Общее устройство центрального процессора, принципы его проектирования, конвейеризация

Луцив Дмитрий Вадимович

Кафедра системного программирования СПбГУ

Содержание

Блоки процессора

Арифметико-логическое устройство

Арифметико-логическое устройство — блок процессора, выполняющий арифметические и логические операции

- В простейшем случае — просто логическая схема
- Может использоваться для прикладных (например, вычисления, заданные программистом) и служебных (например, адресная арифметика)
- Имеет несколько входов для операндов и выход, на который *мультиплексируются* выходы сумматора, мультипликатора и т.д.

Регистровый файл

Регистровый файл — блок процессора, включающий набор регистров; внутренняя память процессора

- Арифметические регистры
  - Аккумулятор
  - Ещё несколько [десятков] регистров
- Регистры состояний
  - Регистр флагов
- Адресные регистры
  - Указатель вершины стека
  - Указатель на текущую инструкцию
  - Указатель на данные [часто несколько]
  - Вспомогательные (например, сегментные)
- Служебные регистры
  - Хранение промежуточных значений, реализация протоколов (например, с ОЗУ) и т.д.

Блок выборки инструкций (устройство чтения программы)

Блок выборки инструкций — блок процессора, выполняющий чтение очередных инструкций из памяти

- Читает из ОЗУ машинный код
  - Выполняет первичную интерпретацию машинного кода

Для CISC-процессоров со сложным машинным кодом это не так-то просто!

Блок выборки инструкций (устройство чтения программы)

Блок выборки инструкций — блок процессора, выполняющий чтение очередных инструкций из памяти

- Читает из ОЗУ машинный код
  - Выполняет первичную интерпретацию машинного кода

Для CISC-процессоров со сложным машинным кодом это не так-то просто!

О том, что такое RISC и CISC, ещё немного позже.

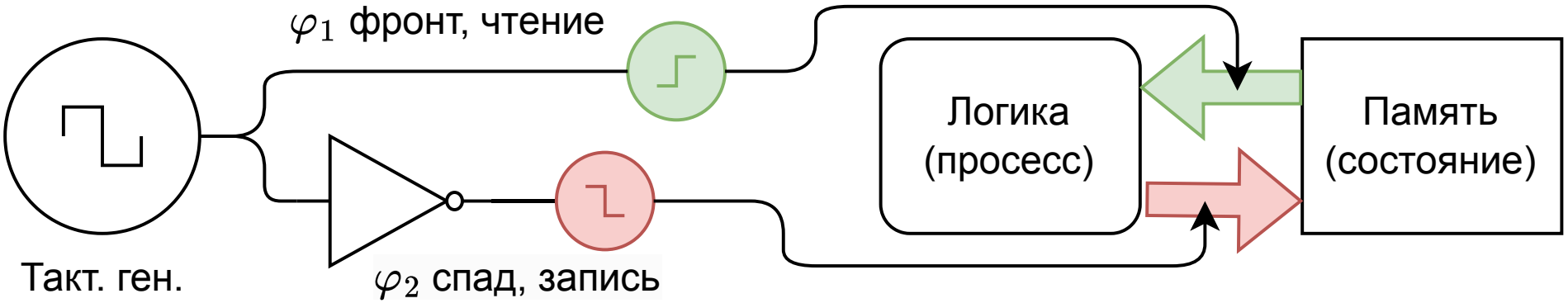
Управляющее устройство

Управляющее устройство — блок процессора, выдающий для исполнения машинных команд управляющие сигналы другим блокам

- В зависимости от команды, выдаёт управляющие сигналы другим блокам процессора
  - Сигналы преимущественно управляют мультиплексорами, т.е. задают *маршруты передачи данных между блоками процессора*
- Если команда выполняется за много тактов (а обычно так и есть), выдаёт не один сигнал, а *последовательность сигналов* разным блокам

Одно-, многотактный процессоры

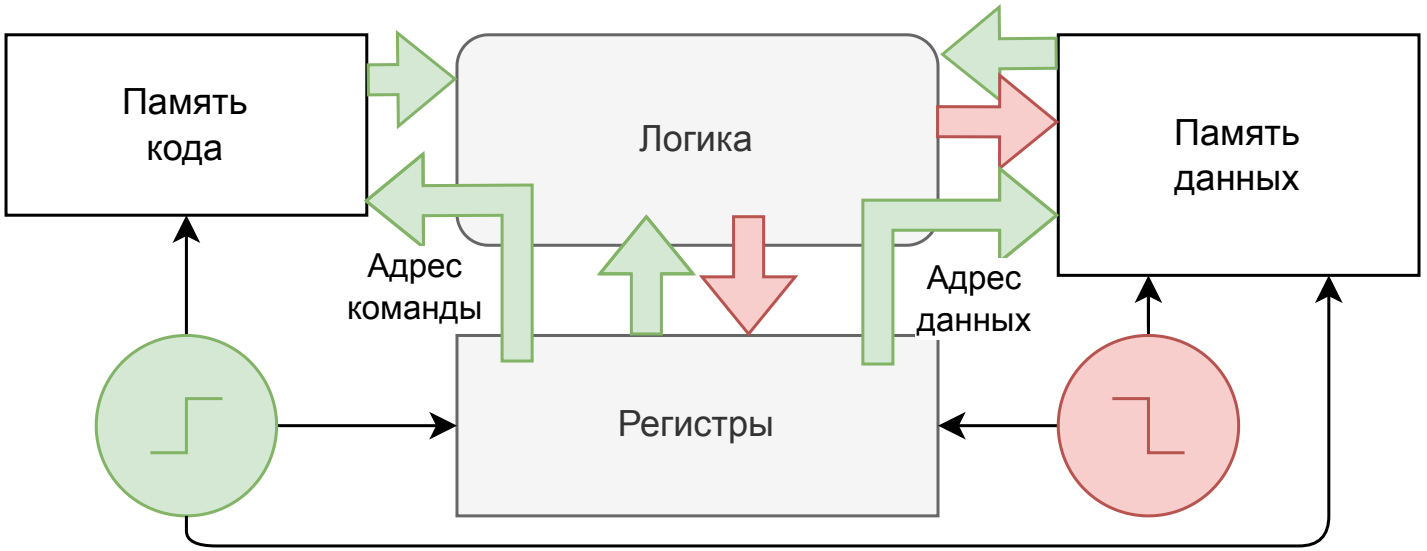
Однотактная схема



Вспоминаем:

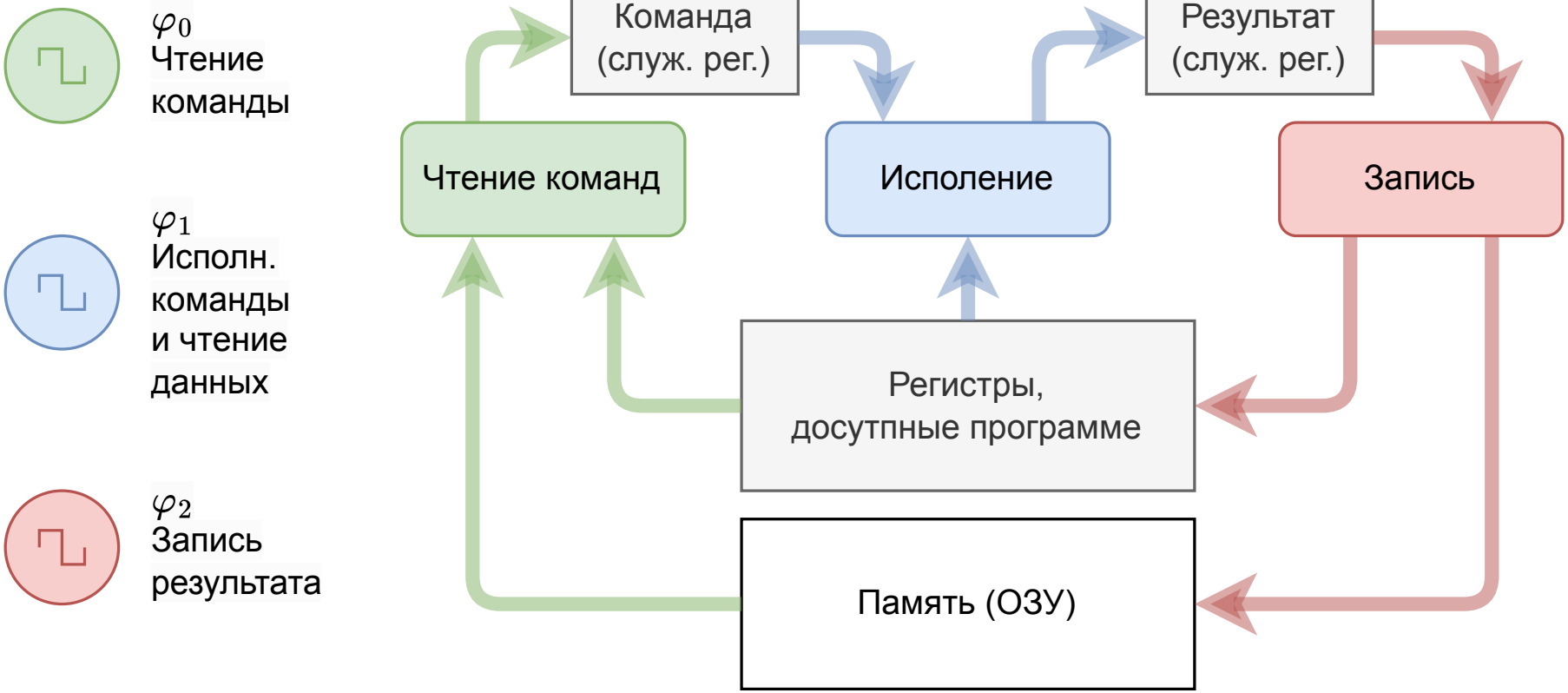
1. Синхронные и асинхронные вычисления, тактовый генератор, тактовые импульсы и тактовую сеть
2. Синхронизацию по фронту и спаду тактового импульса
3. Связку «master-slave»
4. Многофазные тактовые сигналы у старых процессоров

Однотактный процессор



- В принципе, он даже может работать
  - Можно «нарисовать», например, несложный контроллер с такой архитектурой
- Только гарвардский, т.к. нельзя одновременно обращаться и в одну память за кодом и данными

Многотактный процессор (упрощённый пример для 3 тактов)



Как оно работает, и хорошо ли?

Как?

- Используются три фазы — $\varphi_0, \varphi_1, \varphi_2$
  - Эти фазы может генерировать процессор внутри себя по сигналам обычного тактового генератора
- Простая (и не очень) логика реализуется на основе *конечных автоматов* (удобный абстрактный вычислитель, помогающий решать конкретные задачи =))
- Сложная логика реализуется при помощи микрокода — в -1 приближении — последовательности (работает много тактов) управляющих сигналов для мультиплексоров на входах блоков процессора

Как оно работает, и хорошо ли?

Как?

- Используются три фазы — $\varphi_0, \varphi_1, \varphi_2$
  - Эти фазы может генерировать процессор внутри себя по сигналам обычного тактового генератора
- Простая (и не очень) логика реализуется на основе *конечных автоматов* (удобный абстрактный вычислитель, помогающий решать конкретные задачи =))
- Сложная логика реализуется при помощи микрокода — в -1 приближении — последовательности (работает много тактов) управляющих сигналов для мультиплексоров на входах блоков процессора

Хорошо ли?

- Разные фазы выполняются разными блоками по очереди
  - $\frac{2}{3}$ времени блоки процессора простаивают!

Конвейеризация

- Конвейеры в реальной жизни
- Вычислительные конвейеры

Конвейеризация «в жизни»

Предпосылки

Эли Уитни, 1798 (оружейное производство) — одновременное изготовление стандартизованных узлов мушкета разными рабочими, затем быстрая сборка готовых изделий

Реальные конвейеры

- Генри Форд, 1913 и т.д. — сборка электрогенераторов, затем моторов и целых автомобилей
  - Разные этапы производства выполняются разными рабочими
  - Рабочие работают одновременно, каждый на своём этапе
- Конвейер в системе образования
  - Школа: 1–11 классы — выпуск каждый год, но школьник учится 11 лет
  - Высшее образование: *старая* поговорка: Матмех — не школа, за *10 лет* не окончишь =)
- Конвейер в менеджменте
  - Конвейерное исполнение заказов

Недостатки конвейера

- Низкая гибкость — организованный и запущенный конвейер тяжело приспособить к изменяющейся ситуации
- Снижение качества результата в угоду массовости
  - Пример: товар есть на складе в моём городе, но почему-то едет с другого конца страны

Недостатки конвейера

- Низкая гибкость — организованный и запущенный конвейер тяжело приспособить к изменяющейся ситуации
- Снижение качества результата в угоду массовости
  - Пример: товар есть на складе в моём городе, но почему-то едет с другого конца страны

Выход: в менеджменте это преодолевается переходом от конвейера к организации бизнес-процессов — сложнее, но адаптивнее

Что такое конвейер?..

Вычислительный конвейер (водопровод, pipeline) — механизм распараллеливания выполнения машинных команд, позволяющий оптимально задействовать блоки процессора путём разбиения команд на стадии и распределения стадий по блокам

Конвейеры появились в 1950-х годах, термин «конвейер» (pipeline) ввёл конструктор советских ЭВМ С.А. Лебедев.

Пусть все команды выполняются за N тактов. Тогда, что даёт конвейеризация?

- Скорость выполнения отдельных команд:
  - Каждая команда выполняется за N тактов как с конвейером, так и без
  - В отношении одной команды на разных тактах задействованы разные блоки
- Скорость выполнения программы:
  - На каждом такте завершается очередная команда
  - Конвейеризация ускоряет работу программы в N раз, все блоки задействованы всё время

Что такое конвейер?..

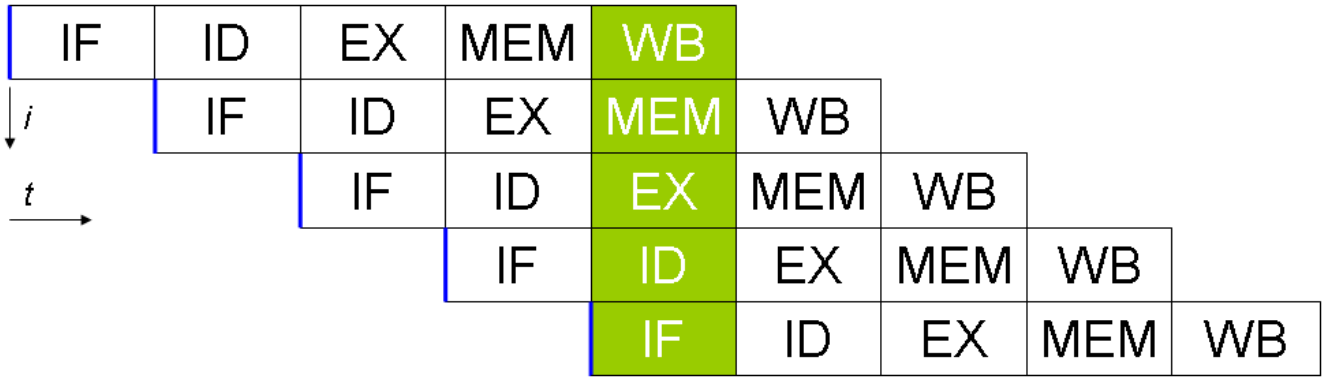
Вычислительный конвейер (водопровод, pipeline) — механизм распараллеливания выполнения машинных команд, позволяющий оптимально задействовать блоки процессора путём разбиения команд на стадии и распределения стадий по блокам

Конвейеры появились в 1950-х годах, термин «конвейер» (pipeline) ввёл конструктор советских ЭВМ С.А. Лебедев.

Пусть все команды выполняются за N тактов. Тогда, что даёт конвейеризация?

- Скорость выполнения отдельных команд:
  - Каждая команда выполняется за N тактов как с конвейером, так и без
  - В отношении одной команды на разных тактах задействованы разные блоки
- Скорость выполнения программы:
  - На каждом такте завершается очередная команда
  - Конвейеризация ускоряет работу программы в N раз, все блоки задействованы всё время
  - Косвенная выгода: «сосредоточение» отдельных стадий в компактных блоках позволяет увеличить тактовую частоту

Пример: Classic RISC Pipeline



Classic RISC Pipeline, Wikipedia

Применялся к популярным RISC-процессорам 1980-х: первым MIPS, Sun SPARC, Motorla 88000. Стадии:

1. IF (instruction fetch) — получение инструкции
2. ID (instruction decode) — декодирование инструкции
3. EX (execute) — выполнение инструкции, если нужно — первый такт доступа к данным в ОЗУ
4. MEM memory access) — второй такт доступа к данным в ОЗУ или бездействие
5. WB (register write back) — запись в регистр

Проблемы конвейеризации

1. Конфликты по данным между зависимыми машинными командами, например:
   - самая частая ситуация — следующей команде нужен результат предыдущей, но предыдущая ещё его не получила
   - следующая команда записывает данные до того, как их читает предыдущая
   - следующая команда записывает свои результаты раньше предыдущей из-за чего предыдущая потом может их перезаписать
2. Конфликты по ресурсам — когда двум командам одновременно нужен эксклюзивный доступ к чему-либо (регистр, шина...)
3. Конфликты по управлению: следующая команда является условным переходом, но условие для него ещё не вычислено предыдущей командой
   - и даже не ясно, из какой ветви дальше выбирать команды

Проблемы конвейеризации

1. Конфликты по данным между зависимыми машинными командами, например:
   - самая частая ситуация — следующей команде нужен результат предыдущей, но предыдущая ещё его не получила
   - следующая команда записывает данные до того, как их читает предыдущая
   - следующая команда записывает свои результаты раньше предыдущей из-за чего предыдущая потом может их перезаписать
2. Конфликты по ресурсам — когда двум командам одновременно нужен эксклюзивный доступ к чему-либо (регистр, шина...)
3. Конфликты по управлению: следующая команда является условным переходом, но условие для него ещё не вычислено предыдущей командой
   - и даже не ясно, из какой ветви дальше выбирать команды
     Спойлер: некоторые RISC процессоры, например ранние MIPS, исполняли до двух команд даже после *безусловного* перехода: конвейер успевал из «засосать» из памяти, не успевая ещё разобрать, что в них

Решение проблем конвейеризации

- Pipeline Stall — торможение команд на конвейере; в предельном случае может свести преимущества конвейеризации на нет
- Предсказание условных переходов — сбор статистики о переходах или подсказки от компилятора

CISC- и RISC процессоры

- Основы
- Особенности конвейеризации

Пример программы и её трансляции (1: x86)

```c
int factorial(int n)
{
    int r = 1;
    while(n > 1)
    r *= n--;
    return r;
}
```

Примеры компиляции и кросс-компиляции для x86, ARMv9, MIPS64, RISC-V

Что отличало данные примеры?

Система команд — соглашение о средствах, предоставляемых машинным языком и о структуре машинного языка

Орлов С.А., Цилькер Б.Я. Организация ЭВМ и систем: Учебник для вузов. 2-е изд. — СПб.: Питер, 2011. — 688 с.

CISC

CISC — Complicated Instruction Set Computer

- Много способов адресовать аргументы команд
- Смешанная адресация — разные операнды из разных мест
- Команды похожи на операторы языков высокого уровня
- Ассемблер дружественный
- Небольшое количество сильно умных команд
- Реализованы при помощи микропрограмм
- Машинный код экономный в смысле занимаемого места в памяти (многие «популярные» команды занимают 1 байт)

Примеры:
IBM/360..z-Architecture, Intel x86 и x86_64, Intel 8080 и Zilog Z80

RISC

RISC — Reduced Instruction Set Computer

- Отдельные команды для обмена данными с памятью
- Команды простые
  - Команды выполняются за фиксированное время (фиксированное количество тактов)
  - Это упрощает конвейеризацию
- Некоторые «долгие» команды выполняются асинхронно
- Ассемблер недружественный
  - Помните про «глупый всеядный» конвейер?
- Машинный код не экономный, все команды занимают одно машинное слово
  - В некоторых режимах половину (для RISC-V попробуйте `-march=rv64g` vs `-march=rv64gc`)

Примеры:

Cray CDC 6600 (1960-е), и большинство новых семейств, начиная с 1980-х: MIPS, SPARC, ARM, RISC-V

CISC vs RISC: недружественный ассемблер — это очень страшно?

CISC vs RISC: недружественный ассемблер — это очень страшно?

Нет.

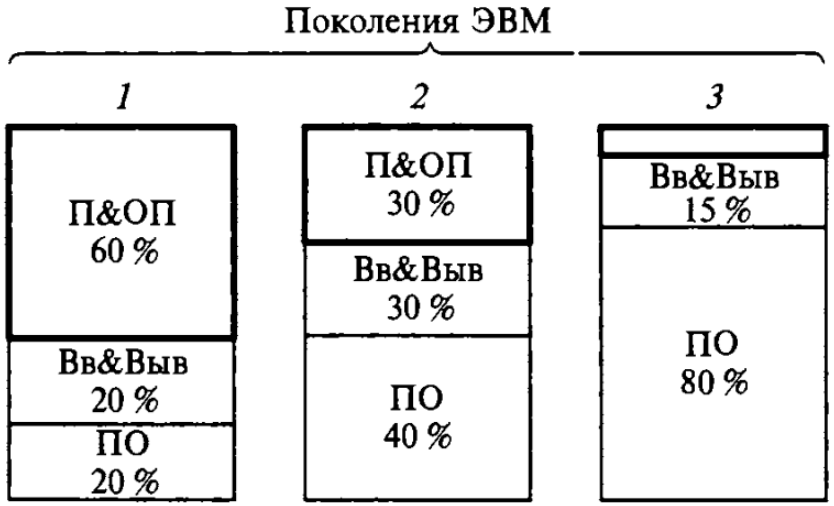


Рис. 2.3. Распределение стоимости между компонентами ЭВМ:

П&ОП — процессор и оперативная память; Вв&Выв — устройство ввода-вывода информации; ПО — программное обеспечение

Архитектура вычислительных систем.: Учеб. пособие. 2-е изд., перераб. и доп. М.: Изд-во МГТУ им. Н.Э. Баумана, 2008. 520 с.

- Cray CDC 6600 — суперкомпьютер середины 1960-х, можно было программировать дорого, т.к. он и сам был очень дорогой
- Большинство — 1980-е и позже, программируются уже на языках высокого уровня

Конвейеризация в RISC

Особенности и трудности

У RISC простой формат машинного кода, простой блок выборки инструкций, простой конвейер. Иногда есть несколько «долгих» команд, которые выполняются асинхронно

Некоторые RISC-семейства не отслеживают конфликты и не обрабатывают их!

Конвейеризация в RISC

Особенности и трудности

У RISC простой формат машинного кода, простой блок выборки инструкций, простой конвейер. Иногда есть несколько «долгих» команд, которые выполняются асинхронно

Некоторые RISC-семейства не отслеживают конфликты и не обрабатывают их!

Решения

- Переупорядочивание команд — конфликтующие команды размещаются «на безопасном расстоянии» друг от друга
- Торможение при помощи вставки «пузырька» (инструкция nop, no operation)

Для некоторых RISC-семейств всё это делает компилятор или программист!

Конвейеризация в CISC

Процессор полностью сам отвечает за корректное исполнение кода — обнаруживает конфликты и обрабатывает их.

Процессор с конвейером должен (кроме скорости) работать так же, как и без конвейера.

А мы же говорили, что конвейер не гибкий?..

- Суперскалярность
- Very Long Instructoin Word
- Внеочередное исполнение

Суперскалярное исполнение: сущность

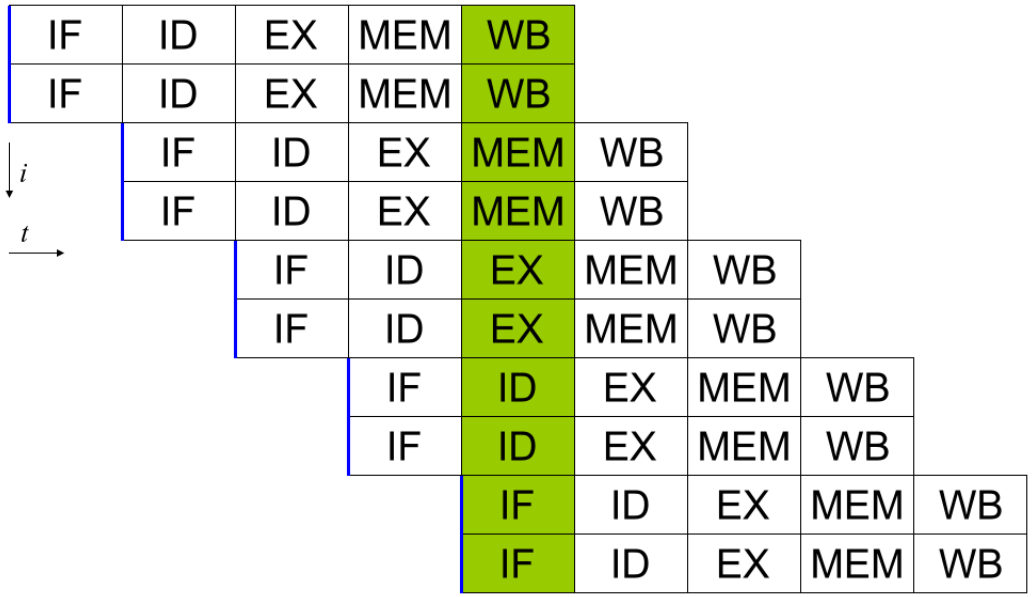


Иллюстрация для Classic RISC pipeline, хотя типичный современный RISC для этого простоват

Суперскалярное исполнение: сущность

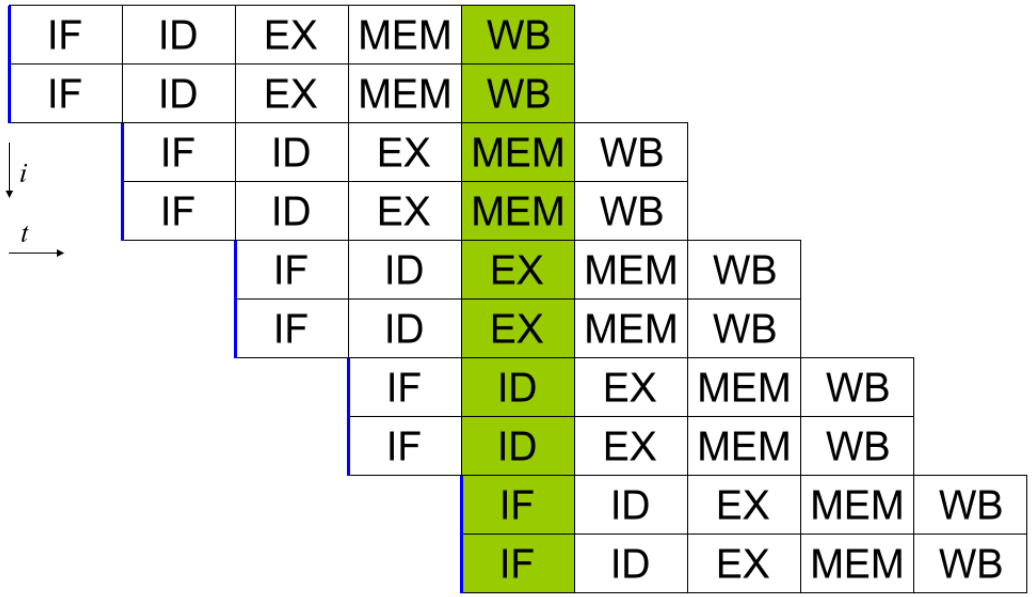


Иллюстрация для Classic RISC pipeline, хотя типичный современный RISC для этого простоват

В 1993 г. — Pentium I, Intel x86

itemize

- Два АЛУ
- Два конвейера, которые их «кормят»
- Компилятор старается располагать рядом независимые инструкции

Суперскалярность — возможность автоматического распараллеливания независимых близлежащих команд в машинном коде

Суперскалярное исполнение: пример

Пример: запись на стек параметров функции при помощи команд `mov`, а не `push`, поскольку соседние `push` изменяют состояние стека $\Longrightarrow$ зависимы

C:

```
f(0, 1, 5, 8);
```

Ассемблер x86 (32 бита):

```
; Без оптимизации vs с оптимизацией
                    sub esp, 16
push 8              mov DWORD PTR [ebp -08h], 8
push 5              mov DWORD PTR [ebp -0Bh], 5
push 1              mov DWORD PTR [ebp -10h], 1
push 0              mov DWORD PTR [ebp -14h], 0
call f              call f
```

Суперскалярный процессор может параллельно исполнить `mov` из примера

VLIW

Very Long Instruction Word — подход к проектированию архитектур, подразумевающий явное распараллеливание независимых близлежащих команд в машинном языке

- Yale Multiflow, 1980-е
- Эльбрус 3, 1993
- Intel Itanium, 2001
- Современные DSP и GPU

Похоже на *микропрограммирование*: в одной машинной команде несколько инструкций для разных блоков процессора, которые задействуются параллельно. Не все блоки разные: м.б. несколько АЛУ, несколько математических сопроцессоров и т.д.

Внеочередное исполнение

Внеочередное исполнение (Out of Order execution) — способ выполнения машинных команд не в порядке следования, а в порядке готовности к исполнению

Впервые: Cray CDC 6600, 1963 г. При приблизительно тех же разрядности и тактовой частоте CDC 6600 был существенно быстрее БЭСМ-6, у которой был конвейер. Но и гораздо сложнее и дороже.

Идея: поток инструкций программы делится на:

1. Последовательность инструкций, к выполнению которых ещё не приступали
2. Инструкции, которыми «занимаются»
3. «Отработанные» инструкции

Внеочередное исполнение

Внеочередное исполнение (Out of Order execution) — способ выполнения машинных команд не в порядке следования, а в порядке готовности к исполнению

Впервые: Cray CDC 6600, 1963 г. При приблизительно тех же разрядности и тактовой частоте CDC 6600 был существенно быстрее БЭСМ-6, у которой был конвейер. Но и гораздо сложнее и дороже.

Идея: поток инструкций программы делится на:

1. Последовательность инструкций, к выполнению которых ещё не приступали
2. Инструкции, которыми «занимаются»
3. «Отработанные» инструкции

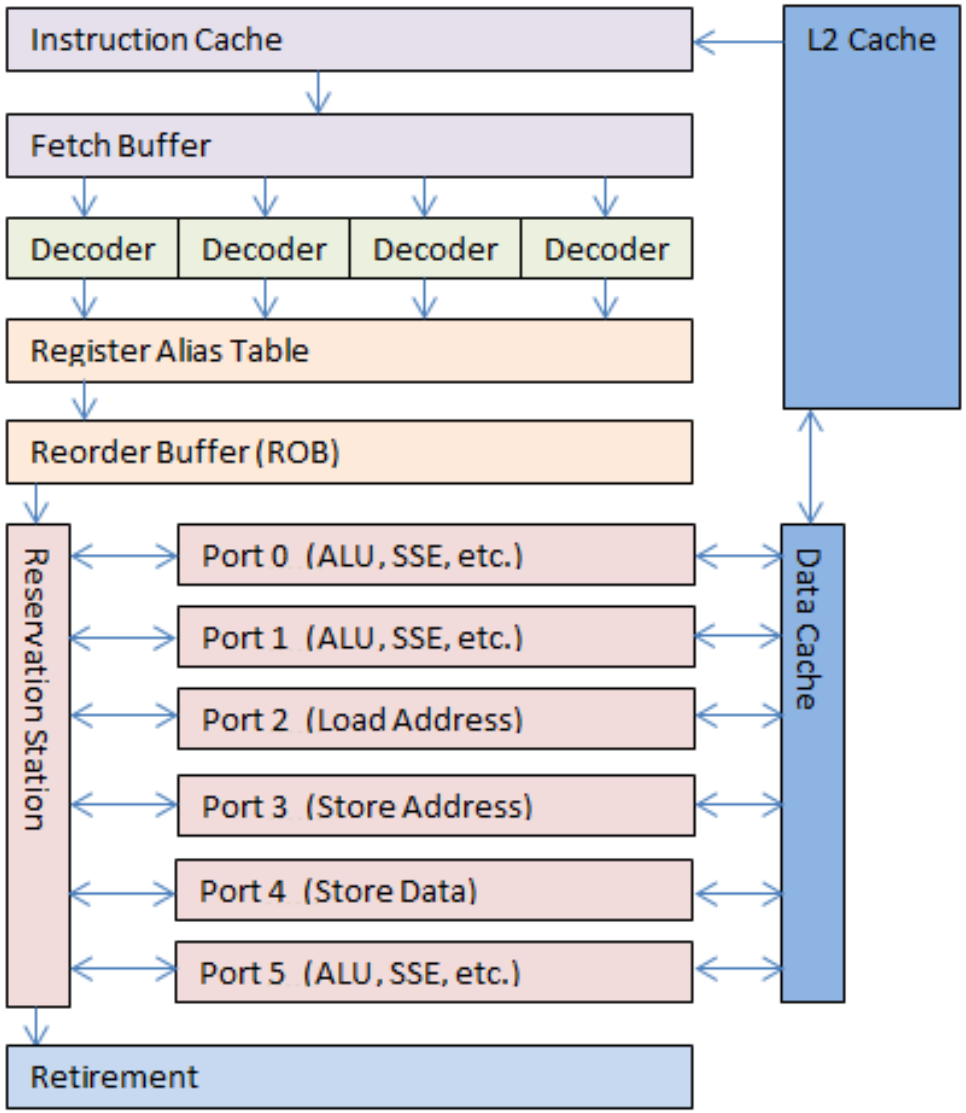
Почти как конвейер, но интеллектуальнее: «текущий» фрагмент программы пропускается не через конвейер, а через «окно», в котором происходит гораздо больше всего.

Современный пример: Intel Core i7

1. Очередная команда разбивается на микрооперации
2. Если фактически независимые микрооперации работают с одними и теми же регистрами, для них выделяются разные экземпляры регистров (стадия *переименования регистров*)
3. Микрооперации помещаются в Reorder Buffer и переупорядочиваются, в т.ч. с микрооперациями близлежащих команд
4. Микрооперации поступают на 6 исполнительных блоков

Это делает процессорное ядро Intel 000

Современный пример: Архитектура Intel 000



Дальнейшее развитие Intel OOO

Впервые OOO применили в Pentium Pro d 1995 г.

- У Intel x86 достаточно сложный машинный код, из-за этого простаивали исполнительные блоки, а блок выборки за ними не успевал

Дальнейшее развитие Intel OOO

Впервые OOO применили в Pentium Pro d 1995 г.

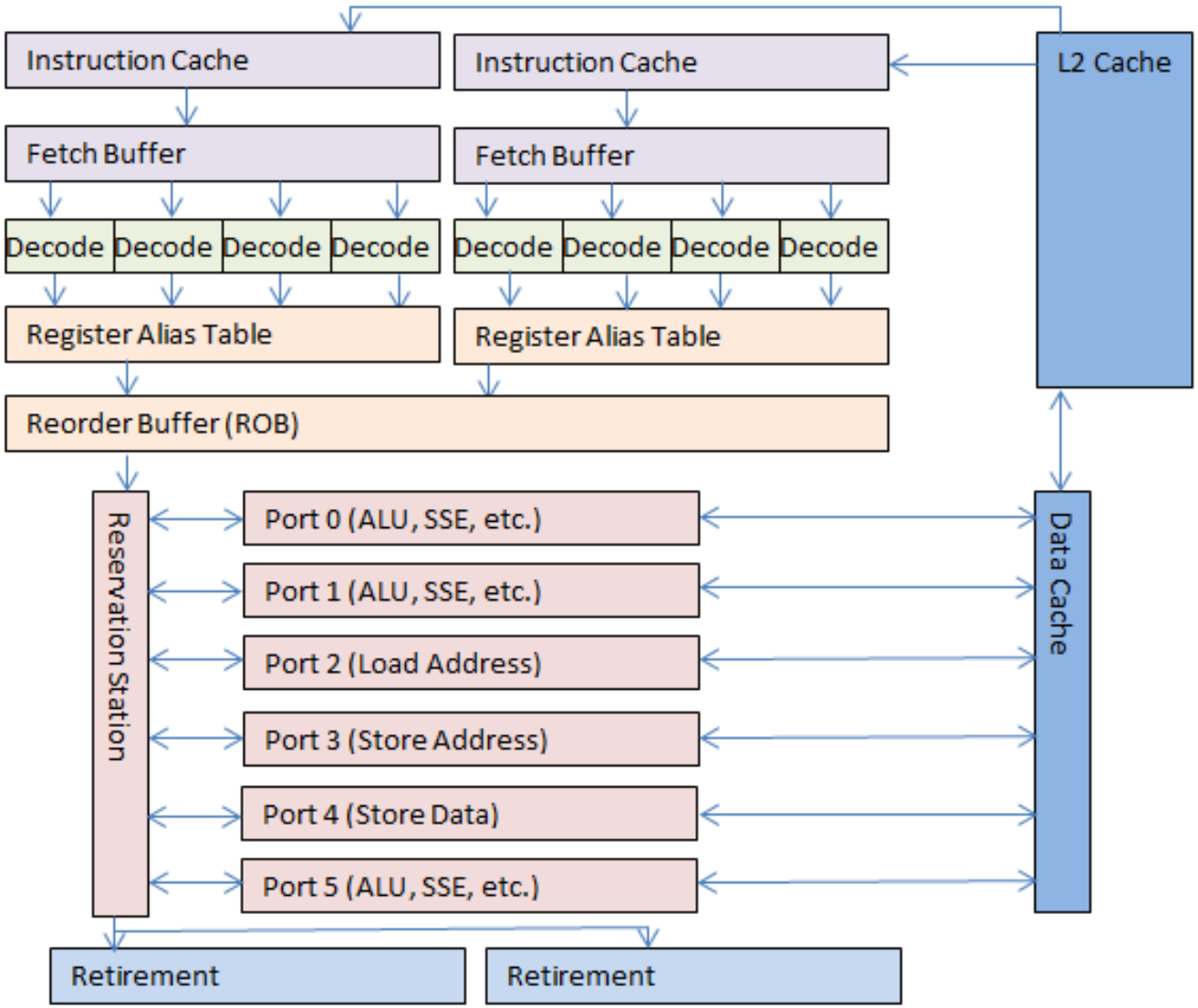
- У Intel x86 достаточно сложный машинный код, из-за этого простаивали исполнительные блоки, а блок выборки за ними не успевал

Идея: можно сделать два блока выборки и исполнять два потока на одном наборе исполнительных блоков

Запатентована в Sun Microsystems в 1994 г., впервые в серийных микропроцессорах реализована в Intel в 2002 г., технология названа *Intel Hyperthreading*

Одно ядро для программиста и ОС выглядит, как два

Intel HyperThreading



Путешествие через вычислительный конвейер процессора (перевод)

Внеочередное исполнение в современных RISC

Многие современные RISC процессоры, например, последние ARM и *отдельные реализации* RISC-V, не только обнаруживают конфликты, но и переупорядочивают инструкции, и выполняют прочие аналогичные оптимизации.
Причина?..

Внеочередное исполнение в современных RISC

Многие современные RISC процессоры, например, последние ARM и *отдельные реализации* RISC-V, не только обнаруживают конфликты, но и переупорядочивают инструкции, и выполняют прочие аналогичные оптимизации.

Причина?..

В условиях параллельного выполнения программ на компилятор положиться уже не получится, т.к. полная информация о последовательности операций доступна лишь во время выполнения.

MIPS, даже современные, часто одноядерные, поэтому могут не обрабатывать конфликты конвейера (помните `nop`?)

Немного мимими: стековые архитектуры

Идея стековой архитектуры

Идея: постфиксная запись операций. Операции либо добавляют данные на стек, либо преобразуют несколько элементов с вершины стека, и помещают на вершину результат

Примеры стековых архитектур

itemize

- Языки программирования и виртуальные машины
  - Forth, PostScript и некоторые современные
  - Lilith (Вирт, Паскаль), AVE (ЛГУ, Алгол68), Java, .NET CLR, CPython и некоторые другие В.М.
- Калькуляторы
  - Калькуляторы HP (с 1970-х – н.в.)
  - Калькуляторы МК-61, МК-52 (1980-е – 90-е) и МК-161, МК-152 (2000-е – н.в.)
- Компьютеры
  - Самсон (Терехов, ЛГУ), Кронос (Котов, Новосибирск) — 1980-е
  - ЛИСП-машины — 1980-е
  - Процессор Intel Itanium (он также VLIW) — 2000-е
  - Бестактовый процессор GA144 для Forth

Преимущества и недостатки стековых архитектур

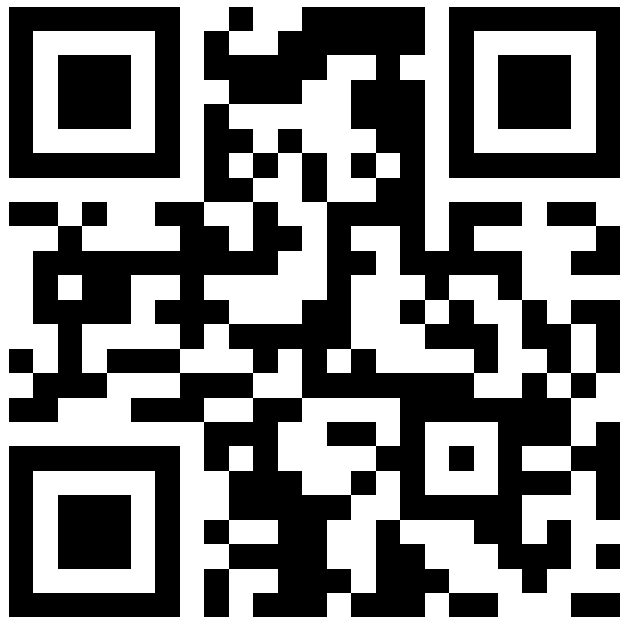
- Преимущества: красиво, легко писать компиляторы
- Недостаток: проблемы с распараллеливанием, неудобный произвольный доступ к верхрим элементам стека (не случайно Itanium VLIW!)

Вопросы и упражнения

Вопросы

- Назовите и определите основные блоки процессора.
- Что такое вычислительный конвейер?
- Во сколько раз может максимально ускорить выполнение программы конвейер, выполняющий команды в 3 этапа?
- Назовите виды конфликтов конвейера.
- Как RISC-процессоры разрешают конфликты конвейера?
- Как CISC-процессоры разрешают конфликты конвейера?
- Дайте определение суперскалярной архитектуре.
- Что такое VLIW?
- Что такое внеочередное исполнение машинных команд?
- В чём идея технологии HyperThreading?
- Почему современные многоядерные RISC сами разрешают конфликты конвейера?
- Что такое стековая архитектура, каковы преимущества и недостатки стековых архитектур?
- Приведите примеры стековых архитектур.

Вопросы



EDU.DLUCIV.NAME