в ту или иную сторону по отношению к адресу, указанному в команде.
5. По значениям регистра адреса осуществляется чтение операндов и помещение их в регистр числа.
6. Непосредственное выполнение операции и помещение результата в соответствующий регистр.
7. Запись результата по одному из адресов (если необходимо).
8. Увеличение содержимого IP (переход к следующей команде).

За счёт увеличения числа регистров возможно распараллеливание, перекрытие операций. Например, при считывании команды можно автоматически увеличить IP, подготовив выборку следующей команды. После расшифровки текущей команды IR

освобождается и в него может быть прочитана следующая команда. При выполнении операции возможна расшифровка следующей команды и т. д. Всё это является предпосылкой построения конвейерных структур (pipeline), подробности которых представлены в п. 5.2. Однако всё это хорошо только при последовательном (естественном) порядке выполнения команд. Появление переходов (особенно по условию, не определенному ранее) нарушает эту картину. Поэтому современные процессоры пытаются предсказывать переходы в программе (branch prediction).

1.2 Система команд

Определение 2. *Полный набор всех инструкций, которые может выполнить процессор называется* ***системой команд.***

Центральный процессор в зависимости от типа, семейства, степпинга обладает тем или иным набором машинных команд, который он может выполнять. Команды современных процессоров можно разделить на три типа: привилегированные, чувствительные и обычные. Первые выполняются только в привилегированном режиме на системном уровне, так как они отвечают за формирование системных таблиц и активизацию системных ресурсов. Чувствительные команды могут выполняться на пользовательском уровне, но при определённых условиях. Остальные команды могут выполняться на любом уровне.

Машинные инструкции классифицируются:

- по функциям (выполняемым операциям),
- направлению приёма-передачи информации,
- адресности.

Команды процессора представляются двоичными числами и, в принципе, не отличимы от данных.

Определение 3. ***Структурой команды процессора*** *называется её логическое разделение на смысловые поля. К основным компонентам структуры команды относят:*

- *код операции (opcode);*
- *управляющая информация (может быть частью кода операции);*
- *операнды: регистры, адреса, непосредственные значения.*

Определение 4. ***Форматом команды процессора*** *называется представление структуры команды в виде битовой последовательности (машинного слова). Формат определяет:*

- *разрядность команды (16, 32, 64 бита);*
- *размер и расположение каждого поля внутри машинного слова;*

Если все команды допускают все способы адресации, то такая система команд называется симметричной.

Основные признаки классификации и типы команд процессоров.

- **Команды пересылки.** Пересылка данных между двумя регистрами или между регистром и ячейкой памяти. В некоторых процессорах реализуется пересылка между двумя ячейками памяти, а также групповая пересылка содержимого нескольких регистров в память или их загрузка из памяти.
- **Команды ввода/вывода.** Реализуют пересылку данных из регистра процессора (или ячейки памяти) во внешнее устройство или приём данных из внешнего устройства в регистр (ячейку памяти). Инструкции ввода/вывода могут отсутствовать, если переферийные устройства отображены на память, когда их адреса и адреса ячеек памяти составляют единое адресное пространство. Однако в процессорах обычно реализуются как адресное пространство памяти, так и адресное пространство ввода/вывода.
- **Команды обработки данных.**
  - Короткие операции (один такт):
    - Логические (`not`, `and`, `or`, `xor`, `test`, ...);
    - Арифметические (`neg`, `add`, `inc`, `sub`, `dec`, `mul`, `div`, `cmp`, ...);
    - Команды сдвига (`shr`, `ror`, `rcr`, ...).
  - Длинные операции (несколько тактов):
    - Сложение/вычитание с плавающей точкой;
    - Умножение/деление с фиксированной и плавающей точкой.
- **Операции управления.**
  - Безусловный переход (ветвление): `jmp`.
  - Вызов и возврат из подпрограмм: `call`, `ret`.
  - Условные переходы: `je`, `jne`, `jg`, `jl`, ...
  - Команды организации программных циклов: `loop`, ...
  - Команды прерывания и системные вызовы: `int`, `syscall`, ...
  - Команды изменения признаков (запись-чтение содержимого регистра состояния, в котором хранятся признаки, а также изменение значений отдельных признаков).

  - Команды управления процессором (команды останова `hlt`, отсутствия операции `nop` и ряд команд, определяющих режим работы процессора или его отдельных блоков).

- Тип выборки и пересылок данных:
  - регистр — регистр;
  - регистр — память;
  - память — память.

- Адресация
  - **Непосредственная.** Операнд непосредственно содержится в поступившей команде, размещаясь следом за кодом операции.
  - **Регистровая.** Операнд выбирается из регистра, номер (имя) которого указано в команде.
  - **Прямая.** Операнд выбирается из ячейки/слова памяти, адрес которой содержится в команде.
  - **Относительная.** Операнд выбирается из ячейки памяти, адрес которой является суммой текущего содержимого `IP` и заданного в команде смещения (числа со знаком). Во многих процессорах этот способ адресации используется не для адресации операнда, а для формирования адреса, к которому переходит программа при ветвлении. При этом сформированный таким образом адрес загружается в `IP`, обеспечивая выборку требуемой следующей команды.
  - **Косвенно-регистровая.** Операнд выбирается из ячейки/слова памяти, адрес которой содержится в регистре, указанном в команде.
  - **Косвенно-регистровая со смещением.** Операнд выбирается из ячейки памяти, адрес которой является суммой содержимого, указанного в команде регистра и заданного в команде смещения (смещение может быть положительным или отрицательным числом).
  - **Косвенно-регистровая с индексированием и смещением.** Операнд выбирается из ячейки памяти, адрес которой является суммой содержимого указанного в команде регистра, индексного регистра и заданного в команде смещения. Иногда имеются специальные индексные регистры, иногда в качестве индексного используется регистр, номер или имя которого указывается в команде. Частным случаем этого способа является индексная адресация, когда адрес образуется суммированием специального индексного регистра и заданного в команде смещения.

- Адресность

- **Одноадресные.** `OpCODE A1` A1 в зависимости от модификации команды может обозначать либо адрес ячейки (регистра), в которой хранится одно из чисел, участвующих в операции, либо адрес ячейки (регистра), куда следует поместить результат.
- **Двухадресные.** `OpCODE A1 A2` A1 — это обычно адрес ячейки (регистра), где хранится первое из чисел, участвующих в операции, и куда после завершения операции должен быть записан результат; A2 — обычно адрес ячейки (регистра), где хранится второе участвующее в операции число.
- **Трёхадресные.** `OpCODE A1 A2 A3` A2 и A3 — адреса ячеек (регистров), где расположены, соответственно, первый и второй операнды, а A1 — адрес ячейки (регистра), куда следует поместить результат выполнения операции.
- **Безадресные.** `OpCODE data` Содержит только код операции, а информация для неё должна быть заранее помещена в определенные регистры машины или содержаться в адресной части.
- **Комбинированые.** `OpCODE addr data` Один или более адресов адресной части предназначены для размещения данных (непосредственных операндов).

- Другие признаки
  - **Операции с фиксированной точкой.** Арифметические и логические операции над числами, обычно занимающими одно машинное слово (пример — системы команд процессоров i8086 – i80486).
  - **Операции с плавающей точкой.** Арифметические операции над числами, представленными в виде «мантисса — экспонента» (пример — операции сопроцессора i80487).
  - **Десятичная арифметика.**
  - **Символьная обработка.** Команды обработки байт памяти как символов ASCII (сравнение, сортировка и пр.)
  - **Обработка чисел большой длины.**
  - **Векторные операции.** Команды SIMD (Single Instruction — Multiple Data) включают MMX, 3DNow!, SSE, SSE2, SSE3, SSSE3.
  - **Команды индексной арифметики.** Изменение содержания индексных регистров (в некоторых машинах — ячеек оперативной памяти), что используется для обращения к последовательным элементам массива.

Очевидна связь таких параметров центрального устройства, как длина адресного пространства, адресность, разрядность. Увеличение разрядности позволяет увеличить адресность команды и длину адреса (т. е. объём памяти, доступный данной команде; например, в 32-х разрядной машине, можно адресовать до 4 ГБайт оперативной памяти). Увеличение адресности, в свою очередь, приводит к повышению быстродействия обработки (за счёт снижения числа требуемых команд).

В трехадресной машине, например, сложение двух чисел требует одной команды (извлечь число по `A2`, число по `A3`, сложить и записать результат по `A1`). В двухадресной машине необходимы две команды (первая — извлечь число по `A1` и поместить в регистр числа (или сумматор), вторая — извлечь число по `A1`, сложить с содержимым регистра числа и результат записать по `A2`). Легко видеть, что одноадресная машина потребует три команды. Поэтому неудивительно, что основная тенденция в развитии центрального процессора состоит в увеличении разрядности.

Иногда используют и четырёхадресные команды, в которых последний адрес определяет адрес следующей команды.

1.3 Классификация архитектур систем команд по типам памяти

Типы организации внутренней памяти процессора определяют различия в архитектуре систем команд. Различают стековую, аккумуляторную, регистровую (Load — Store), регистр — память и память — память архитектуры. При стековой организации внутренней памяти операнды расположены на верхушке стека и задаются неявно. В акумуляторной архитектуре один операнд задаётся неявно. В архитектуре с общими регистрами операнды задаются только явно указанием номера регистра или ячейки памяти.

Программы выполнения операции $C = A + B$ для разных архитектур выглядят по-разному.

- Стековая архитектура[1]:

```
push A
push B
add
pop C
```

- Аккумуляторная архитектура:

```
load A
add B
store C
```

[1] См. обратная польская нотация.

- Архитектура регистр — регистр:

```
load R1, A
load R2, B
add R3, R1, R2
store R3, C
```

- Архитектура регистр — память:

```
load R1, A
add R2, R1, B
store R2, C
```

Явные операнды могут прямо читаться из памяти или предварительно загружаться в промежуточную память, что зависит от типа архитектуры и выбора конкретных команд.

Из примера с программированием операции сложения видно, что компьютеры с регистрами принадлежат к двум классам. В архитектуре регистр — память любая команда имеет доступ к памяти. В архитектуре регистр — регистр доступ к памяти возможен лишь с помощью команд `load` (загрузка регистра) и `store` (запись регистра в память). Поэтому такая архитектура называется также Load — Store. Компьютеры с архитектурой память — память сегодня не распространены.

Во многих компьютерах, раработанных после 1980 года, используется регистровая архитектура Load — Store. Этому способствовали две причины: быстрый доступ к регистрам и большая эффективность для компиляторов. Например, выражение $A \cdot B - B \cdot C + A \cdot D$ в регистровой архитектуре может вычисляться в любом порядке, а в стековой машине — только в одном, поскольку операнды размещены в стеке. Если переменные располагаются в регистрах, то трафик с памятью уменьшается, и скорость выполнения программы возрастает, так же как и плотность кода из-за укороченного адреса регистра. Разработчики компиляторов предпочитают, чтобы все регистры были эквивалентны.

Архитектурой регистр — регистр обладают процессоры Alpha, ARM, MIPS, PowerPC, SPARC, Super-H, Trimedia TMS200. Архитектура регистр — память характерна для процессоров Intel 80x86, Motorola 68000.

1.4 Порядок байт и выравнивание информации

Машинные команды обеспечивают доступ к ячейкам памяти разной длины: байтам, словам, двойным словам, учетверённым словам (64 бита). Данные в памяти могут располагаться, начиная с младшего бита в младшем разряде (Little Endian) или со старшего бита в старшем разряде (Big Endian). Порядок байт представляет собой проблему при обмене данными между машинами с разным упорядочением бит.

Данные в памяти могут располагаться (byte order):

- от младшего к старшему — Little Endian (PDP-11, VAX, x86, в том числе и x86-64),
- от старшего к младшему — Big Endian (SPARC, Motorola 68k, IBM z/Architecture, TCP/IP),
- в переключаемом порядке — Bi-Endian (PowerPC, ARM, RISC-V, IA-64),
- в смешанном порядке.

Как правило, информация в памяти должна быть выравнена (Aligned), т. е. слово должно иметь чётный адрес, двойное слово — кратный четырём и т. д. В некоторых процессорах допускается невыравненное расположение информации, что сопровождается её чтением за большее число тактов, а аппаратная составляющая усложняется.

1.5 Режимы адресации

Режимы адресации определяют методы доступа к памяти для получения операндов и для записи результатов. Действительный адрес ячейки памяти, вычисленный в соответствии с режимом адресации, называется эффективным (или исполнительным) адресом. Режимы адресации, зависящие от IP, называются относительными.

Применение различных режимов адресации способно значительно существенно сократить количество команд в программе. Однако большое число возможных режимов адресации усложняет строение компьютера и может увеличивать среднее число тактов на команду. Поэтому для разработчика архитектуры важно, какие режимы адресации выбрать.

Для наглядности и простоты изложения на рис. 2 приведена структура простейшей одноадресной команды.



Рис. 2: Структура одноадресной команды. OpCODE — поле кода операции, AddMode — поле режима адресации, Address — поле адреса

Для многоадресных команд поля AddMode и Address повторяются соответствующее число раз.

Чаще всего используются следующие режимы адресации.

- Регистровая — в поле Address указывается номер регистра, содержащего операнд.
- Абсолютная — в поле Address указывается полный адрес операнда в памяти. Используется для доступа к неперемещаемым данным;

- Страничная;
- Относительная — в поле Address указывается смещение со знаком в старшем разряде, которое алгебраически суммируется с содержимым IP. Для реализации относительной адресации требуется полноразрядный сумматор. Программы, написанные только с использованием относительной адресации, не зависят от расположения в памяти.
- Индексная — является расширением относительной, при которой адрес операнда получается суммированием смещения (находящегося в поле Address) и содержимого выделенного индексного регистра. Этот способ адресации позволяет организовать доступ ко всему объёму памяти и полезен при обработке массивов — при этом доступ к его последовательным ячейкам организуется изменением в цикле содержимого индексного регистра.
- Базово-индексная — эффективный адрес получается суммированием смещения, индексного и базового регистров, что позволяет с помощью одной команды получить доступ к элементам двумерного массива или для перехода между элементами двух массивов.
- Автоинкрементная — к содержимому регистра или указателю стека прибавляется инкрементирующее значение (1, 2, 4) и полученное число используется как адрес.
- Автодекрементная — сначала содержимое регистра или указателя стека используется как адрес, затем из него вычитается декрементирующее значение.

Все перечисленные выше способы адресации являются прямыми — вычисленный адрес является исполнительным, т. е. адресом операнда или результата.

При косвенной адресации вычисленный адрес интерпретируется как указатель адреса, т. е. как адрес адреса операнда. Косвенная адресация позволяет модифицировать адрес, не затрагивая адресную часть команды. Создав предварительно некоторую таблицу адресов и используя эффективный адрес для входа в неё, можно обеспечить доступ к произвольно организованным структурам данных. Иногда используют многократную косвенную адресацию.

Особняком стоит непосредственная адресация, когда в поле адреса задаётся константа (слагаемое, количество сдвигов, число для сравнения и пр.). Такие команды выполняются быстрее, экономят пространство памяти и уменьшают количество обращений к ней.

2 Архитектура и микроархитектура процессора

Определение 5. *Под* ***архитектурой центрального процессора*** *понимается его программная модель, то есть основные программно-видимые свойства. К*

ним относят: набор регистров, систему команд и механизм обработки прерываний.

Пример 1. *Процессоры семейства x86 — представители CISC-архитектуры, т. к. по сложности системы команд им нет равных, при этом базовых архитектурных регистров мало.*

По мере развития компьютеров, в систему команд процессоров включаются всё более и более мощные инструкции, позволяющие сократить их общее число, требуемое для решения одних и тех же прикладных задач, однако эти команды становится всё сложнее исполнять.

Определение 6. ***Микроархитектура*** *— это конкретная внутренняя реализация данной программной модели.*

Для одной и той же архитектуры, например, IA-32, применяются существенно различающиеся микроархитектурные реализации. При переходе от одной микроархитектуры к другой, как правило, стремятся к повышению производительности.

Начиная с процессоров шестого поколения семейства x86 (Intel Pentium Pro и AMD K5) в микроархитектуре центрального процессора применяется RISC-ядро, исполняющее микрооперации, на которые раскладываются сложные инструкции x86. В то же время, существуют и компьютеры на "чистых" RISC-процессорах.

Пример 2. *Компьютеры Power Macintosh фирмы Apple основаны на процессорах архитектуры PowerPC (PPC), разработанной в 1991 году альянсом компаний Apple, IBM и Motorola (известным как AIM). Power Mac обеспечивает ту же прикладную производительность, что и IBM PC, но на более низких тактовых частотах центрального процессора.*

3 Основные классы архитектур процессоров

В зависимости от набора и порядка выполнения команд процессоры подразделяются на несколько классов.

3.1 CISC

CISC (Complex Instruction Set Computer) — классическая архитектура процессоров, которая начала своё развитие в 1940-х гг. с появлением первых компьютеров, и в которой ЦП использует микропрограммы для выполнения большого набора разноформатных команд с использованием многочисленных способов адресации. Для этого требуется наличие сложных электронных цепей для декодирования и исполнения. В течение длительного периода производители компьютеров разрабатывали и воплощали в изделиях всё более сложные и полные системы команд.

Типичным примером CISC являются процессоры Intel x86 (в частности, семейство Pentium). Они выполняют более 200 команд разной степени сложности, которые имеют размер от 1 до 15 байт, и обеспечивают более десяти различных способов адресации. Такое многообразие выполняемых команд и способов адресации позволяет программисту реализовать наиболее эффективные алгоритмы решения различных задач. Однако при этом существенно усложняется структура процессора, особенно его устройства управления, что приводит к увеличению размеров и стоимости кристалла, снижению производительности.

Архитектура CISC характеризуется так же малым числом регистров процессора, что порождает конфликты по данным при конвейеризации вычислений. Большое число команд усложняет УУ, а их сложность предопределяет их выполнение за большое число тактов, что в целом снижает производительность машины.

В то же время анализ работы процессоров показал, что в течение примерно 80% времени выполняется лишь 20% общего набора команд. Поэтому была поставлена задача оптимизации выполнения небольшого количества часто используемых команд. В середине 70-х это привело многих производителей компьютеров к пересмотру своих позиций и к разработке ЦП с ограниченным набором команд.

3.2 RISC

Архитектура RISC (Redused Instuction Set Computer) отличается использованием ограниченного набора команд фиксированного формата. Первый процессор RISC был создан корпорацией IBM в 1979 г. и имел шифр IBM 801. Современные RISC-процессоры обычно реализуют около 100 команд, имеющих фиксированный формат длиной 4 байта. Также значительно сокращается число используемых способов адресации. Обычно в RISC-процессорах все команды обработки данных выполняются только с регистровой или непосредственной адресацией. При этом для сокращения количества обращений к памяти RISC-процессоры имеют увеличенный объём внутренней регистровой памяти — от 32 до нескольких сотен регистров (в CISC-процессорах число регистров общего назначения обычно составляет 8 – 16). В результате процессор на 20 – 30% реже обращается к оперативной памяти, что также повышает скорость обработки данных. Упростилась топология процессора, выполняемого в виде одной интегральной схемы, сократились сроки её разработки, она стала дешевле.

Обращение к памяти в RISC-процессорах используется только в операциях загрузки данных в регистры или пересылки результатов из регистров в память. При этом используется небольшое число наиболее простых способов адресации — косвенно-регистровая, индексная и некоторые другие. В результате существенно упрощается структура процессора, сокращаются его размеры и стоимость, значительно повышается производительность. Начиная с процессора Pentium, корпорация Intel начала внедрять элементы RISC-технологий в свои устройства.

В то время, как в процессоре CISC для выполнения одной команды необходимо в большинстве случаев десять тактов и более, процессоры RISC близки к тому, чтобы выполнять по одной команде в каждом такте. Следует также иметь в виду, что благодаря своей простоте процессоры RISC не патентуются (в то же время широко распространена практика лизензирования процессоров этого класса архитектур). Это также способствует их быстрой разработке и широкому производству. Можно упомянуть, например, широко известную архитектуру этого типа ARM — Advanced RISC Machine, лицензиатами которой являются множество известных компаний (AMD, Apple, Analog Devices, Atmel, Xilinx, Cirrus Logic, Intel, Marvell, NXP, STMicroelectronics, Samsung, LG, MediaTek, Qualcomm, Sony, Texas Instruments, Nvidia, Freescale, Миландр, HiSilicon, Байкал электроникс).

К недостаткам RISC-архитектуры относится увеличение длины программы в среднем на 30% по сравнению с CISC-архитектурой, отсутствие прямой адресации памяти, замедленный доступ к регистрам из-за сложности дешифратора их номеров, сложность отладки и устранения ошибок в аппаратно реализованном УУ, поскольку в случае ошибки требуется его переконструрование. Правда, современные средства автоматизированного проектирования и моделирования позволяют устранить ошибки до реализации устройства “в железе”.

3.3 MIPS

MIPS (Microprocessor without Interlocked Pipeline Stages — без блокировок в конвейере) — система команд и микропроцессорных архитектур, разработанных компанией MIPS Computer Systems (в 2018 году приобретена корпорацией Wave Computing) в соответствии с концепцией проектирования процессоров RISC (то есть для процессоров с упрощенным набором команд). Ранние модели процессора имели 32-битное машинное слово, позднее появились его 64-битные версии. Существует множество модификаций процессора, включая MIPS I – V, MIPS 32 и MIPS 64, из них действующими являются последние две.

Различные реализации MIPS используются в основном во встраеваемых системах, например, в смартфонах, маршрутизаторах, шлюзах, а также до 2010-х годов широко применялись в игровых консолях.

3.4 VLIW

VLIW (Very Large Instruction Word) — архитектура, которая появилась относительно недавно (в 1990-х гг.). Её особенностью является использование очень длинных команд (256 бит и более), отдельные поля которых содержат коды, обеспечивающие выполнение различных операций.

Архитектура VLIW является развитием RISC-архитектуры применительно к суперконвейерным структурам с управлением выполнением команд на основе потока

данных, т. е. не в порядке их расположения в программе, а по мере готовности операндов и освобождения соответствующих операционных блоков. В обычных процессорах (не VLIW) готовые к выполнению команды выбираются из буфера ограниченного размера, поэтому дистанция просмотра программы оказывается недостаточной для оптимальной загрузки исполнительных устройств.

Специальный компилятор планирования VLIW перед выполнением прикладной программы проводит её анализ и по множеству ветвей последовательности операций определяет группу команд, которые могут выполняться параллельно. Каждая такая группа образует одну сверхдлинную команду. Это позволяет решать две важные задачи. Во-первых, в течение одного такта выполнять группу коротких («обычных») команд, а во-вторых — упростить структуру процессора. Этим технология VLIW отличается от суперскалярности (здесь отбор групп одновременно выполняемых команд происходит непосредственно в ходе выполнения прикладной программы, а не заранее, из-за этого усложняется структура процессора и замедляется скорость его работы).

При архитектуре сверхдлинных командных слов упаковкой RISC-команд в сверхдлинное командное слово занимается компилятор во время трансляции программы. При этом ему, в принципе, доступна вся программа для просмотра. Все команды, упакованные в сверхдлинное слово выполняются одновременно. Если для некоторого поля сверхдлинной команды не удается подобрать подходящую RISC-команду, то на это место ставится команда «ничего не делать». Длина VLIW-слова составляет обычно от 256 до 1024 бит (32 – 128 байт). Архитектура VLIW является безконфликтной по конвейеру.

Процессоры типа VLIW выпускают фирмы Transmeta, Intel и Hewlett-Packard. К VLIW-типу можно отнести и процессоры семейства Эльбрус.

Недостатками VLIW-архитектуры являются сложность регистрового файла, а также повышенная сложность разработки компилятора.

3.5 MISC

Процессор MISC (Minimal Instruction Set Computer) работает с минимальным набором длинных команд и характеризуется небольшим набором чаще всего встречающихся команд. Вместе с этим, в данной архитектуре активно применяется выполнение групп команд за один цикл работы процессора (при этом несколько команд “сливаются” в сверхдлинные командные слова). Порядок выполнения команд при этом распределяется так, чтобы в максимальной степени загрузить маршруты, по которым проходят потоки данных. Таким образом, архитектура MISC объединяет вместе суперскалярную (многопоточную) и VLIW концепции. Компоненты процессора просты и работают с высокими скоростями. Ещё одной особенностью архитектуры MISC является использование стековой организации.

3.6 EPIC

Архитектура EPIC (Explicitly Parallel Instruction Computing) — совместная разработка фирм Intel и Hewlett-Packard, развивающая VLIW-архитектуру. Архитектура EPIC реализована фирмой Intel в 64-разрядном процессоре Itanium и получила название IA-64. В этом процессоре имеется по 120 64-битных регистров общего назначения и 80-битных регистров для данных с плавающей точкой, а также 64-битный регистр предикатов. Группы команд (по три инструкции) упаковываются компилятором в связки (Bundle) длиной 128 бит (16 байт).

По сравнению с архитектурой VLIW, связка в EPIC имеет специальное поле — шаблон, в котором указывается зависимость или её отсутствие команд друг от друга в связке и между связками. Благодаря указанным в шаблоне зависимостям между связками, процессору с N группами функциональных блоков (одна группа содержит три блока) будет соответствовать связка из $3N$ команд. При этом процессоры остаются совместимыми по коду. Таким образом достигается масштабируемость процессоров Itanium.

При ветвлении обе ветви выполняются параллельно, а в регистре предикатов ветвей устанавливаются 1 или 0 при определении истинного направления ветвления. Результаты команд правильной ветви записываются по местам назначения. Два разряда предиката позволяют эффективно выполнять конструкции IF-THEN-ELSE. Таким образом, имея 64-битный регистр предикатов можно выполнить анализ ветвлений на глубину в 32 ветвления.

3.7 Комбинированные системы команд

Комбинированные системы команд CISC — RISC, CISC — RISC — VLIW являются попытками воспользоваться преимуществами объединённых систем. Например, компиляторы и компоненты ОС удобно разрабатывать используя сложные CISC-команды, а RISC-команды при этом могут использоваться как внутренний инструмент обработки информации на более простой аппаратуре устройства управления. Таким образом, сложные исходные команды во время дешифровки заменяются одной, двумя или несколькими более простыми RISC-командами.

Подобная система появилась впервые уже в процессоре Intel Pentium Pro и продолжает использоваться в архитектуре IA-32 (Intel Architecture). В процессорах AMD K8, Intel Pentium 4 можно увидеть уже и элементы VLIW-архитектуры, поскольку на одной из ступеней конвейера RISC-команды объединяются в тройки, запускаемые на исполнение в функциональные блоки.

4 Процессоры семейства x86

4.1 Предшественники x86 — первые модели восьмиразрядных микропроцессоров

4.1.1 Intel 8008

Первый 8-разрядный микропроцессор (1972 г.). Этот процессор содержал 3 500 транзисторов, работал на частоте 500 кГц при длительности машинного цикла 20 мкс (10 периодов задающего генератора) и в отличии от предшественников (Intel 4004 и Intel 4040) имел архитектуру принстонского типа (компьютер с единой памятью для команд и данных — архитектура фон Неймана). В нём допускалось применение комбинаций постоянной и оперативной памяти. Значительные изменения (кроме увеличения разрядности) произошли и в регистровом файле. Из-за ограниченных возможностей применяемой технологии в качестве блока регистров общего назначения была применена динамическая память, которая требовала производить регенерацию (были введены дополнительные аппаратные средства). Большинство команд микропроцессор выполнял за 1 – 3 машинных цикла. Для работы с медленно действующими устройствами был введен сигнал готовности (`ready`). Система команд содержала 65 инструкций и отличалась значительным количеством команд условного перехода, логических команд и команд сдвигов. Процессор мог адресовать память объёмом 16 КБайт, однако объём и организация стека остались прежними, — реализация операций со стеком возлагалась на программиста. Узкий интерфейс связи с «внешним миром» требовал применения около 20 микросхем средней интеграции для сопряжения процессора с памятью и устройствами ввода-вывода.

4.1.2 Zilog Z80

Разработка фирмы Zilog — процессор Z80, помимо расширенной системы команд, одного номинала питания и способности исполнять программы, написанные для i8080, имел ряд архитектурных особенностей.

Регистровая архитектура определяется наличием достаточно большого регистрового файла внутри микропроцессора. Команды получают возможность обратиться к операндам, расположенным в одной из двух запоминающих сред: оперативной памяти или регистрах. К любому регистру можно обратиться непосредственно, поскольку регистры представлены в виде массива запоминающих элементов — регистрового файла. Типичным является выполнение арифметических операций только в регистре, при этом команда содержит два операнда (оба операнда в регистре или один операнд в регистре, а второй в оперативной памяти).

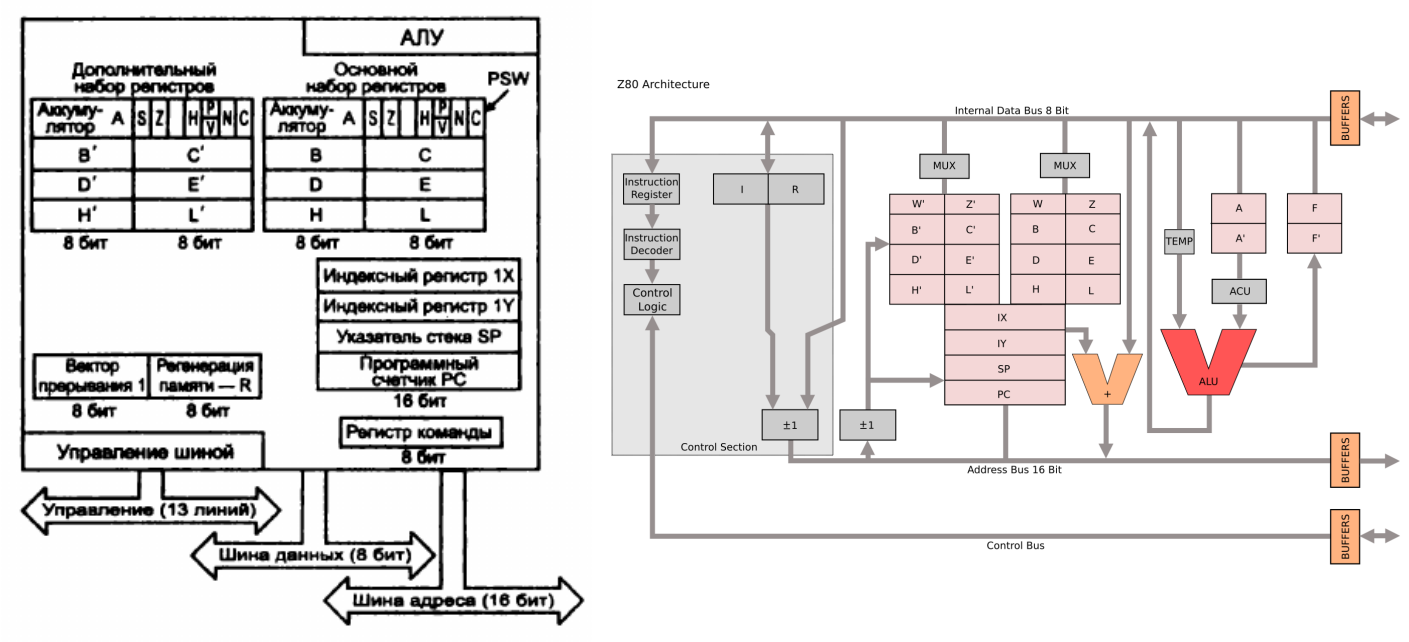


Рис. 3: Архитектура процессора Zilog Z80

4.1.3 Motorola MC6800

Микропроцессор Motorola MC68000 также имел ряд существенных особенностей. Прежде всего, кристалл MC6800 требовал для работы одного номинала питания, а система команд оказалась весьма прозрачной для программиста. Микропроцессор содержал два аккумулятора, и результат операции АЛУ мог быть помещен в любой из них. Но самым ценным качеством структуры MC6800 было автоматическое сохранение в стеке содержимого всех регистров процессора при обработке прерываний (Z80 требовалось для этого несколько команд `push`). Процедура восстановления регистров общего назначения из стека тоже выполнялась аппаратно.

4.1.4 Intel 8080

Микропроцессор Intel 8080 был представлен 1 апреля 1974 г. Благодаря использованию технологии 6 мкм, на кристалле было размещено 6 000 транзисторов. Тактовая частота процессора была доведена до 2 МГц, а длительность цикла команд составила 2 мкс. Объём памяти, адресуемой процессором — 64 КБайт. За счёт использования 40-выводного корпуса удалось разделить шину адреса и шину данных. Общее число микросхем, требовавшихся для построения системы в минимальной конфигурации, равнялось шести.

В блок регистров общего назначения были введены указатель стека, активно используемый при обработке прерываний, а также два программно недоступных регистра для внутренних пересылок. Блок регистров общего назначения был реализован на микросхемах статической памяти. Исключение аккумулятора из блока GPR и введение его в состав АЛУ упростило схему управления внутренней шиной.

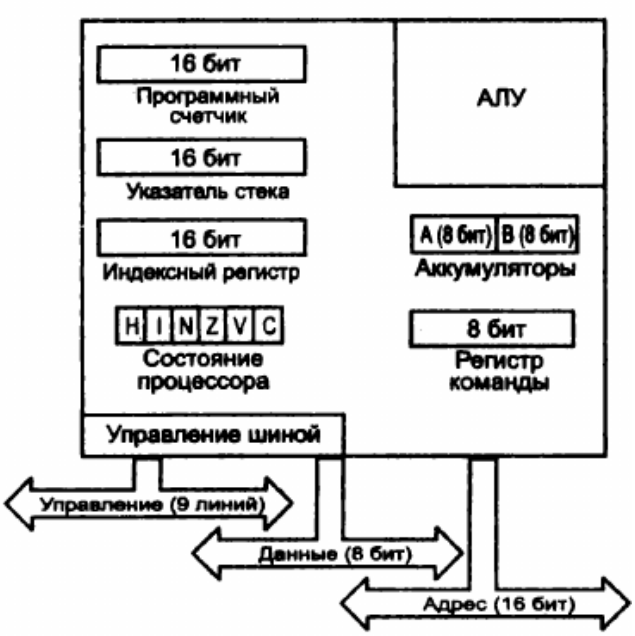


Рис. 4: Архитектура процессора Motorola MC6800

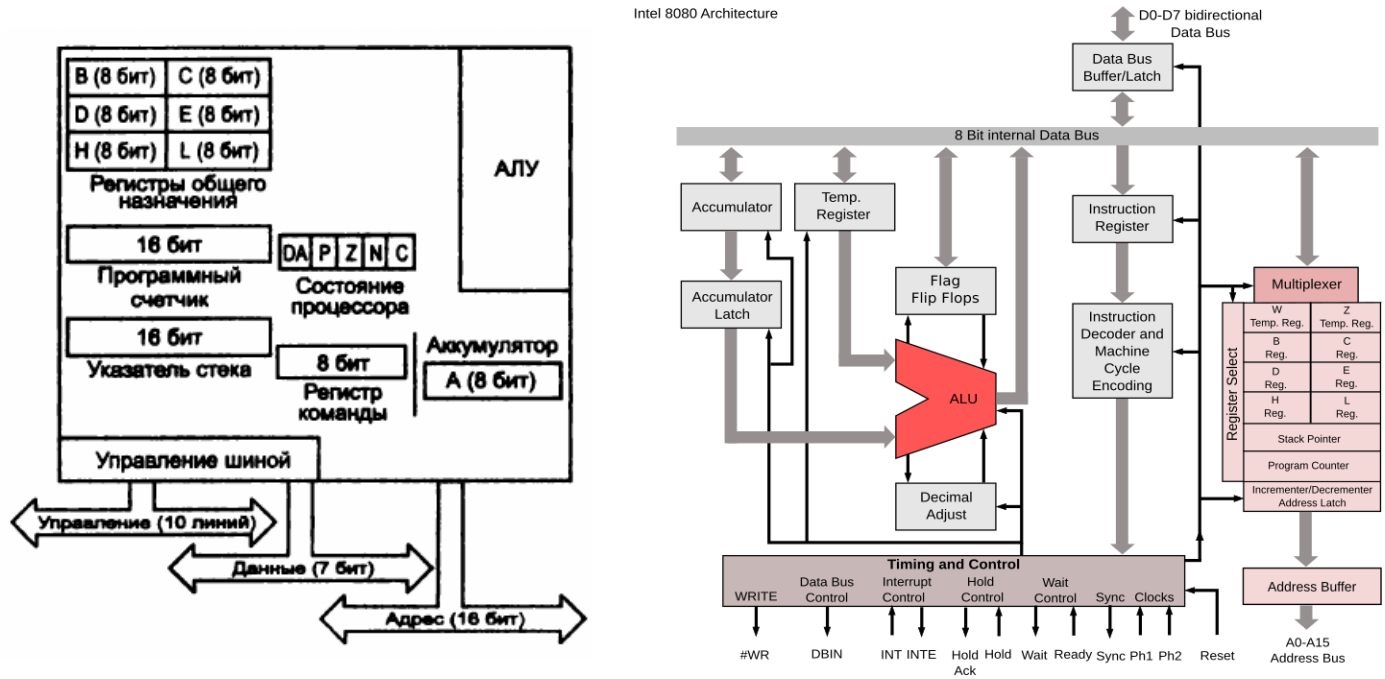


Рис. 5: Архитектура процессора Intel 8080

Новое в архитектуре микропроцессоров — использование многоуровневой системы прерываний по вектору. Такое техническое решение позволило довести общее число источников прерываний до 256.

В i8080 появился механизм прямого доступа в память [2] (ПДП, DMA — Direct Memory Access). ПДП открыл возможность для применения в μЭВМ таких сложных

[2] Как ранее в мэйнфреймах IBM System 360 и др.

устройств, как накопители на магнитных дисках и лентах, дисплеи на ЭЛТ, которые превратили микрокомпьютеры в полноценную вычислительные системы.

Микропроцессор был окружен целым семейством новых микросхем (БИС контроллера ПДП, контроллера прерываний и др.). В результате этого проектирование микроЭВМ на базе семейства БИС значительно упростилось. Микропроцессор i8080 стал стандартом де-факто, но у него были и недостатки (например, уже имелись микропроцессоры других фирм, которые оказались более "прозрачными" для программистов). Далее была выпущена серия периферийных контроллеров. Но самое значительное достижение состояло в том, что компания Intel создала системное программное обеспечение — однопользовательскую операционную систему ISIS II и ОС реального времени iRMX-80 (мощнейшая в то время программная поддержка своих изделий). Фирма в 1976 г. приступила к выпуску одноплатных микроЭВМ серии iSBC на базе своих комплектов микросхем.

4.2 Первое поколение x86

4.2.1 Intel 8086

Объявлен 8 июня 1978 г. Микропроцессор содержал 29 000 транзисторов. Высокое быстродействие элементов обеспечило тактовую частоту 5 МГц, а 16-разрядная архитектура и машинный цикл 200 нс — производительность, превышающую аналогичный параметр 8080 на порядок. Именно стратегия эволюционного, а не революционного развития, выбранная фирмой Intel была верна и давала свои плоды. Программная совместимость была исключительно важной характеристикой, которая объединяла 86-й кристалл с его предшественниками. Структура микропроцессора была полностью перестроена, он как бы был разделен на два одновременно работающих функциональных блока. Это операционный блок (EU — Execution Unit) и блок интерфейса (BIO — Bus Interface Unit). В результате исполнение одной команды совмещалось во времени с выборкой следующей команды или данных из памяти. Таким образом, из универсальных ЭВМ микропроцессоры позаимствовали ещё одно техническое решение — реализацию принципов параллелизма. В ЦП появился небольшой буфер команд, что давало дополнительную экономию времени при обращениях к памяти.

Адресация 1 МБайт оперативной памяти (благодаря 20 адресным линиям) и её сегментация могут быть отнесены к одним из наиболее существенных достижений. Сегментация памяти и большое число уровней прерываний были ориентированы на работу системы в многозадачном режиме. Однако механизм защиты памяти пока реализован не был, и это в ряде случаев существенно усложняло разработку программного обеспечения.

Наряду с поддержкой ввода-вывода по каналу ПДП дополнительно обеспечивалась адресация до 64 000 портов программно-управляемого ввода-вывода. Процессор Intel 8086 мог работать в двух режимах: минимальном (рассчитанном на исполь-

зование в небольших системах без применения БИС контроллера шины) и максимальном (ориентированном на применение микропроцессора в сложных системах с использованием БИС контроллера шины). В систему команд входило 147 инструкций, позволяющих решать задачи управления практически любой сложности. Появились операции умножения и деления 16-разрядных чисел со знаком и без знака, команды обработки массивов данных, программно-управляемые прерывания и др., что превратило микросхему в универсальный прибор, который мог успешно применяться как для построения сложных контроллеров, так и в качестве центрального процессора ЭВМ общего назначения. Кроме того, микропроцессор вышел в мощном сопровождении средств поддержки (вспомогательных БИС, средств разработки и отладки аппаратуры и системного ПО и т. д.).

Использование микропроцессора Intel 8086 в IBM PC предопределило дальнейшее развитие корпорации Intel как разработчика и изготовителя универсальных процессоров общего назначения. Был изготовлен 16-разрядный арифметический сопроцессор 8087, который позволил превратить ПК ещё и в достаточно мощный инструмент для решения задач вычислительного характера.

4.2.2 Intel 8088

Наряду с этим фирмой Intel был выпущен процессор 8088 с 8-разрядной внешней шиной данных. Из-за применения экономичных 8-разрядных микросхем появился персональный компьютер с микропроцессором Intel 8088 (IBM PC XT — Extended Technology — расширенная технология, тактовая частота 4.77 МГц).

4.3 Архитектура и микроархитектура базового процессора x86

Определение 7. *Под **базовым микропроцессором** семейства x86 мы будем понимать упрощённую схему процессора, основными устройствами которого являются шестнадцатиразрядные регистры и арифметико-логическое устройство процессора Intel 80286.*

Процессор Intel 80286 (выпущенный 1 февраля 1982 г.) относится ко второму поколению процессоров семейства x86. В этом процессоре было введено большое количество новшеств по сравнению с процессорами предыдущего поколения.

- Повышенное быстродействие (4 МГц, однако вскоре рабочая частота была повышена до 8 МГц и стала стандартной, хотя производители клонов эту частоту довели до 10, 12.5, 16 и 20 МГц).

- Адресное пространство составляло 16 МБайт (вместо 1 МБайт у предшественников), так как использовалась 24-разрядная шина адресов.

- Аппаратная мультизадачность (эта архитектурная новинка позволяла в ПК одновременно выполнять несколько задач с большой скоростью переключения с одной на другую).
- Встроенная система управления памятью и средства её защиты (открывали широкие возможности использования микропроцессора в многозадачных средах).
- Поддержка виртуальной памяти (это позволяло использовать внешнюю память для имитации большой реальной внутренней памяти ёмкостью до 1 ГБайт).
- Дополнение системы команд 16 новыми инструкциями;
- Размещение на одном кристалле контроллеров прерываний и DMA, а также таймера и системного генератора.

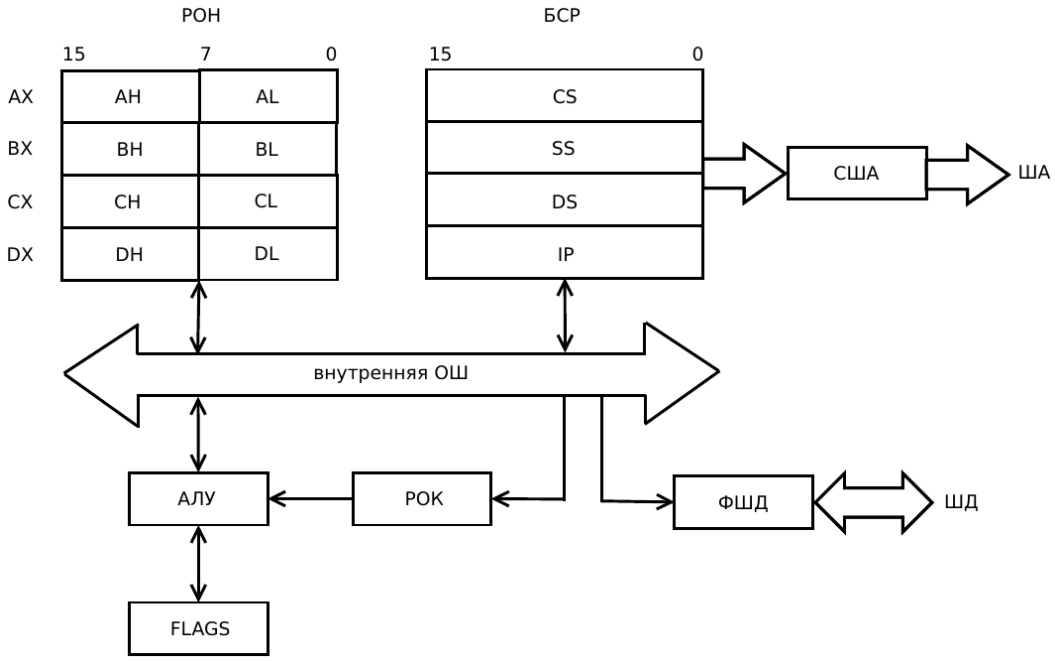


Рис. 6: Упрощённая схема процессора Intel 80286

Все регистры процессора 80286 имеют размер слова (16 бит), за каждым из них закреплено определенное имя (`AX`, `SP` и т. п.). По назначению и способу использования регистры можно разбить на следующие группы.

- Регистры общего назначения: `AX`, `BX`, `CX`, `DX`, `SP`, `BP`, `SI`, `DI`.
- Сегментные регистры: `CS`, `DS`, `SS`, `ES`.
- Счётчик команд `IP`.

Обозначение	Название	Старший байт	Младший байт
AX	Accumulator	AH	AL
BX	Base	BH	BL
CX	Counter	CH	CL
DX	Data	DH	DL
SP	Stack Pointer	-	-
BP	Base Pointer	-	-
SI	Source Index	-	-
DI	Destination Index	-	-
CS	Code Segment	-	-
DS	Data Segment	-	-
SS	Stack Segment	-	-
ES	Extra Segment	-	-
IP	Instruction Pointer	-	-

Таблица 1: Регистры процессора 80286

- Регистр флагов FLAGS.

Регистры общего назначения можно использовать во всех арифметических и логических командах. В то же время каждый из них имеет определённую специализацию (некоторые команды «работают» только с определёнными регистрами, см. таб. 1). Например, команды умножения и деления требуют, чтобы один из операндов находился в регистре AX (либо в паре регистров AX и DX, в зависимости от размера операнда), а команды управления циклом используют регистр CX в качестве счётчика цикла. Регистры BX и BP очень часто используются как базовые регистры, а BP и DI — в роли индексных. Регистр SP обычно указывает на вершину стека, поддерживаемого аппаратно.

Регистры AX, BX, CX и DX конструктивно устроены так, что возможен независимый доступ к их старшей и младшей половинам. Можно сказать, что каждый из этих регистров состоит из двух байтовых регистров, обозначаемых AH, AL, BH и т. д. (H — High, старший; L — Low, младший). Таким образом, с каждым из этих регистров можно работать как с единым целым, так и с его частями, например — записать слово в AX, а затем считать только часть слова из регистра AH или заменить только часть в регистре AL и т. д. Такое устройство регистров позволяет использовать их как для работы с числами, так и со строками. Все остальные регистры не делятся на полурегистры, поэтому прочитать или записать их содержимое (16 битов) можно только целиком.

Сегментные регистры CS, DS, SS и ES не могут быть операндами никаких команд, кроме команд пересылки и стековых команд. Эти регистры используются только для сегментирования адресов.

Счётчик команд IP всегда содержит адрес (смещение от начала программы) той

команды, которая должна быть выполнена процессором следующей (адрес начала программы хранится в регистре CS). Содержимое регистра IP можно изменить только командами перехода.

Также в схеме базового процессора имеется особый регистр флагов. Флаг — это отдельный бит, принимающий значение «1» (флаг поднят или установлен) или «0» (флаг сброшен). Различают два вида флагов.

- Флаги условий — автоматически меняются при выполнении команд и фиксируют те или иные свойства их результата (например, равен ли он нулю).
- Флаги состояний — они меняются программно и оказывают влияние на дальнейшее поведение процессора (например, блокируют прерывания).

В процессоре Intel 80286 используются девять флагов, каждому из них присвоено определенное имя (см. таб. 2).

Биты	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Флаги	-	-	-	-	OF	DF	IF	TF	SF	ZF	-	AF	-	PF	-	CF

Таблица 2: Регистр флагов процессора 80286

Флаги условий.

- Флаг переноса CF (Carry Flag). Принимает значение 1, если при сложении целых чисел появилась единица переноса, не укладывающаяся в разрядную сетку, или если при вычитании чисел без знака первое из них было меньше второго. В командах сдвига в CF заносится бит, вышедший за разрядную сетку. CF фиксирует также особенности команды умножения.
- Флаг переполнения OF (Overflow Flag). Устанавливается в 1, если при сложении или вычитании целых чисел со знаком получился результат, по модулю превосходящий допустимую величину (произошло переполнение “смысловой части числа” и она оказалась в знаковом разряде).
- Флаг нуля ZF (Zero Flag). Устанавливается в 1, если результат команды оказался равным нулю.
- Флаг знака SF (Sign Flag). Устанавливается в 1, если в операции над числами со знаками получился отрицательный результат.
- Флаг чётности PF (Parity Flag). Поднимается, если результат очередной команды содержит чётное количество двоичных единиц. Учитывается обычно только при операциях ввода/вывода.

- Флаг дополнительного переноса `AF` (Auxiliary carry Flag). Фиксирует особенности выполнения операций над двоично-десятичными числами.

Флаги состояний.

- Флаг направления `DF` (Direction Flag). Устанавливает направление просмотра строк в строковых командах: при `DF = 0` строки просматриваются «вперед» (от начала к концу), при `DF = 1` — в обратном направлении.

- Флаг прерываний `IF` (Interrupt Flag). При `IF = 0` процессор перестаёт реагировать на поступающие к нему прерывания. В обратном случае блокировка прерываний снимается.

- Флаг трассировки `TF` (Trap Flag). При `TF = 1` после выполнения каждой команды процессор делает прерывание (с номером 1), чем можно воспользоваться при отладке программы для ее трассировки.

В 1984 г. фирма IBM использовала микропроцессор Intel 80286 в компьютерах PC AT (AT — Advanced Technology — улучшенная технология). Данный процессор мог работать в двух режимах — реальном (действовал как 8086, что обеспечивало совместимость с DOS и существующим ПО) и защищённом (в этом режиме МП реализовывал режим виртуальной памяти, аппаратную мультизадачность и адресацию к большему пространству памяти).

Операционная система MS DOS может работать только в реальном режиме. Другие операционные системы, например, OS/2 (предложенная фирмой IBM альтернатива DOS) и UNIX (XENIX и AIX), могут использовать защищённый режим и, следовательно, расширенные возможности 80286. Многие новшества, впервые введённые в этом процессоре, впоследствии переходили к процессорам фирмы Intel следующих поколений. Имелись и определённые нерешённые проблемы, связанные с многозадачностью, повышением производительности, совершенствованием тракта процессор — память и устройства управления памятью (для эффективного функционирования ПК под управлением многозадачных ОС).

4.4 Архитектура IA-32 (x86-32)

Первым представителем новой архитектуры IA-32 был микропроцессор Intel 80386SX, выпущенный 17 октября 1985 г. Этот процессор относят к третьему поколению процессоров семейства x86. Данный микропроцессор был процессором для ЭВМ общего назначения. Размещение на кристалле 275 тысяч транзисторов дало возможность полностью реализовать 32-разрядную архитектуру.

Главные особенности:

1. обеспечивает 32-разрядный ввод-вывод, 32-битовую адресацию основной памяти (адресуемая реальная память — до 4 ГБайт, т. е. 2^{32} байт) и ёмкостью до 64 Тбайт ($64 \cdot 2^{40}$ байт) виртуальной памяти;

2. рабочая тактовая частота равнялась 33 МГц;

3. в микропроцессор были встроены система управления памятью и защиты (регистры преобразования адреса, механизмы защиты оперативной памяти, улучшенные аппаратные средства поддержки многозадачных ОС), средства работы с виртуальной памятью со страничной организацией памяти (в устройство менеджера памяти — MMU (Memory Management Unit) помимо блока сегментации — SU (Segmentation Unit) был включен блок управления страницами — PU (Paging Unit), благодаря этому относительно просто реализовывались процессы свопинга (перестановка сегментов из одного места памяти в другое).

Предварительная выборка команд, буфер для команд (внутренняя кэш-память) 16 байт, конвейер команд и механизмы выполнения функций преобразования адреса значительно уменьшили среднее время выполнения команды (3 – 4 млн. команд в секунду, что в 6 – 8 раз превышало аналогичный показатель 8086). Как и раньше, новый процессор был совместим со своими предшественниками на уровне объектных кодов.

Наиболее существенной особенностью 80386 было использование кэш-памяти, значительно повышающей производительность системы (ещё один атрибут универсальных ЭВМ, который начал применяться в микропроцессорных системах). Для управления этой памятью был разработан специальный контроллер, с помощью которого формировался двухходовый множественный ассоциативный кэш (обеспечивал буфер ёмкостью до 32 КБайт и высокий коэффициент удачных обращений). Но математический сопроцессор был ещё автономным на отдельном кристалле (80387).

В процессоре 80386 реализованы три режима работы: реальный, защищенный и виртуальный процессор 8086. За счёт этого можно считать, что процессор 80386 как бы включает в себя три разных процессора.

В реальном режиме (Real Mode, стартовый режим работы компьютера) процессор 80386 ведёт себя как 8086 с расширенным набором команд и имеющий доступ только к первому МБайту памяти (т. к. используется 20-разрядная адресация). При этом возможности 80386 используются не полностью, но могут выполняться все программы, написанные для 8086/8088 и причём значительно быстрее.

Защищённый режим (Protected Mode) 80386 соответствует аналогичному режиму 80286, имеет доступ к 16 МБайт памяти и расширенному набору команд, а также имеет возможность использовать систему мультипрограммирования (в основном могут выполняться несколько прикладных программ, чаще работающих в среде Windows, поддерживающей защищённый режим лучше, чем DOS). Если заменить ОС на OS/2 (разработка специально для защищенного режима) или UNIX/Linux, то появится возможность полностью использовать функциональные возможности этого режима.

Третий режим 80386 — Virtual Real Mode. В этом режиме процессор 80386 одновременно заменяет некоторое количество параллельно работающих 8086/8088, т. е.

одновременно могут быть задействованы несколько программ, которые выполняются соответствующими процессорами 8086/8088. Здесь нет ограничения 1 МБайт на память. Ядром многозадачности является основная программа, переключающая процессор в виртуальный режим и контролирующая текущие процессы выполнения различных программ (например, система Windows).

80386 внутренне одновременно оперирует 32 битами и имеет внешний 32-битный интерфейс, но к тому времени большинство устройств и микросхем были 16-разрядными и не могли использовать эту возможность процессора. Поэтому Intel повторила опыт процессора 8088 (8086 — 16 бит внутренние и внешние, а 8088 — 16 бит внутренние и 8 бит внешние) и создала 80386 с 16-битным интерфейсом (он получил название 80386 SX), который оказался меньше и дешевле. Полноразрядный 80386 получил название 80386 DX.

В 32-разрядных (и выше) ЦП картина регистров изменяется. Как 386, так и все другие процессоры IA-32 имеют восемь 32-разрядных РОН (GPR) для использования прикладными программами.

Есть также системные регистры, которые используются главным образом операционными системами, но не приложениями. Это — сегментные (segment), контрольные (control), отладочные (debug) и тестовые (test). Шесть сегментных регистров используются главным образом для управления памятью. Число контрольных, отладочных и тестовых регистров изменяется от модели к модели ЦП.

Регистры общего назначения x86-32 также являются специализированными и подразделяются на регистры данных и регистры адресации[3].

Программно-доступны также именуемые 8- и 16-битные части регистров, аналогичные регистрам 80286.

5 Технологии повышения производительности процессоров

5.1 Операции над вещественными числами (с плавающей точкой)

Сопроцессоры. Для расширения вычислительных возможностей центрального процессора — выполнения арифметических операций, вычисления основных математических функций (тригонометрических, показательных, логарифмических) и т. д. — в состав ВМ добавляется математический сопроцессор. Применение сопроцессора повышает производительность вычислений в сотни раз. В разных поколениях процессоров он назывался по-разному — FPU (Floating Point Unit — блок чисел/операций с плавающей точкой) или NPX (Numeric Processor eXtension — числовое расширение

[3]Отметим, что в расширении AMD64 это разграничение ликвидируется, и регистры общего назначения здесь действительно могут применяться для любых операций.

Имя	Номер	Назначение
		Регистры данных
EAX	001	Выделенный сумматор, используемый во всех основных вычислениях
ECX	001	Универсальный счётчик циклов
EDX	010	Регистр данных, дополняющий сумматор
EBX	011	Буферное запоминающее устройство; в 16-битном режиме — указатель
		Адресные регистры
ESP	100	Указатель стека
EBP	101	Указатель базы. Используется для хранения адреса текущего фрейма стека. Иногда применяется в качестве буфера
ESI	110	Индекс источника
EDI	111	Индекс назначения

Таблица 3: Регистры общего назначения IA-32

процессора).

Для процессоров 386 и младше сопроцессор был отдельной микросхемой, подключаемой к локальной шине основного процессора. В любом случае сопроцессор исполняет только свои специфические команды, а всю работу по декодированию инструкций и доставке данных осуществляет ЦП.

Блоки операций с плавающей точкой. С программной точки зрения сопроцессор и процессор выглядят как единое целое. В современных (486+) процессорах FPU располагается на одном кристалле с центральным процессором.

Начиная с 80486 восемь регистров математического сопроцессора (именуемых `ST(0)` – `ST(7)`) встраивают в центральный процессор. Каждый из этих регистров имеет разрядность 80 бит и хранит числа в формате расширенной точности стандарта IEEE 754 (15 бит — экспонента, 64 — мантисса).

Эти регистры доступны в стековом порядке. Имена (номера) регистров устанавливаются относительно вершины стека (`ST(0)` — вершина стека, `ST(1)` — следующий регистр ниже вершины стека, `ST(2)` — второй после вершины стека и т. д.). Таким образом, вводимые данные всегда сдвигаются «вниз» от вершины стека, а текущая операция совершается с содержимым вершины стека.

5.2 Конвейерная обработка команд

Как уже говорилось выше, обработка команды, или цикл процессора, может быть разделена на несколько основных этапов (микрокоманд), которых как минимум пять

(выборка, декодирование, чтение исходных данных, выполнение, запись результата).

Каждая короткая операция требует для своего выполнения времени, равного такту генератора процессора (tick of the internal clock). Отметим, что к длинным операциям (плавающая точка) это не имеет отношения. Очевидно, что при тактовой частоте в 100 МГц быстродействие составит 20 миллионов таких операций в секунду.

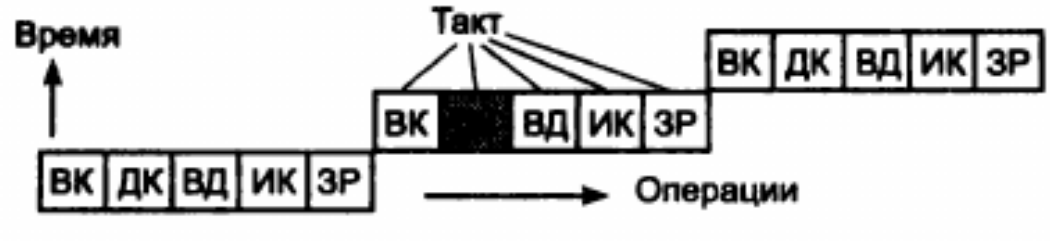


Рис. 7: Выполнение команд без конвейеризации

Все этапы команды задействуются только один раз и всегда в одном и том же порядке — одна за другой. Это, в частности, означает, что если логическая схема первой микрокоманды выполнила свою работу и передала результаты второй, то для выполнения текущей команды она больше не понадобится, и, следовательно, может приступить к выполнению следующей команды программы.

Такая технология обработки команд носит название конвейерной (pipeline) обработки. Каждая часть устройства называется ступенью (стадией) конвейера, а общее число ступеней — длиной линии конвейера.

Конвейеризация осуществляет многопоточную параллельную обработку команд, так что в каждый момент одна из команд считывается, другая декодируется и т. д., и всего в обработке одновременно находится пять команд. Таким образом, на выходе конвейера на каждом такте процессора появляется результат обработки одной команды (одна команда в один такт).

Из рисунка видно, что, начиная с пятого такта, все ступени конвейера работают непрерывно и параллельно. Все пять команд выполняются за 9 тактов, тогда как при их обычном выполнении понадобилось бы $5 \cdot 5 = 25$ тактов. Таким образом, ускорение для первых пяти команд составит $25/9 \approx 2.78$. В общем виде коэффициент ускорения S в идеальном случае будет

$$S = \frac{N_\mathrm{c} \cdot N_\mathrm{s}}{N_\mathrm{c} + N_\mathrm{s} - 1}, \tag{1}$$

где N_c — это число команд, запущенных в конвейер, а N_s — число стадий конвейера (количество тактов выполнения команды).

Предельное ускорение при $N_\mathrm{c} \to \infty$ будет равно N_s, т. е. количеству ступеней конвейера. Чтобы заработали все N стадий конвейера, требуется $N - 1$ такт, после чего достигается максимальная производительность.

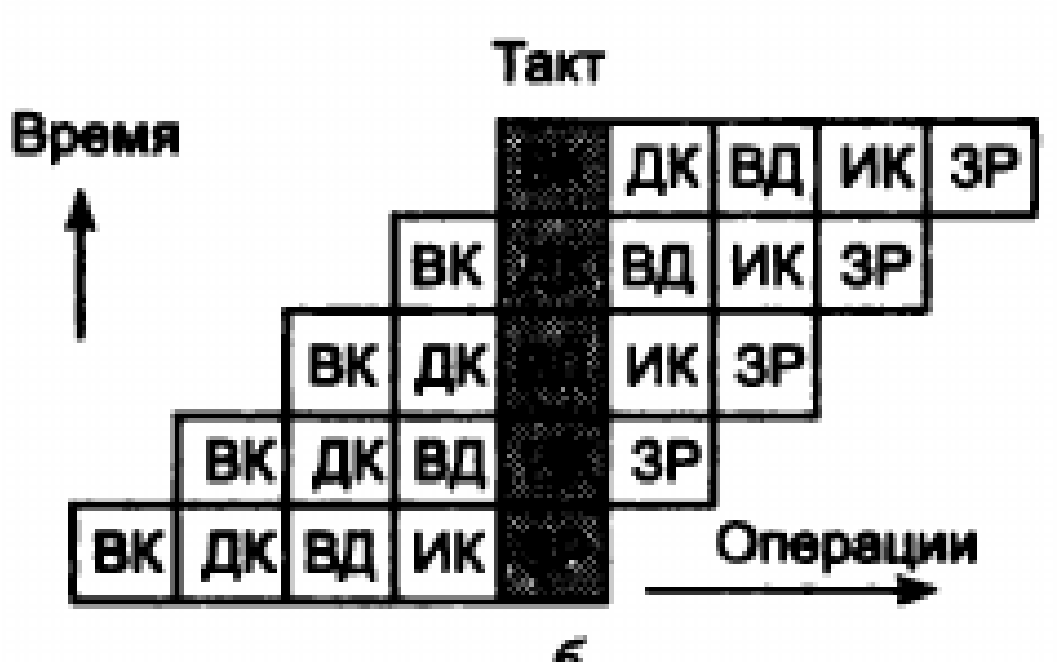


Рис. 8: Пятиступенчатый конвейер

Приведенный пример процессора (5 микроопераций) является гипотетическим — в реальных ЦП конвейер обработки команд сложнее и включает большее количество ступеней. Причина увеличения длины конвейера заключается в том, что многие команды являются довольно сложными и не могут быть выполнены за один такт процессора, особенно при высоких тактовых частотах. Поэтому каждая из упомянутых пяти стадий обработки команд в свою очередь может состоять из нескольких ступеней конвейера.

Во многих вычислительных системах, наряду с конвейером команд, используются конвейеры данных. Сочетание этих двух конвейеров дает возможность достичь очень высокой производительности на определенных классах задач, особенно если используется несколько различных конвейерных процессоров, способных работать одновременно и независимо друг от друга.

5.2.1 Конфликты в конвейерах

Под конфликтом конвейера понимается невозможность выполнения команды в соответствующем ей такте. При этом в конвейер нельзя загрузить очередную команду и его производительность падает. Выделяют три вида конфликтов:

- структурные;
- конфликты по данным;
- конфликты по управлению.

Структурные конфликты порождаются ограниченностью ресурсов, что не позволяет их использовать одновременно несколькими командами, например, ограниченное число общих регистров, количество входов в буферы, доступ в кеш для одновременной выборки команды и операнда и пр. Для устранения структурных конфликтов ресурсы дублируют, используют раздельные кэш для команд и данных, что усложняет и удорожает процессор.

Конфликты по данным — это следствие логической зависимости команд между собой, например, если результат предшествующей команды является операндом следующей. При этом очередная команда не будет выполняться до тех пор, пока не выполнится до конца предшествующая и не передаст ей свой результат. По конвейеру будет проходить «пузырь», равный времени задержки. Это значит, что конвейер будет работать безрезультатно некоторое время. Эффективность работы конвейера при зависимостях по данным определяеся особенностями программы и архитектурой процессора.

Конфликты по управлению порождаются использованием в программах условных переходов. Какая ветвь программы будет исполняться можно определить только после анализа условия ветвления. Простои из-за конфликтов по управлению могут оказаться значительными, т. к. в среднем треть операторов программы являются операторами переходов. Операции безусловных переходов выполняются за счёт предвыборки команд в буфер, из которого выбирается команда соответствующей ветви, поскольку такие команды распознаются без потери времени прцессором и, следовательно, поток команд, запускаемых в конвейер не прерывается. С оптимизацией работы конвейера при обработке условных переходов дело обстоит сложнее, но также решается путём предсказания перходов с вероятностью более 90%. Если предсказание оказалось ошибочным, что определяется при фактическом выполнении команды перехода, то конвейер сбрасывается и запускается с команды верного целевого адреса.

Уменьшить количество остановок конвейера можно, изменяя очерёдность команд при трансляции прграммы. Такой метод называется планированием загрузки конвейера.

5.2.2 Суперскалярные процессоры

Скалярный процессор — это процессор с единственной линией конвейера (к этому типу процессоров относятся, например, все процессоры Intel до 486 включительно). Суперскалярный процессор имеет более одной линии конвейера. С ростом числа линий конвейера и увеличением числа ступеней на линии увеличивается пропускная способность процессора при неизменной тактовой частоте. Наоборот, чем больше ступеней насчитывается в конвейере, тем меньшая работа выполняется за такт, и тем выше требуется поднимать частоту процессора для повышения производительности.

Pentium — первый суперскалярный процессор Intel. В нём организованы две линии, что позволяет ему при одинаковых частотах быть вдвое производительней

i80486, выполняя сразу две инструкции за такт.

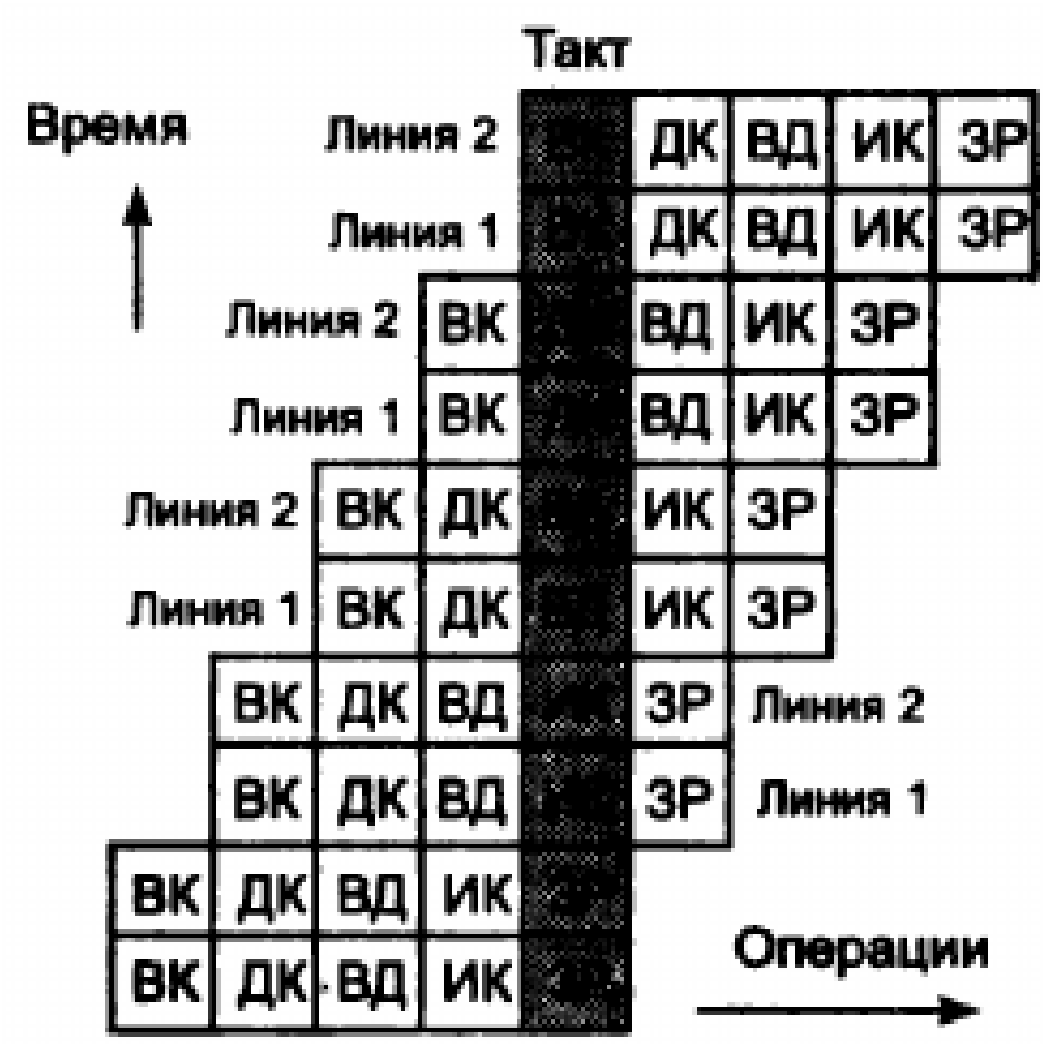


Рис. 9: Суперскалярный конвейер из двух линий

В суперскалярных процессорах используется параллелизм на уровне команд. Однако не всегда возможно парллельное выполнение команд по трём причинам.

Конфликты по доступам к ресурсам происходят, если несколько команд одновременно должны использовать один ресурс: регистр, оперативную память, АЛУ или что-то ещё. Это аналогично структурным конфликтам. Уменьшить их количество можно дублированием устройств, увеличением количества регистров, созданием буферов для отложенных записей, расслоением памяти на независимые банки, увеличением портов доступа к ресурсу.

Зависимость по управлению связана с обработкой ветвлений и использованием команд переменной длины. В последнем случае выборку следующей команды нельзя сделать до тех пор, пока не завершено декодирование предыдущей. В первом случае используют методику предсказания ветвлений и предварительгую (упреждающую) выборку команд предсказанной ветви и её исполнение до реального выполнения перехода. В более продвинутых процессорах выполняют одновременно обе ветви и вы-

бирают результаты нужной ветви по результату выполнения перехода. Используется также оптимизация операций ветвлений при трансляции программы. Во втором случае предусматривают буфера предвыборки команд, а в Pentium 4 — и предварительную расшифровку выбранных в буфер предвыборки команд и их упорядочение в виде трасс микроопераций.

Конфликты по данным возникают, если очередная команда не может выполниться, т. к. для этого требуется результат предыдущей. Особенно длительные задержки происходят при промахах по кэш и вынужденным обращениям в оперативную память за нужной единицей данных. Такая зависимость — это свойство программы и, следовательно, не может устраняться компилятором или аппаратно. Можно хотя бы частично избежать простоев, если на время формирования результата предшествующей команды загрузить исполнительные узлы выполнением других команд.

Итак, для разрешения возможных конфликтов используют предвыборку, прогнозирование переходов и спекулятивное (условное) выполнение команд. В суперскалярных процессорах используется динамический запуск команд в конвейеры, т. е. не в порядке их следования в программе (Out-of-Order). Однако выборка команд из памяти и признание их выполненными и размещение их результатов по логическим адресам (регистрам и ячейкам памяти) производится в естественном порядке (in Order). Для эффективной реализации такого подхода требуется большой буфер (Pool) для размещения команд — кандидатов на исполнение.

В суперскалярных процессорах одновременно можно запускать на обработку столько команд сколько имеется конвейеров. Однако в длительном режиме работы неизбежны конфликты и среднее число запускаемых команд оказывается меньше пикового. Поэтому производительность суперскалярных процессоров оказывается переменной.

5.3 Увеличение разрядности систем

В 1980-е годы соответствие между типом ЭВМ и её разрядностью имело простейший вид:

- микроЭВМ — 8 разрядов;
- мини-ЭВМ — 16 разрядов;
- большие ЭВМ — 32 разряда;
- сверхбольшие (супер) ЭВМ — 64 разряда.

В процессе развития микропроцессоров рубежи в 16 и 32 разряда (IA-32) были преодолены довольно быстро, а в начале 2000-х г. в процессорах фирм Intel и AMD произошел переход на 64-разрядные архитектуры.

Преимущества 64-битной архитектуры микропроцессоров главным образом относятся к памяти. Если взять два идентичных микропроцессора, и один из них будет

32-битным, а другой — 64-битным, то последний сможет адресовать намного больший объём памяти, чем первый. Известны следующие 64-разрядные архитектуры (64-bit architecture).

5.3.1 IA-64

Спецификация IA-64 означает «Архитектура Intel, 64 бита», но связь с IA-32 здесь только по названию, так как архитектура IA-64 не совместима непосредственно с набором команд IA-32. Здесь появляется полностью отличный набор команд, а также используются принципы VLIW. IA-64 — архитектура, используемая линией процессоров Itanium.

Усовершенствования:

- в 16 раз увеличено количество регистров общего назначения и регистров для чисел с плавающей точкой (теперь по 128);
- механизм переименования/ротации регистров, чтобы сохранять значения в регистрах при вызове функций.

5.3.2 AMD64

Набор команд AMD64, первоначально названный x86-64, в значительной степени построен на основе IA-32 и таким образом обеспечивает наследственность семейства x86. При расширении набора команд компания AMD воспользовалась возможностью, чтобы очистить часть его от ряда устаревших команд — «16-bit legacy».

Основные усовершенствования.

- в два раза увеличено количество регистров общего назначения и SSE (теперь по 16);
- регистры общего назначения (`RAX` – `RDI`, `R8` – `R15`) теперь действительно **общего назначения** и ничем больше не ограничены.

Архитектура использована в ЦП Athlon 64, Athlon 64 X2, Athlon 64 FX, Opteron, Turion 64, Turion 64 X2, Sempron («Palermo», «Manila»).

5.3.3 EM64T или Intel 64

EM64T — Extended Memory 64-bit Technology — набор команд (ранее известный как Yamhill), объявленный Intel в феврале 2004 г., в подражание AMD64. EM64T в целом совместим с кодами, написанными для AMD64, хотя и имеет ряд недостатков сравнительно с AMD64.

Intel стала использовать набор EM64T, начиная с ЦП Хеоп (ядро Nocona) в конце 2004 г., а затем вышла с ним на рынок настольных ПК в начале 2005 г. (Pentium IV, версия E0).

EMT/Intel 64 используется в ЦП архитектуры Intel NetBurst — Xeon («Nocona»), Celeron D («Prescott» и далее), Pentium 4 («Prescott» и далее), Pentium D, Pentium Extreme Edition и архитектуры Intel Core — Xeon («Woodcrest»), Intel Core 2.

Поскольку AMD64 и EMT64 почти не различаются, для ссылки на них используются нейтральные названия — x86-64, x86_64 (Linux и Apple Mac OS X), x64 (Microsoft и Sun Microsystems).

5.4 Векторная обработка (SIMD-команды)

В классификации Флинна имеется раздел SIMD — множество потоков данных, обрабатываемых одной командой. Процессоры, реализующие такую обработку, именуют потоковыми процессорами. Могут быть определены как однопотоковые (Single-Streaming Processor — SSP), так и многопотоковые процессоры (Multi-Streaming Processor — MSP).

Типичными представителями класса SIMD считаются матричные процессоры — ILLIAC IV, ICL DAP, Goodyear Aerospace MPP, Connection Machine 1 и т. п. В таких системах единое управляющее устройство контролирует множество процессорных элементов. Каждый процессорный элемент получает от устройства управления в каждый фиксированный момент времени одинаковую команду и выполняет её над своими локальными данными. В последовательных расширениях системы команд x86, выполненных Intel и AMD, всё более полно используются принципы обработки одной командой вектора (потока) данных.

5.4.1 MMX

MMX (MultiMedia eXtension) — архитектура системы команд (57 команд для арифметики с фиксированной точкой), непосредственно предназначенных для задач мультимедиа, связи и графических приложений, которые часто используют сложные алгоритмы, исполняющие одинаковые операции на большом количестве типов данных (байты, слова и двойные слова). Анализ участков таких программ с большим объёмом вычислений показал, что такие приложения имеют следующие общие свойства, определившие выбор системы команд и структуры данных:

- небольшая разрядность целочисленных данных (например, 8-разрядные пиксели для графики или 16-разрядное представление речевых сигналов);
- небольшая длина циклов, но большое число их повторений;
- большой объём вычислений и значительный удельный вес операций умножения и накопления;
- существенный параллелизм операций в программах.

Это и определило новую структуру данных и расширение системы команд. При этом было достигнуто общее повышение производительности на 10–20%, а в программах обработки мультимедиа — до 60%.

В процессоре Pentium MMX (1996 г.) появляются 8 новых 64-разрядных «регистров» с именами `MM0`–`MM7` (или же `MMn`). В действительности, эти новые «регистры» были только псевдонимами для регистров стека математического сопроцессора x87. В связи с тем, что каждый из регистров стека содержит 80 бит, старшие 16 бит регистров стека оказываются неиспользованными в MMX, поэтому в режиме MMX они заполняются всеми единицами, что позволяет отличать данные формата с плавающей точкой от MMX-данных.

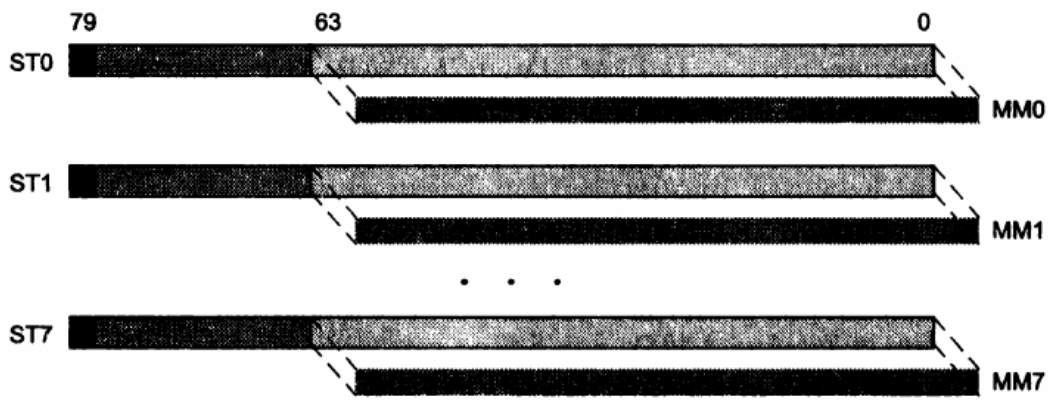


Рис. 10: Совмещение стека регистров FPU с MMX-регистрами

Преимущество совмещения MMX с регистрами FPU состоит в том, что одни и те же команды и структуры данных различных операционных систем (при обработке прерываний и вызове подпрограмм) могут использоваться как для сохранения содержания регистров FPU, так и регистров MMX.

Каждый из регистров MMX предназначен для целых чисел на 64 бита. Однако главным понятием набора команд MMX является концепция упакованных типов данных.

Упакованные целые числа (и знаковые, и беззнаковые):

1. Packed byte (B) — восемь упакованных байт;
2. Packed word (W) — четыре упакованных слова;
3. Packed doubleword (D) — два упакованных двойных слова;
4. Quadword (Q) — учетверённое слово.

MMX-команды используются в ЦП, начиная с Pentium MMX (AMD K6) и имеют следующий синтаксис:

```
instruction [src, dest]
```

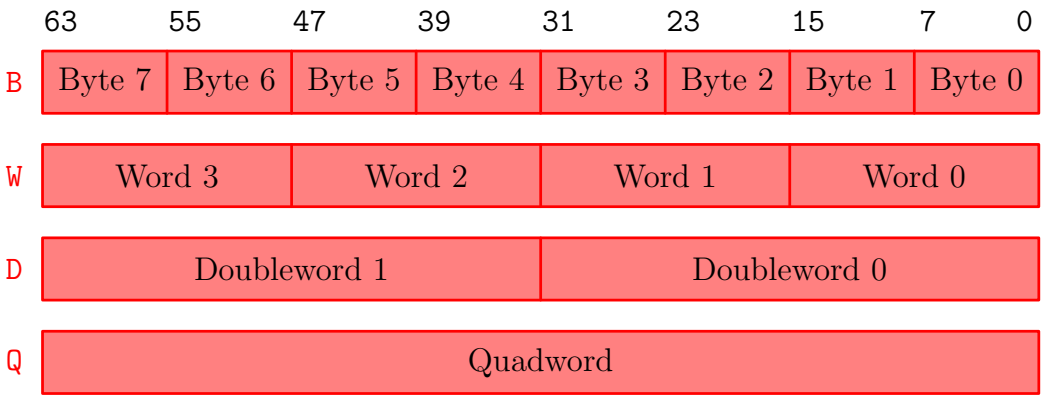


Рис. 11: Упакованные и неупакованные форматы данных MMX

Здесь `instruction` — имя команды, `dest` — выходной, а `src` — входной операнды.

Большинство команд имеют суффикс, который определяет тип данных и используемую арифметику:

- `US` (Unsigned Saturation) — арифметика с насыщением, данные без знака;
- `S` или `SS` (Signed Saturation) — арифметика с насыщением, данные со знаком. Если в суффиксе нет ни `S`, ни `SS`, используется циклическая арифметика (Wrap Around);
- `B`, `W`, `D`, `Q` указывают тип данных. Если в суффиксе есть две из этих букв, первая соответствует входному операнду, а вторая — выходному.

Возьмем в качестве примера единицу звуковой информации — 16-разрядный элемент аудиотрека. Четыре элемента можно разместить в 64-разрядном регистре MMX. При выполнении «упакованной» команды из регистров `i` и `j` параллельно выбирается по элементу, над ними одновременно выполняется арифметическая или логическая операция (например, «+»), и результат записывается в регистр `k`. На рис. 12 показана MMX-команда `paddusw`, выполняющая упакованное сложение беззнаковых слов с насыщением).

5.4.2 3DNow!

Архитектура 3DNow! впервые реализована в процессорах AMD K6-2 (май 1998 г.). Технология 3DNow! включает 21 дополнительную команду, новые типы данных и использует регистры MMX (`MM0` – `MM7`) для поддержки высокопроизводительной обработки 3D-графики и звука.

В то время как архитектура MMX предполагает целочисленную арифметику, векторные команды 3DNow! параллельно обрабатывают две пары 32-разрядных вещественных операндов одинарной точности.

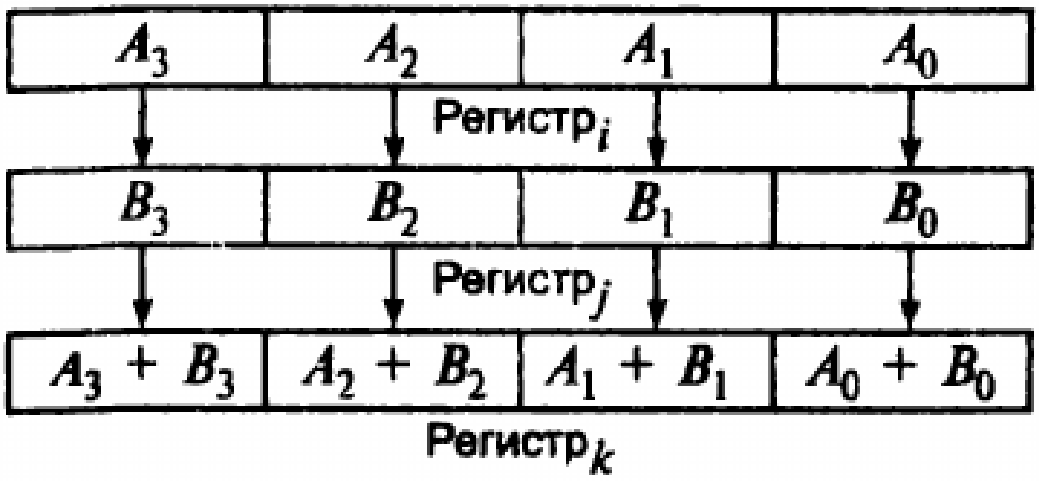


Рис. 12: Команда `paddusw` — пример векторной операции над упакованными словами

Процессор может выполнять две 3DNow! команды за такт и, следовательно, 4 операции над числами с плавающей точкой одновременно. Технология 3DNow! предполагает наличие в ЦП для выполнения векторных MMX и 3DNow! операций нескольких устройств (пара умножителей, сумматоров и т. д.). Все команды 3DNow! имеют длительность исполнения 2 такта и полностью конвейеризированы.

5.4.3 SSE

SSE (или SIMD-FP) — система команд Streaming SIMD Extensions — SIMD-расширение, предложенное Intel в 1999 г. в Pentium III (ядро Katmai), отсюда вариант названия — KNI (Katmai New Instructions). Это 70 новых команд, в числе которых выделяются следующие.

- 50 команд предназначаются для повышения эффективности операций над числами с плавающей точкой. С этой целью в ЦП встроены 128-битовые регистры — восемь регистров, названные `XMM0` – `XMMM7` (в AMD64 число SSE/XMM регистров было увеличено от 8 до 16). В результате операции с плавающей точкой могут совершаться за один цикл процессора.

- 12 команд (New Media) дополняют ранее введенные 57 команд MMX для чисел с фиксированной точкой.

- оставшиеся 8 команд (New Cacheability) повышают производительность кэш-памяти L1 при работе с мультимедийными данными.

SSE — набор команд, которые обрабатывают только значения с плавающей точкой, подобно 3DNow!. Поскольку здесь используются более длинные регистры, чем в 3DNow!, SSE может упаковать четыре числа с плавающей точкой в каждый регистр. Первоначальная версия SSE была ограничена только числами одинарной точности, подобно 3DNow!.

5.4.4 SSE2

SSE2 — введенный с Pentium IV набор команд является существенным развитием SSE, оперирует с теми же самыми регистрами и обратно совместим с SSE процессора Pentium III. В расширении SSE2 операции со 128-битовыми регистрами могут выполняться не только как с четвёрками вещественных чисел одинарной точности, но и как с парами вещественных чисел двойной точности, с шестнадцатью однобайтовыми целыми и пр. SSE2 представляет собой симбиоз MMX и SSE, и позволяет работать с любыми типами данных, вмещающимися в 128-битовые регистры.

5.4.5 SSE3

SSE3 — набор команд, также известный как Prescott New Instructions (PNI), является третьей версией команд SSE для IA-32. Intel использует SSE3 с начала 2004 г. в ЦП Pentium IV Prescott. В апреле 2005 г. AMD также включает SSE3 в ЦП Athlon 64 (версия E — ядра Venice и San Diego). SSE3 содержит 13 дополнительных по отношению к SSE2 команд. Самое существенное новшество — «горизонтальная арифметика».

В частности, были добавлены команды сложения/вычитания по отношению к наборам (векторам) значений, расположенных в пределах единственного регистра (см. рис. 13), которые улучшают эффективность цифровой обработки сигналов и трёхмерных преобразований.

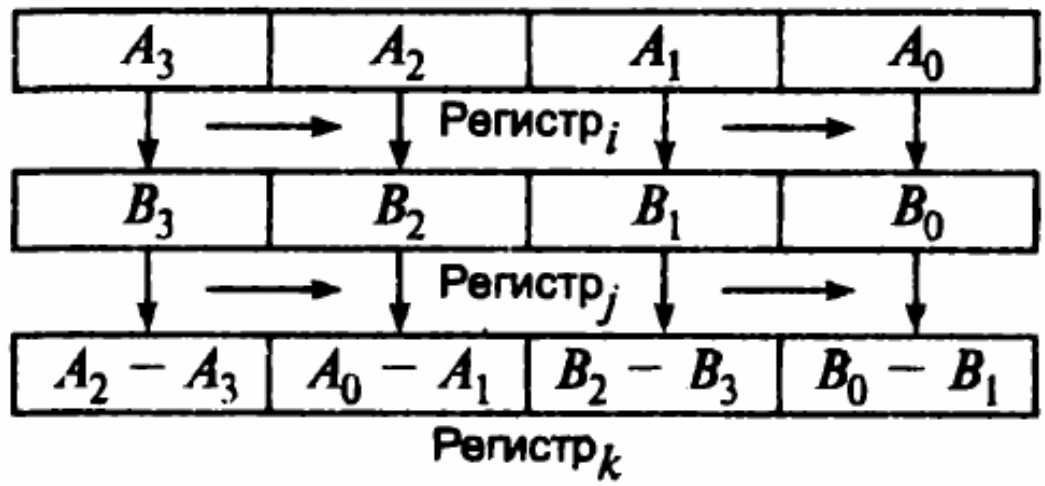


Рис. 13: SSE3-инструкция `hsubps` — Horizontal-SUBtract-Packed-Single — горизонтальное вычитание упакованных слов одинарной длины

Кроме рассмотренных здесь арифметических операций, в системах команд компьютеров присутствуют также и логические, а также сдвиговые операции.

5.5 Динамическое исполнение

Динамическое исполнение (dynamic execution technology) — технология обработки данных процессором, обеспечивающая более эффективную работу процессора за счет манипулирования данными, а не просто линейного исполнения списка инструкций.

5.5.1 Предсказание ветвлений

С большой точностью (более 90%) процессор предсказывает, в какой области памяти можно найти следующие инструкции. Это оказывается возможным, поскольку в процессе исполнения инструкции процессор просматривает программу на несколько шагов вперед.

Это обеспечивает значительное повышение производительности. Например, программный цикл, состоящий из пересылки, сравнения, сложения и перехода в 80486 DX, выполняется за шесть тактов синхронизации, а в Pentium — за два (команды пересылки и сложения, а также сравнения и перехода сочетаются и предсказывается переход).

5.5.2 Внеочередное выполнение

Согласно внеочередному выполнению (out-of-order execution) процессор анализирует поток команд и составляет график исполнения инструкций в оптимальной последовательности, независимо от порядка их следования в тексте программы. Для этого процессор просматривает декодированные инструкции и определяет, готовы ли они к непосредственному исполнению или зависят от результата других инструкций. Далее, процессор определяет оптимальную последовательность выполнения и исполняет инструкции наиболее эффективным образом.

5.5.3 Переименование (ротация) регистров (register rename)

Чтобы избежать пересылок данных между регистрами в соответствующей команде изменяется адрес регистра, содержащего данные, участвующие в следующей операции. Поэтому вместо пересылки данных в регистр-источник осуществляется трактовка регистра с данными как источника.

5.5.4 Выполнение по предположению

Процессор выполняет инструкции (до пяти инструкций одновременно) по мере их поступления в оптимизированной последовательности. Поскольку выполнение инструкций происходит на основе предсказания ветвлений, результаты сохраняются как предположительные («спекулятивные»). На конечном этапе порядок инструкций восстанавливается.

На рис. 14 и 15 представлены варианты спекулятивного выполнения:

- предикация (predication) — одновременное исполнение нескольких ветвей программы вместо предсказания переходов (выполнения наиболее вероятного);
- опережающее чтение данных (speculative loading), т. е. загрузка данных в регистры с опережением, до того, как определилось реальное ветвление программы (переход управления).

Эти возможности осуществляются комбинированно — при компиляции и выполнении программы.

Предикация. Обычный компилятор транслирует оператор ветвления (например, `if-then-else`) в блоки машинного кода, расположенные последовательно в потоке. Обычный процессор, в зависимости от исхода условия, исполняет один из этих базовых блоков, пропуская все другие. Более развитые процессоры пытаются прогнозировать исход операции и предварительно выполняют предсказанный блок. При этом в случае ошибки много тактов тратится впустую. Сами блоки зачастую весьма малы — две или три команды, — а ветвления встречаются в коде в среднем каждые шесть команд. Такая структура кода делает крайне сложным его параллельное выполнение.

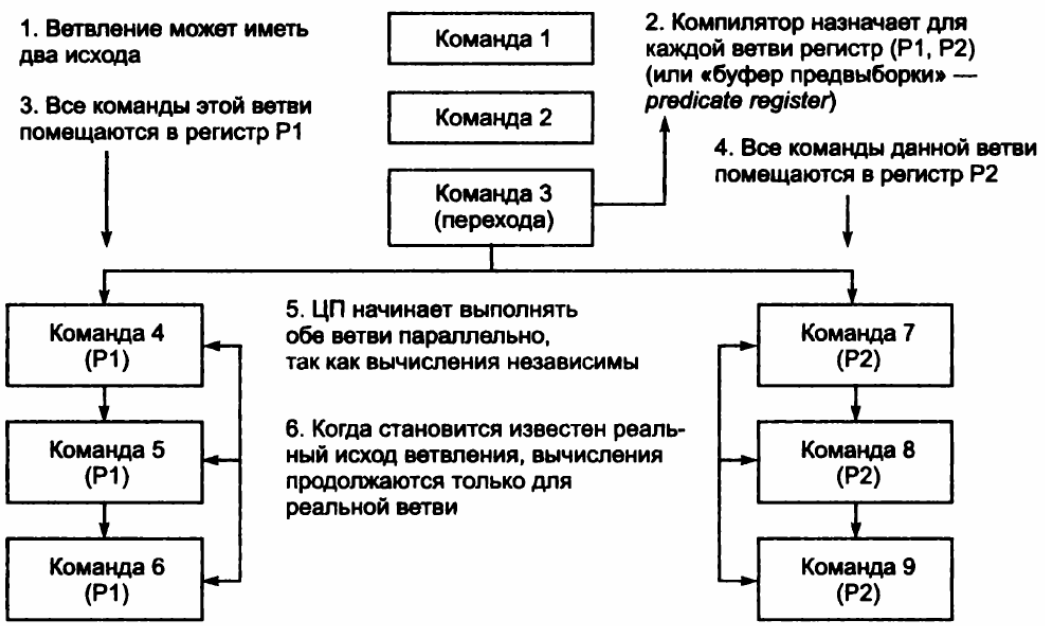


Рис. 14: Предикация всех альтернативных ветвей оператора условного перехода

При использовании предикации компилятор, обнаружив оператор ветвления в исходной программе, анализирует все возможные ветви (блоки) и помечает их метками или предикатами (predicate). После этого он определяет, какие из них могут быть выполнены параллельно (из соседних, независимых ветвей).

В процессе выполнения программы ЦП выбирает команды, которые взаимно независимы и распределяет их на параллельную обработку. Если ЦП обнаруживает

оператор ветвления, он не пытается предсказать переход, а начинает выполнять все возможные ветви программы.

Таким образом, могут быть обработаны все ветви программы, но без записи полученного результата. В определенный момент процессор наконец «узнает» о реальном исходе условного оператора, записывает в память результат «правильной ветви» и отменяет остальные результаты.

В то же время, если компилятор не «отметил» ветвление, процессор действует как обычно — пытается предсказать путь ветвления и т. д. Испытания показали, что описанная технология позволяет устранить более половины ветвлений в типичной программе, и, следовательно, уменьшить более чем в два раза число возможных ошибок в предсказаниях.

Опережающее чтение. Другие названия техники — предварительная загрузка данных или чтение по предположению. Идея состоит в разделении этапов загрузки данных в регистры и их реального использования. Это позволяет избегать ситуации ожидания процессором поступления данных для начала их обработки.

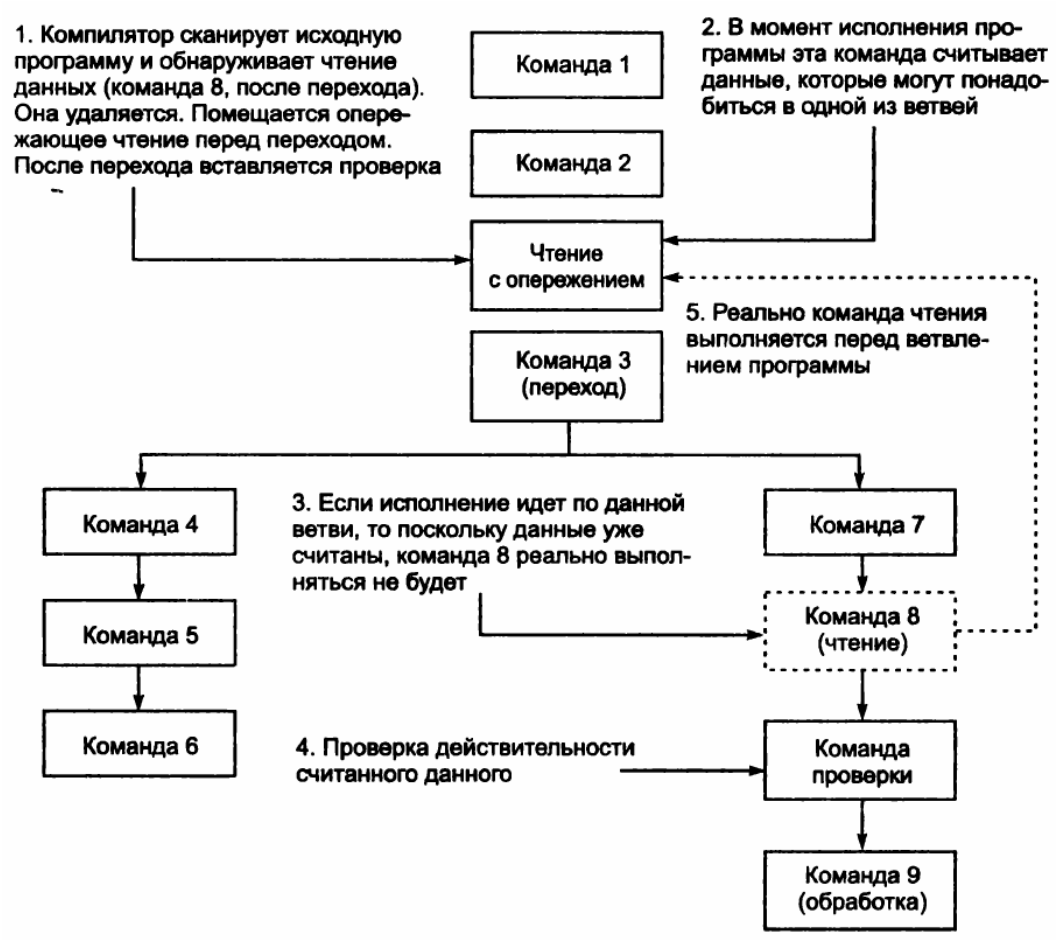


Рис. 15: Опережающее считывание данных в регистры ЦП из памяти (speculative loading)

Прежде всего, компилятор анализирует программу, определяя команды, которые требуют приёма данных из оперативной памяти. Там, где это возможно, он вставляет

команды опережающего чтения и парную команду контроля опережающего чтения (speculative check). В то же время компилятор переставляет команды таким образом, чтобы ЦП мог их обрабатывать параллельно.

В процессе работы ЦП встречает команду опережающего чтения и пытается выбрать данные из памяти. Может оказаться, что они еще не готовы (результат работы блока команд, который ещё не выполнился). Обычный процессор в этой ситуации выдает сообщение об ошибке, однако система откладывает «сигнал тревоги» до момента прихода процесса в точку «команда проверки опережающего чтения». Если к этому моменту все предшествующие подпроцессы завершены и данные считаны, то обработка продолжается. В противном случае вырабатывается сигнал прерывания.

Возможность располагать команду предварительной загрузки до ветвления очень существенна, так как позволяет загружать данные задолго до момента использования (напомним, что в среднем каждая шестая команда является командой ветвления).

5.6 Многократное декодирование команд

В то время как традиционный процессор линейно переводит команды в тактовые микрокоманды и последовательно их выполняет, ЦП с многократным декодированием сначала преобразует коды исходных команд программы в некоторые вторичные псевдокоды (предварительное декодирование, или предекодирование), которые затем более эффективно исполняет ядро процессора. Эти преобразования могут содержать несколько этапов. В качестве примеров рассмотрим 2 – 3-ступенчатые декодеры (рис. 16).

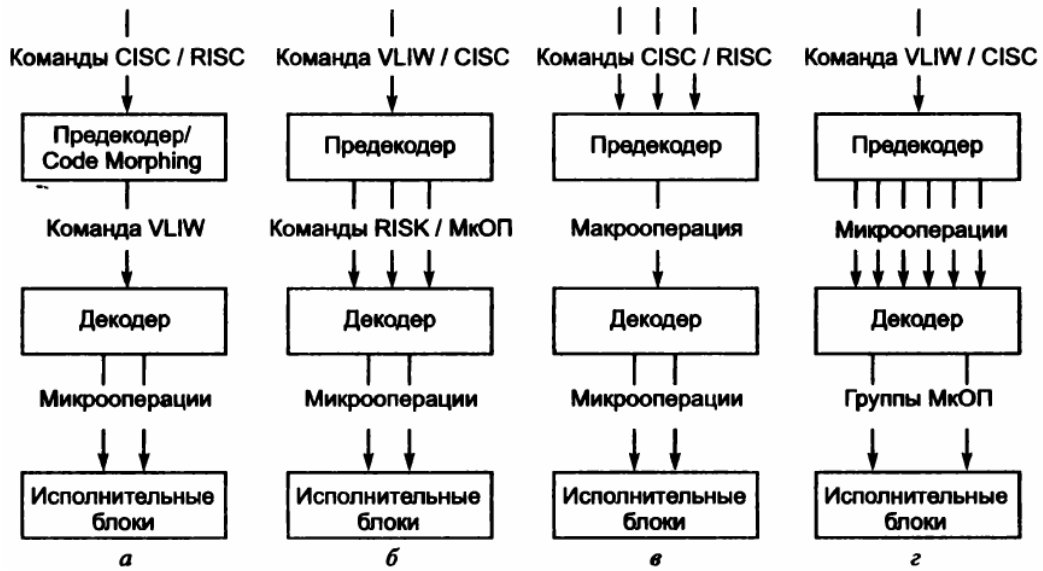


Рис. 16: Многократные декодеры и смежные технологии: а — преобразование CISC/RISC в VLIW; б — преобразование VLIW/CISC в RISC; в — макрослияние; г — микрослияние

5.6.1 Декодирование команд CISC/RISC в VLIW

.

Эти технологии (рис. 16, а) использованы в мобильных процессорах Crusoe (фирма Transmeta) и некоторых ЦП Intel — архитектуры IA-64 и EPIC (Explicitly Parallel Instruction Computing — вычисления с явной параллельностью инструкций).

В частности, в Crusoe на входе процессора — программы, подготовленные в системе команд Intel x86, однако внутренняя система команд VLIW не имеет ничего общего с командами x86 и разработана для быстрого выполнения при малой мощности, используя обычную CMOS-технологию. Окружающий уровень программного называют программным обеспечением модификации кодов (Code Morphing software — CMS, или CM), здесь осуществляется динамический перевод команд x86 в команды VLIW.

Программное обеспечение модификации кодов переводит сразу блоки команд x86, записывая результат в кэш перевода. Этот подход к выполнению кода x86 устраняет миллионы транзисторов, заменяя их программным обеспечением. Процессор Crusoe содержит не более 1/4 числа логических транзисторов, требуемых для достижения подобной производительности только аппаратными средствами, и имеет следующие преимущества:

- аппаратный компонент значительно меньше, быстрее и энергетически эффективнее, чем обычные микросхемы;
- аппаратные средства полностью отделены от команд x86, давая возможность использовать в проектах наиболее современные и эффективные тенденции в проектировании электроники, не загружая программное обеспечение проблемами совместимости;
- программное обеспечение CM может развиваться отдельно от аппаратных средств, и обновление программной части процессора может быть выполнено независимо от аппаратных версий процессора;
- упрощение аппаратуры позволило уменьшить габариты процессоров и потребление энергии (эти процессоры иногда называют «холодными»).

Аналогичные приемы используются в процессорах Intel Itanium — здесь при компиляции готовятся пакеты (связки, bundles) команд (по 3 команды в 128-битовом пакете). Тем самым, компилятор выполняет в данном случае функции CM.

5.6.2 Декодирование команд CISC/VLIW в RISC

Указанные выше достоинства RISC-архитектуры привели к тому, что во многих современных CISC-процессорах используется RISC-ядро, выполняющее обработку

данных. При этом поступающие сложные и разноформатные команды предварительно преобразуются в последовательность простых RISC-операций, быстро выполняемых этим процессорным ядром (рис. 16, б). Таким образом, работают, например, современные модели процессоров семейства x86-64, которые по внешним показателям относятся к CISC-процессорам. Использование RISC-архитектуры является характерной чертой многих современных процессоров.

5.6.3 Макрослияние (macrofusion)

В процессорах предыдущих поколений каждая выбранная команда отдельно декодируется и выполняется. Макрослияние позволяет объединять типичные пары последовательных команд (например, сравнение, сопровождающееся условным переходом) в единственную внутреннюю команду-микрооперацию (μOp, micro-op) в процессе декодирования (рис. 16, в). В дальнейшем две команды выполняются как одна μOp, сокращая полный объём работы процессора.

5.6.4 Микрослияние (micro-op fusion)

В современных массовых процессорах команды x86 (macro-ops) обычно расчленяются на μOp прежде, чем передаются на конвейер процессора. Микрослияние группирует и соединяет микрооперации, уменьшая их число (рис. 16, г). Исследования показали, что слияние μOps вкупе с выполнением команд в измененном порядке может уменьшить число μOps более чем на 10%. Данная технология использована в системах Intel Core, а ранее апробировалась в ЦП мобильных систем Pentium M.

В процессорах AMD K8 конвейер строится на том, что работа с потоком μOps происходит тройками инструкций (AMD называет их линиями — line). Конвейер K8 обрабатывает именно линии, а не x86-инструкции или отдельные микрооперации.

5.7 Технология Hyper-Threading (HT)

Здесь реализуется разделение времени на аппаратном уровне, разбивая физический процессор на два логических процессора, — каждый из которых использует основные ресурсы микросхемы процессора — ядро, кэш-память, шины, исполнительное устройство (рис. 17). Благодаря HT многопроцессная операционная система использует один физический процессор как два логических, и выдает одновременно два потока команд. Смысл технологии заключается в том, что в большинстве случаев исполнительные устройства процессора далеки от полной загруженности. От передачи на выполнение вдвое большего потока команд повышается загрузка исполнительных устройств.

Специалисты Intel оценивают повышение эффективности в 30% HT-процессоров при использовании многопрограммных ОС и обычных прикладных программ.

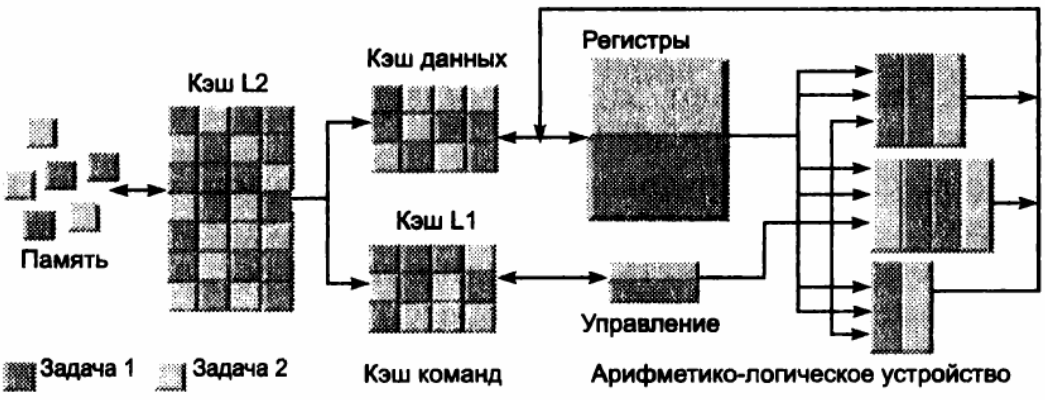


Рис. 17: Технология Hyper-Threading (HT)

5.8 Многоядерные процессоры

В октябре 1989 г., рассматривая будущее «через призму закона Мура», специалисты Intel в статье «Microprocessors Circa 2000» предсказали, что многоядерные процессоры могут выйти на рынок вскоре после начала столетия. Пятнадцать лет спустя их предсказания оправдались — развитие процессоров в этом направлении стало приоритетной задачей как для Intel, так и для конкурирующей AMD.

Многоядерная архитектура предполагает размещение двух или более основных вычислительных агрегатов в пределах единственной микросхемы процессора. Этот многоядерный процессор имеет единственный интерфейс с системной платой, но операционные системы «видят» каждое из его ядер как дискретный логический процессор со всеми связанными ресурсами. Это отличает их от технологии Hyper-Threading (где отдельные процессы выполняются единственным ядром), и существующие ресурсы используются более эффективно.

Разделяя вычислительную нагрузку, выполняемую единственным ядром в традиционных процессорах между многими ядрами, многоядерный процессор может выполнить большую работу в пределах отдельного цикла ЭВМ. Чтобы реализовать это увеличение эффективности, соответствующее программное обеспечение должно поддерживать это распараллеливание. Эти функциональные возможности называют «параллелизмом уровня подпроцесса (нити)» или «threading». Приложения и операционные системы, которые поддерживают это, упоминаются как мультиподпроцессные или «multi-threaded».

Процессор, оборудованный параллелизмом уровня подпроцесса, может выполнить полностью отдельные «нити» кода. Это может означать или один подпроцесс, выполняемый из приложения, и второй подпроцесс, выполняемый из операционной системы, или параллельные подпроцессы, выполняемые из единственного приложения. Параллелизм уровня подпроцесса приносит особенный выигрыш для многих мультимедийных приложений, потому что многие из их операций способны к выполнению параллельно.

Если при этом используется режим Hyper-Threading, двухядерные процессоры

способны обрабатывать четыре программных подпроцесса одновременно более эффективно имеющимися ресурсами, которые иначе, возможно, остались бы незанятыми.

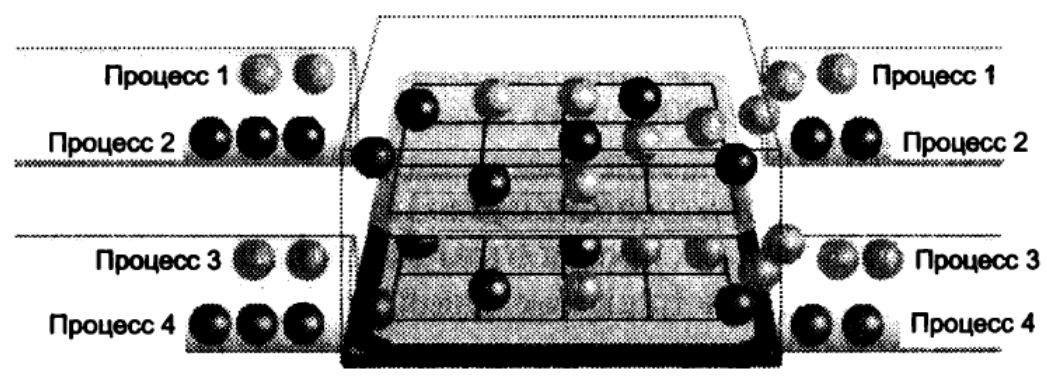


Рис. 18: Выполнение процессов Hyper-Threading на двухъядерном процессоре