Алгоритмические и логические основы цифровой вычислительной техники / Компьютерные системы и сети

БГТУ
кафедра ПИ

доцент Самаль Дмитрий Иванович
dmitry_samal@mail.ru,
а.408-1

Лекция 2
«Архитектура системы команд»

2020

1

План лекции

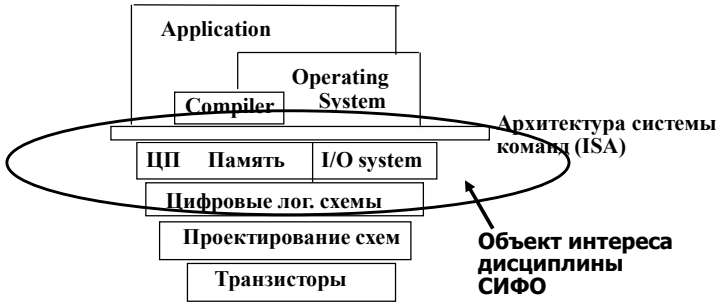
1. Задачи архитектуры системы команд
2. Аккумуляторная архитектура
3. Стековая архитектура
4. Регистровая архитектура
5. Архитектура с выделенным доступом к памяти

Слайд 2

2

The Instruction Set Architecture

Архитектура системы команд – интерфейс между всем выполняемым на машине ПО и аппаратным обеспечением.



Слайд 3

3

What is Computer Architecture?

Computer Architecture =
Machine Organization +
Instruction Set Architecture

What the machine looks like

How you talk to the machine

Слайд 4

4

Архитектура системы команд
Instruction Set Architecture

- Перечень видимых для программиста составных частей архитектуры системы команд:
  - коды команд - opcodes (доступные для выполнения операции)
  - число и типы регистров
  - форматы команд
  - способы адресации и доступа к памяти
  - условия возникновения исключений (exceptional conditions)

Слайд 5

5

Архитектура системы команд
Instruction Set Architecture

- АСК (ISA) – очень важный уровень абстракции:
  - *интерфейс* между аппаратурой и низко-уровневым ПО
  - *стандартизирует* команды, шаблоны битов машинного языка и т.д.
  - преимущества: *позволяет реализовывать различные аппаратные варианты одной программной архитектуры*
  - недостатки: *периодически препятствует внедрению новых технологий и прочих инноваций*
- Современные АСК:
  - 80x86/Pentium/K6, PowerPC, DEC Alpha, MIPS, SPARC, HP

Слайд 6

6

Ключевые моменты АСК
Key ISA decisions

длина команды
- все команды одной длины или нет?

сколько необходимо регистров?

каким образом организовать работу с памятью?
- например, во все регистры можно загружать операнды из памяти напрямую или нет?

формат команды
- какие биты что определяют?

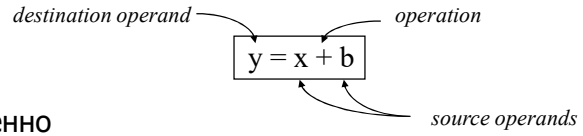
операнды
- сколько? размер?
- как вычислять исполнительные адреса памяти?

операции
- какие операции будут производится?

Слайд 7

7

Ключевые моменты АСК
Key ISA decisions



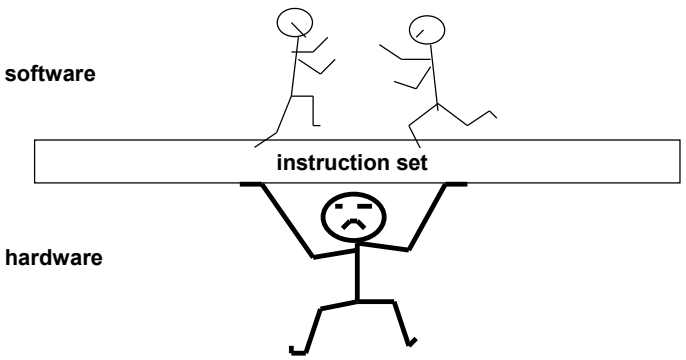
- операции
  - сколько?
  - какие именно
- операнды
  - сколько?
  - местоположение
  - типы
  - каким образом определяются?
- формат команд
  - размер
  - сколько всего форматов в данном ЦП?

(add r1, r2, r5)

Слайд 8

8

The Instruction Set: a Critical Interface



Слайд 9

9

ИТОГО:

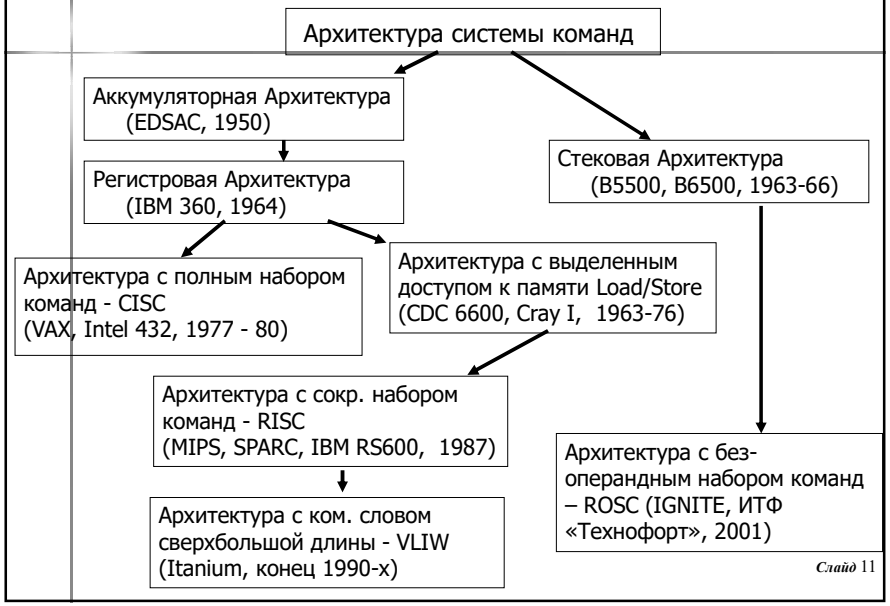
АСК(ISA) характеризуется ответами на вопросы:

- Какого вида данные будут представлены в ВМ и в какой форме?
- Где эти данные могут храниться помимо основной памяти?
- Каким образом будет осуществляться доступ к данным?
- Какие операции могут быть выполнены над данными?
- Сколько операндов может присутствовать в команде?
- Как будет определяться адрес очередной команды?
- Каким образом будет закодированы команды?

Слайд 10

10

Классификация архитектур системы команд



Слайд 11

11

Цикл выполнения команды

Получение команды из «хранилища» программ

Определение задаваемого действия и фактического размера команды

Вычисление адресов и получение операндов

Вычисление результирующего значения либо состояния

Сохранение результатов операции в «хранилище» для последующего использования

Вычисление адреса следующей команды

Слайд 12

12

Классификация архитектур системы команд (по месту хранения операндов)

- Аккумуляторная Архитектура (EDSAC, 1950)
- Стековая архитектура (B5500, B6500, 1963-66)
- Регистровая Архитектура (IBM 360, 1964)
- Архитектура с выделенным доступом к памяти Load/Store (CDC 6600, Cray I, 1963-76)

Слайд 13

13

Аккумуляторная архитектура

Исторически первая архитектура.

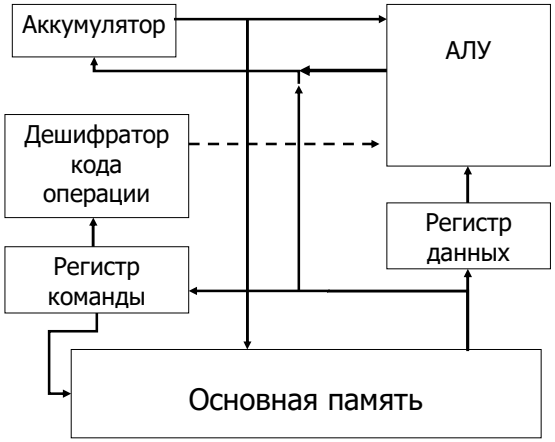
В процессоре выделенный регистр – аккумулятор. В этот регистр – заносится результат операции.

Изначально оба операнда в памяти. Перед операцией один из них – в аккумулятор. После выполнения результат в аккумуляторе (либо в памяти - если он не является операндом для следующей команды).

Слайд 14

14

Аккумуляторная архитектура



Слайд 15

15

Аккумуляторная архитектура

Загрузка в акк. ячейки x – load x.

Запись из акк. в ячейку x – store x.

Один операнд – в регистр данных. Второй в аккумуляторе.

<u>Достоинства:</u>

Короткие команды

Простота декодирования команд

<u>Недостатки:</u>

многократные обращения к памяти

Популярна в ранних ВМ – IBM 7090, DEC PDP-8, MOS 6502

Слайд 16

16

Стековая архитектура

Стек – множество логически взаимосвязанных по принципу LIFO ячеек.

Две операции – push и pop. Запись и чтение только в/из вершины стека.

Операнды помещаются в стек, в процессе операции они выталкиваются из стека. Результат операции – в стек.

Для стековой архитектуры лучше всего подходит обратная польская нотация (Чарльз Хэмблин, 1957), разработанная на основе «польской нотации» (Я. Лукасевич, 1920).

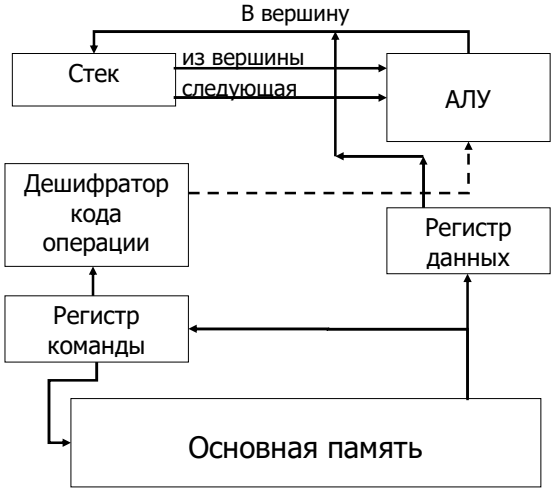
a = a+b+a*c

a = ab+ac*+

Слайд 17

17

Стековая архитектура



Слайд 18

18

Стековая архитектура

Информация в стек – из памяти или АЛУ (only).

Результат – в АЛУ (автоматически).

Сохранение из стека в память x по команде (pop x).

Достоинства:

- стек может быть неоднороден (верхние ячейки быстрые, нижние медленные).
- сокращение адр. части команд.
- компактный код
- простой компилятор
- простое декодирование команд

Слайд 19

19

Стековая архитектура

Недостатки:

Компилятор не может создать эф. код (нет произвольного обращения к памяти).

Стек – узкое место. Низкая производительность.

Расцвет архитектуры – 60е годы: HP 2116B, HP 3000/70.

Последние годы – повышение интереса к стековой архитектуре. Удобен для Java и Forth. Машины – JEM 1 и JEM 2 от aJile Systems, Clip от Imsys.

ROSC (Removed Operand Set Computer) – новая реинкарнация стековой архитектуры. IGNITE от Patriot Scientist.

Слайд 20

20

Регистровая архитектура

Массив регистров (РОН) – явно управляемый кэш для хранения недавно использовавшихся данных.

Размер регистров – обычно совпадает с маш. словом.

К люб. регистру можно обратиться по номеру.

Меньшее количество регистров – меньше разрядов на кодирование номера регистра – короче команда.

Операнды – либо в памяти либо в регистрах.

Три подвида команд:

Регистр-регистр

Регистр- память

Память-память

Слайд 21

21

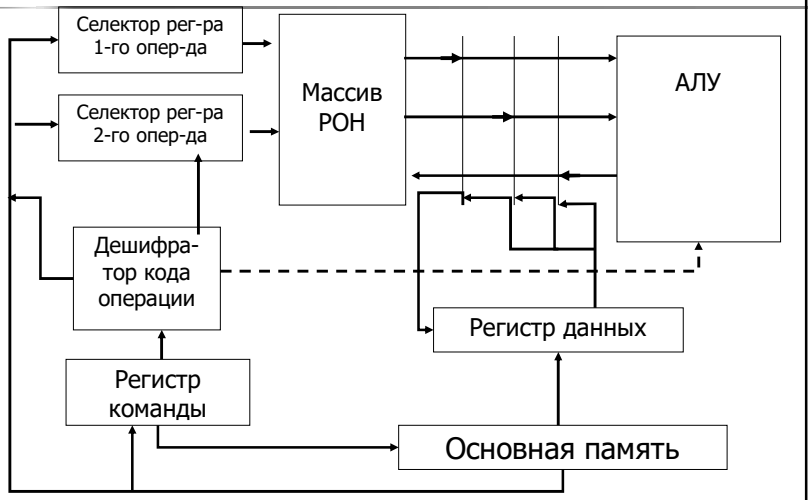
Регистровая архитектура

Вариант	Достоинства	Недостатки
Регистр-регистр	Простота реализации, фикс. длина команд, простота кода при компиляции, возможность выполнения всех команд за одинак. кол-во тактов	Большая длина кода, из-за фикс. длины команд – часть разрядов в команде не используется.
Регистр-память	Данные могут быть доступны и без загрузки в рег., простота кодирования команд, компактный код	Потеря одного из опер. при записи результата, длинное поле адреса в команде – сокращает кол-во РОН, CPI зависит от места размещ. операнда.
Память-память	Компактность кода, малая потребность в регистрах для хранения промежут. результатов	Разнообразие форматов команд и времени их исполнения, низкое быстродействие.

Слайд 22

22

Регистровая архитектура



Слайд 23

23

Регистровая архитектура

Операции загрузки операндов в РОН – идентичны работе с аккумулятором. Отличие – выбор нужного регистра.

Выполнение операции АЛУ включает в себя:

Выбор регистра первого операнда

Определение местоположения второго операнда (память или регистр)

Подача на вход АЛУ операндов и выполнение операции

Выбор регистра результата и занесение в него результата операции из АЛУ.

Между АЛУ и массивом регистров должно быть минимум три шины. Почему?

Слайд 24

24

Регистровая архитектура

Достоинства – компактность кода, высокая скорость.

Недостатки – более длинные инструкции (по сравнению с аккумуляторной архитектурой)

Регистровая АСК – преобладающая в настоящее время (все современные персоналки).

Первые машины – IBM 360/370, PDP-11.

Слайд 25

25

Архитектура с выделенным доступом к памяти

Загрузка в регистр N ячейки x – load x N

Запись из регистра N в ячейку x – store N x

Данная АСК характерна для всех ВМ с RISC архитектурой.

Команды как правило – трёх адресный формат и 32 бита длина.

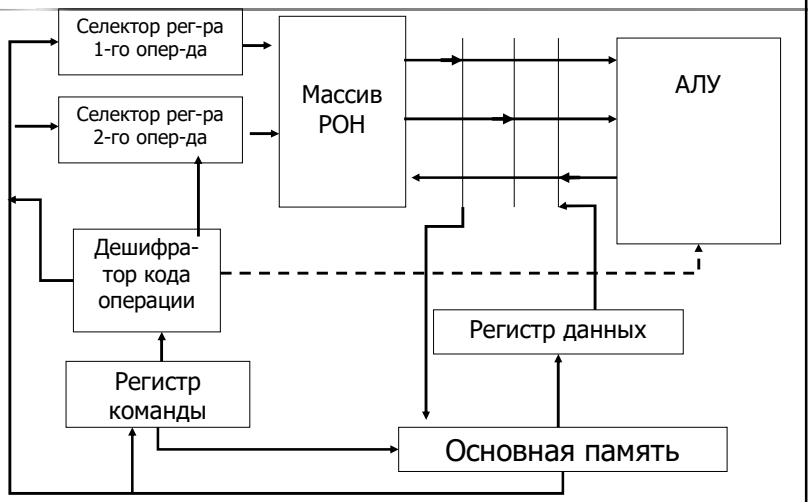
Достоинства Load/Store АСК – простота декодирования и исполнения команд.

HP PA-RISC, IBM RS/6000, SUN Sparc, MIPS R4000, DEC Alpha, etc.

Слайд 26

26

Архитектура с выделенным доступом к памяти



Слайд 27

27

ИТОГО:

Stack machine:

- "Push" загружает содержимое памяти в 1ый регистр ("вершину стека"), передвигая все остальные регистры вниз
- "Pop" действует в обратном порядке
- "Add" комбинирует содержимое первых двух регистров, передвигая остальные вверх.

Accumulator machine:

- Только один регистр (называемый "аккумулятором")
- Команды включают в себя "store" и саму операцию "acc ← acc + mem"

Register-Memory machine :

- Арифметические команды могут использовать данные как из регистров так и/или из памяти

Load-Store Machine (aka Register-Register Machine):

- Арифметические команды могут оперировать только с содержимым регистров.

Слайд 28

28

Сравнение классов АСК

Последовательности кода для
C = A + B

Stack	Accumulator	Register-Memory	Load-Store
Push A	Load A	Add C, A, B	Load R1,A
Push B	Add B		Load R2,B
Add	Store C		Add R3,R1,R2
Pop C			Store C,R3

Слайд 29