Языки описания схем

(mk.cs.msu.ru → Лекционные курсы → Языки описания схем)

Блок 11

Что такое процессор

Архитектура системы команд

лектор:

Подымов Владислав Васильевич

e-mail:

valdus@yandex.ru

Осень 2018

Архитектура процессора

Процессор — это цифровая схема, исполняющая программу (последовательность команд), подаваемую на вход в машинном коде

Архитектура процессора — это понятие, в широком смысле означающее общее устройство и детали реализации процессора, включая

1. то, как устроены "понимаемые" процессором команды (архитектура системы команд)
2. то, через какие этапы проходит инструкция при выполнении, и как функционируют и соединены модули процессора, отвечающие этим этапам (микроархитектура)
3. (*не будет рассматриваться*) промежуточные и более низкие уровни представления схемы процессора
4. (*почти не будет рассматриваться*) особенности огранизации окружения процессора

Архитектура системы команд

(ISA; Instruction Set Architecture)

Это перечень команд, при помощи которых программист может последовательно управлять вычислением процессора

ISA включает в себя

- синтаксис команд: то, какие массивы логических значений являются командами
- операционную семантику команд:
  - как выглядит состояние вычисления программы
  - как каждая команда (*последовательно*) изменяет состояние вычисления

Архитектура системы команд

При описании ISA обычно используются два способа записи команд:

- машинный код: описание команды как массива логических значений
  - это то, как команда выглядит с точки зрения процессора
  - например, [00000000001000100001100000100000]
- ассемблерная запись: удобочитаемая короткая символьная запись команды
  - это то, как пользователь может коротко и удобно описывать команду
  - ассемблерная запись (*как правило*) соответствует одному или нескольким равноценным машинным кодам
  - например, **add** $\$1, \$2, \$3$

Архитектура системы команд

Точный вид состояния вычисления программы зависит от назначения и устройства процессора

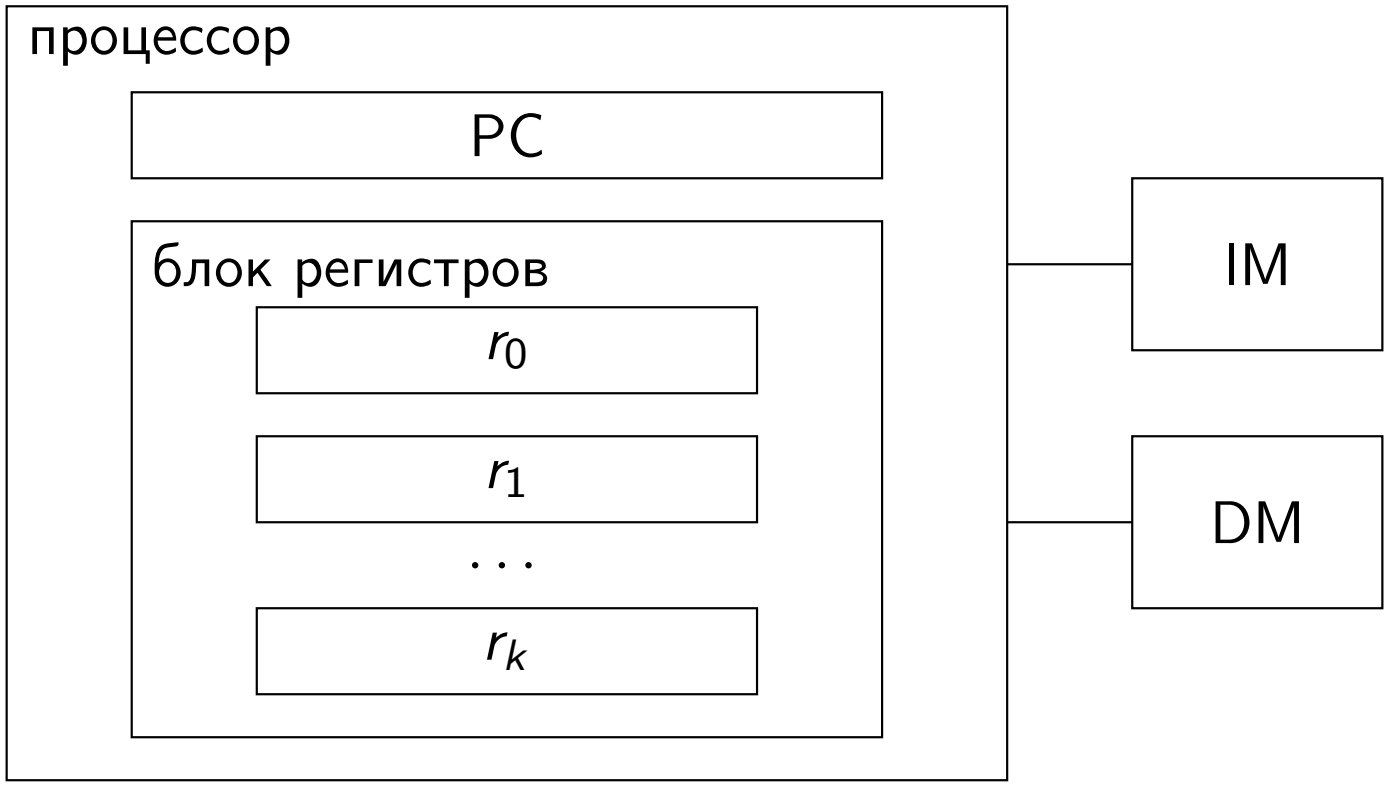
Компоненты состояния вычисления определяются так, чтобы

- естественным образом располагаться в определении **последовательной** операционной семантики
- некоторые из компонентов были *наблюдаемы* пользователем — например, значения переменных во внешней памяти

Основная задача процессора — обеспечить **схемную** обработку программы, согласно которой наблюдаемая часть состояния вычисления **последовательно** изменяется согласно операционной семантике команд

Архитектура системы команд

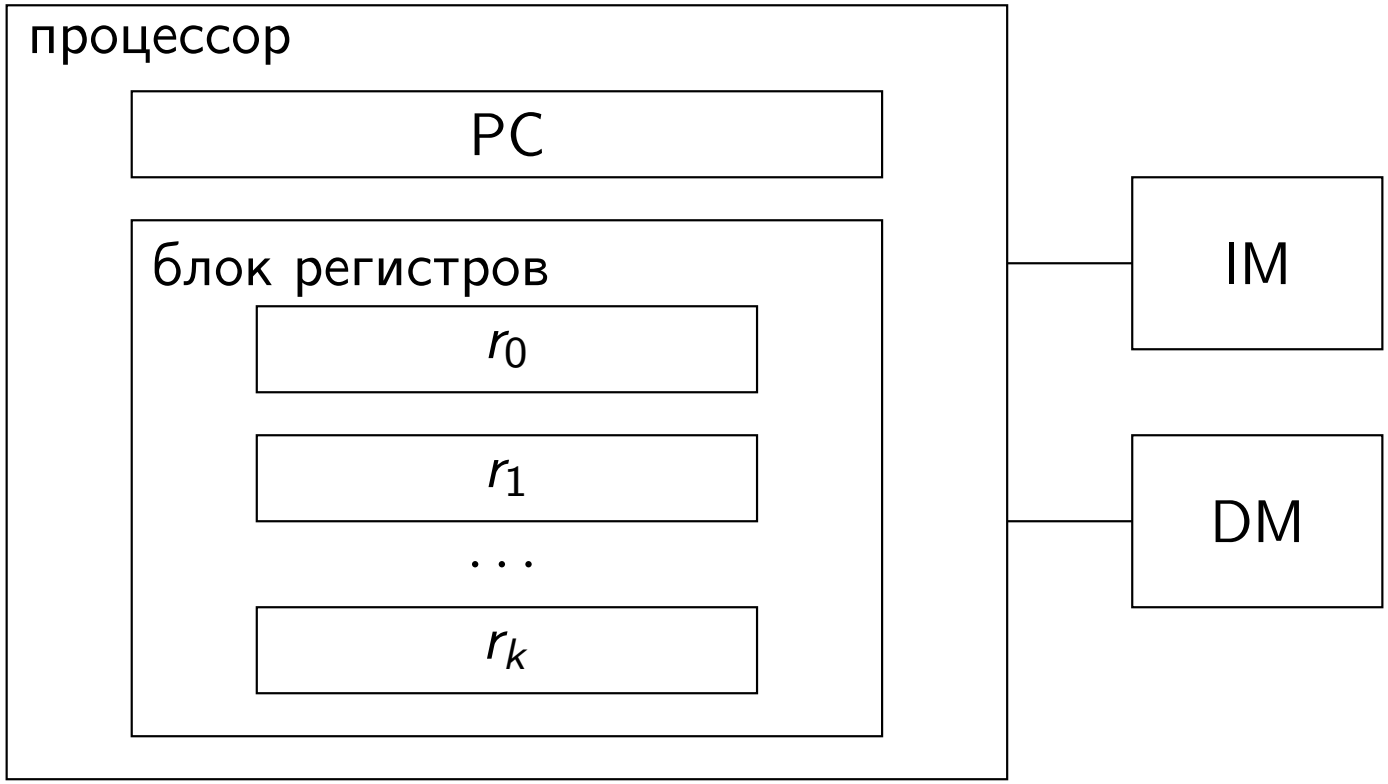
Типовые элементы процессора и его окружения, значения которых являются компонентами состояния вычисления:



Значение счётчика команд (PC) — это текущая выполняемая инструкция

Архитектура системы команд

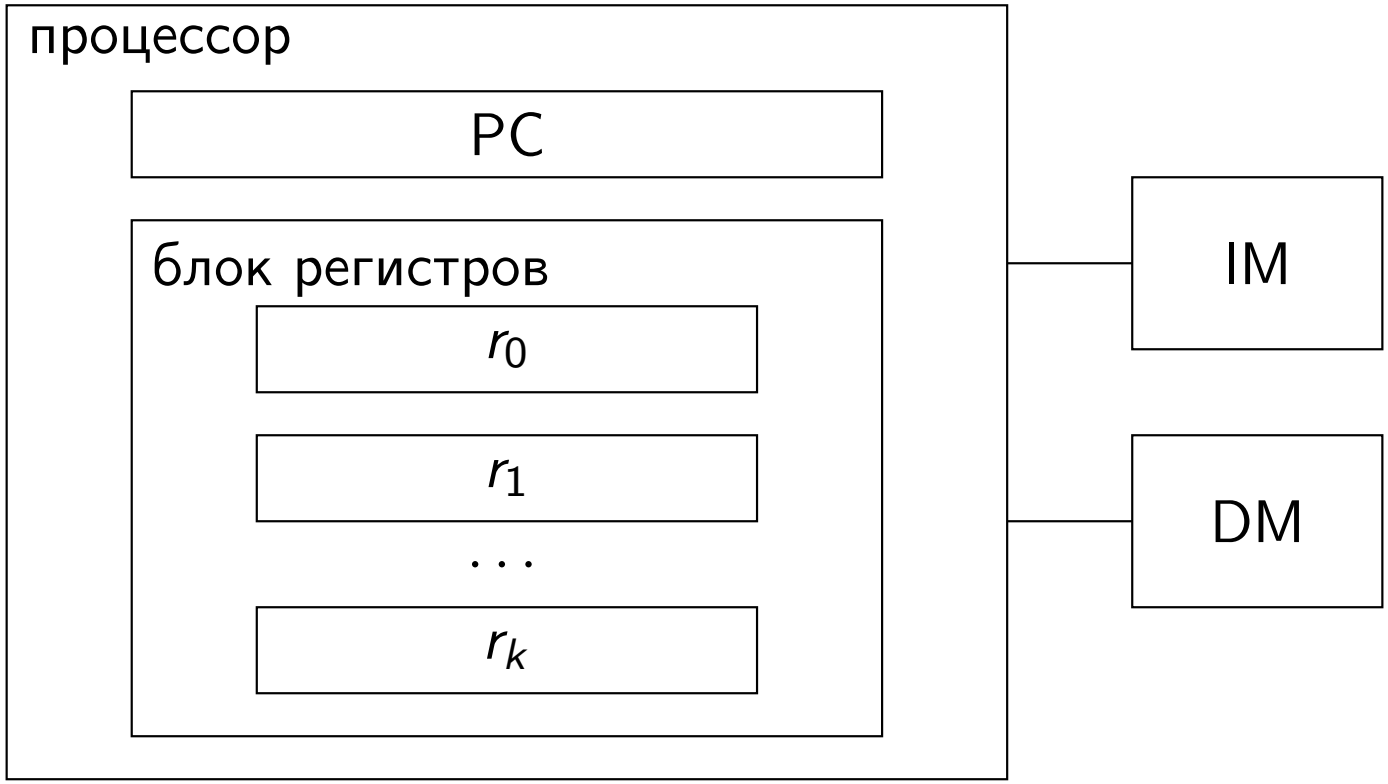
Типовые элементы процессора и его окружения, значения которых являются компонентами состояния вычисления:



Регистры — это нумерованные переменные внутри процессора

Архитектура системы команд

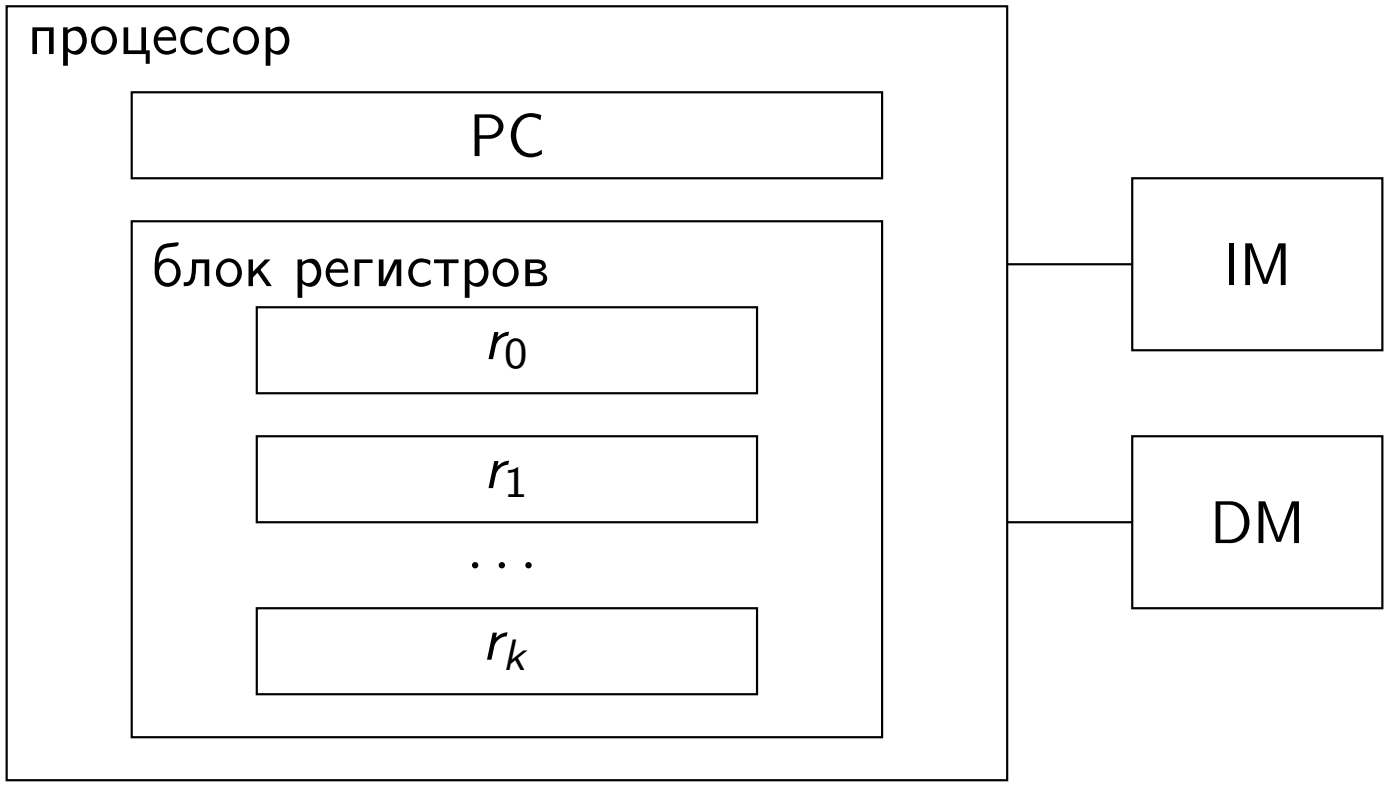
Типовые элементы процессора и его окружения, значения которых являются компонентами состояния вычисления:



Внешняя память — это набор (массив) ячеек данных, доступных по адресу (смещению в массиве)

Архитектура системы команд

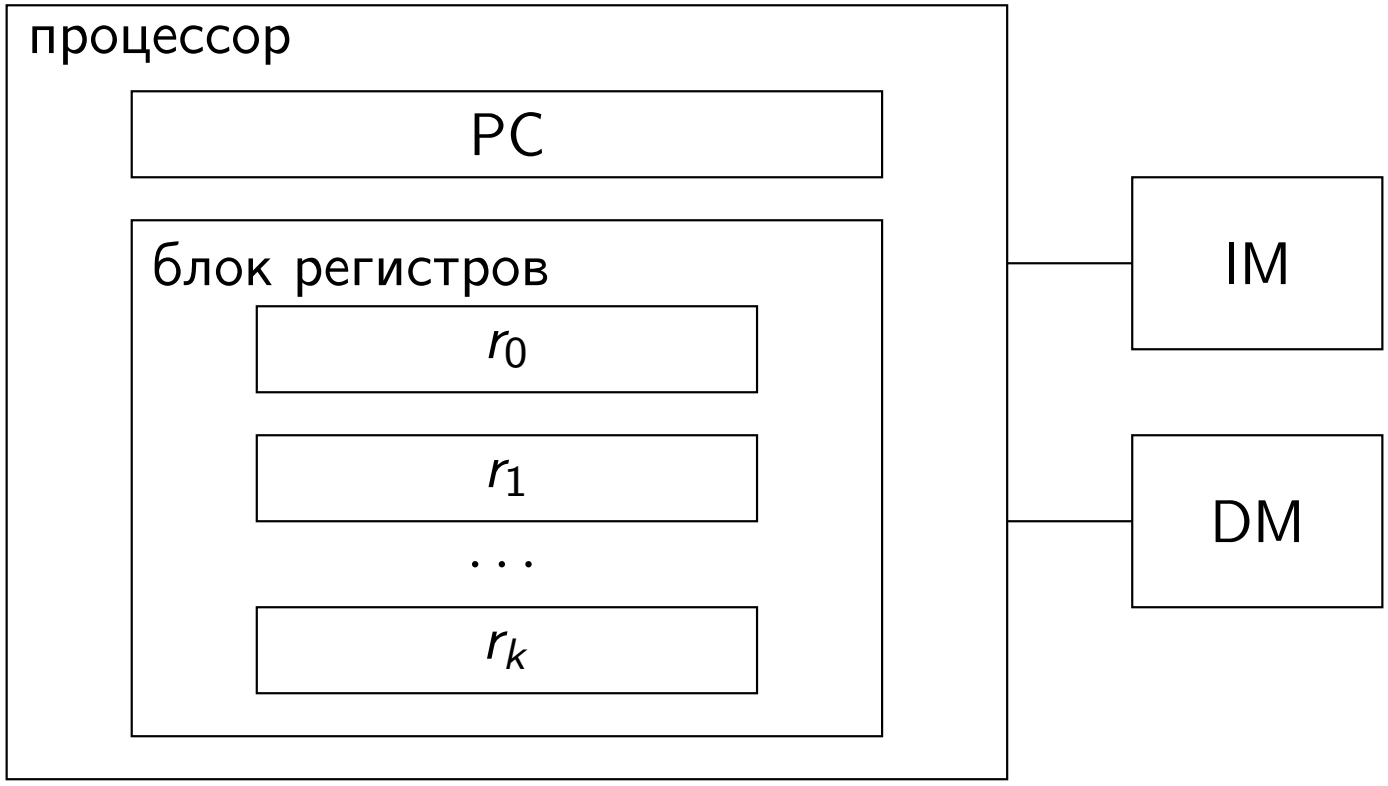
Типовые элементы процессора и его окружения, значения которых являются компонентами состояния вычисления:



Память инструкций (IM) доступна только для чтения и хранит выполняемую программу

Архитектура системы команд

Типовые элементы процессора и его окружения, значения которых являются компонентами состояния вычисления:



Память данных (DM) доступна для чтения и записи

Модельная архитектура: битность

В рамках курса для примера остановимся на простой показательной архитектуре процессора, демонстрирующей ключевые особенности “реальных” архитектур

Так будут выделяться комментарии, относящиеся к реальным архитектурам

“(П) :” — обозначение особенностей процессора, рассматриваемого в рамках основного задания курса

Как правило, архитектура процессора соответствует заданной битности: это число, обозначающее ширину основных шин, по которым пересылаются данные, и всех базовых элементов состояния вычисления

(П) : счётчик команд, каждый регистр, машинный код команд, каждая ячейка данных, . . . имеют ширину 16

Модельная архитектура: регистры

В общем случае каждый регистр имеет своё специальное назначение: трактовку хранимого значения и способы использования и перезаписи этого значения, определяемые системой команд

Некоторые регистры имеют общее назначение: хранимое значение всегда доступно и может быть произвольно использовано и перезаписано

(П) : все регистры процессора — это 4 регистра общего назначения, доступных по номерам $0, 1, 2, 3$.

Модельная архитектура: типы команд

“Реальный” процессор имеет огромный спектр команд

Некоторые команды похожи друг на друга: имеют одинаковое устройство машинного кода и/или схожую семантику

Команды принято группировать по типам согласно такой схожести

Модельная архитектура: типы команд

Группировка команд может производиться, например, по следующим признакам:

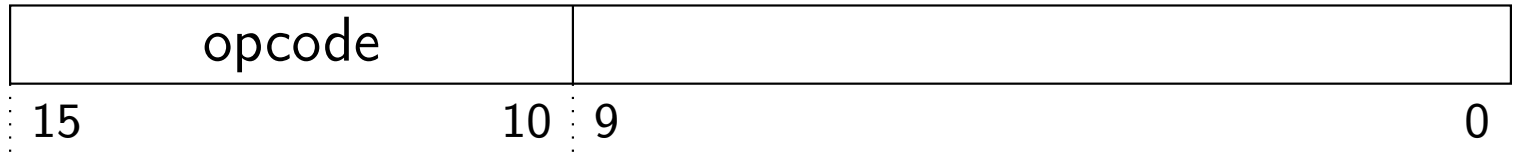
- арифметико-логические команды производят арифметическую или логическую операцию над заданными значениями и записывают результат в регистр
- регистровые команды используют и перезаписывают только значения, хранящиеся в регистрах
- немедленные команды используют несколько значений регистров, а также константу, записанную непосредственно в машинном коде команды (немедленное значение)
- команды ветвления определяют расположение следующей выполняемой команды в памяти инструкций
  - при выполнении других команд следующей выполняется инструкция, расположенная в следующей ячейке памяти инструкций
- команды доступа к памяти управляют значениями в ячейках памяти данных

Модельная архитектура: типы команд

(П) : имеются следующие типы команд:

- AR: арифметико-логические команды на регистрах
- AI: немедленные арифметико-логические команды
- BI: немедленные команды условного ветвления
- MI: немедленные команды доступа к памяти
- J: команды безусловного ветвления

Общие слова о формате команд



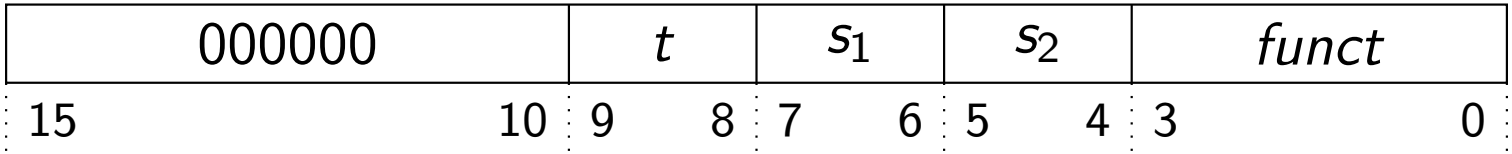
Общая часть машинного кода всех команд, определяющая тип команды — это код операции (opcode)

(П) : код операции располагается в 6 старших битах машинного кода команды

В дальнейшем описании команд

- $\$i$ — значение, хранимое в регистре с номером i
- $\leftarrow$ — “записать значение выражения в правой части на место значения левой части”
- pc — значение счётчика команд
- $M[i]$ — 16-битное значение в памяти данных, хранимое по смещению $16 \cdot i$ бит от начала массива памяти

AR-команды



Семантика AR-команд:

$$\$t \leftarrow f(\$s_1, \$s_2);\ pc \leftarrow pc + 1;$$

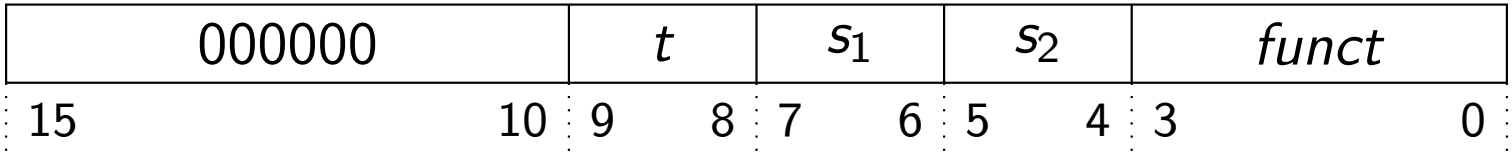
Функция f определяется значением *funct*:

Ассемблерная запись:

$$\textbf{op}\ \$t, \$s_1, \$s_2$$

- $funct = [1000]$:
  - f — сумма чисел
  - **op** = **add**
- $funct = [1010]$:
  - f — разность чисел
  - **op** = **sub**

AR-команды



Семантика AR-команд:

$$\$t \leftarrow f(\$s_1, \$s_2);\ pc \leftarrow pc + 1;$$

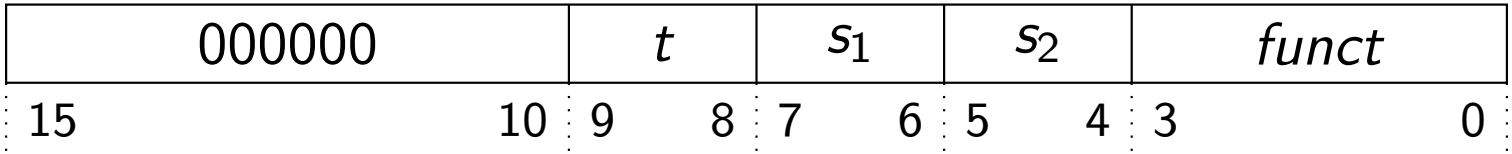
Функция f определяется значением *funct*:

Ассемблерная запись:

$$\textbf{op}\ \$t, \$s_1, \$s_2$$

- *funct* $= [0100]$:
  - f — побитовое И
  - **op** $=$ **and**
- *funct* $= [0110]$:
  - f — побитовое ИЛИ
  - **op** $=$ **or**

AR-команды



Семантика AR-команд:

$$\$t \leftarrow f(\$s_1, \$s_2);\ pc \leftarrow pc + 1;$$

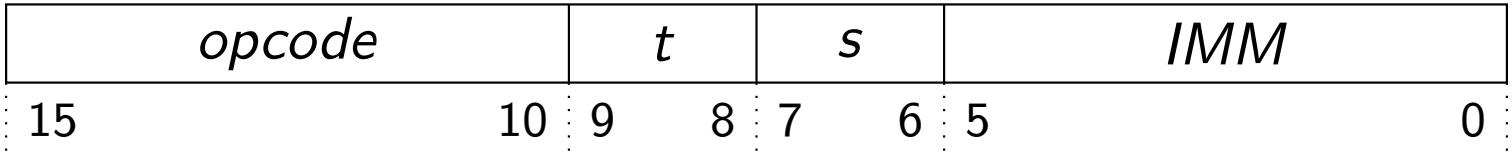
Функция f определяется значением *funct*:

Ассемблерная запись:

$$\textbf{op}\ \$t, \$s_1, \$s_2$$

- $funct = [0010]$:
  - $f(a, b) = 1$, если $a < b$, и $f(a, b) = 0$ иначе
  - числа трактуются как беззнаковые
  - **op** = **sltu**
- $funct = [0011]$:
  - $f(a, b) = 1$, если $a < b$, и $f(a, b) = 0$ иначе
  - числа трактуются как знаковые
  - **op** = **slt**

AI-команды



Семантика AI-команд:

$$\$t \leftarrow f(\$s, IMM);\ pc \leftarrow pc + 1;$$

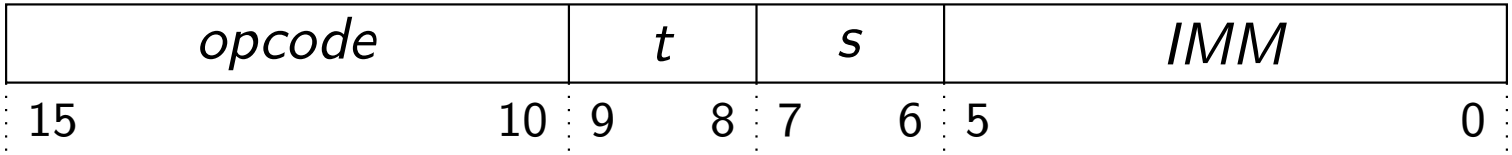
Функция f определяется значением *opcode*:

Ассемблерная запись:

$$\textbf{op}\ \$t, \$s_1, IMM$$

- *opcode* $= [101000]$:
  - f — сумма чисел
  - IMM — беззнаковое число
  - **op** $=$ **addiu**
- *opcode* $= [101001]$:
  - f — сумма чисел
  - IMM — знаковое число
  - **op** $=$ **addi**

AI-команды



Семантика AI-команд:

$$\$t \leftarrow f(\$s, IMM);\ pc \leftarrow pc + 1;$$

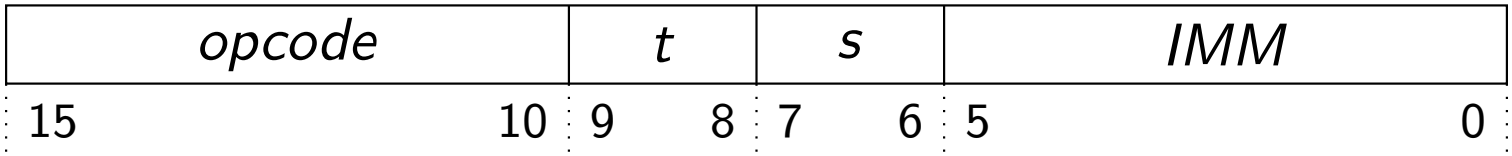
Функция f определяется значением *opcode*:

Ассемблерная запись:

$$\textbf{op}\ \$t, \$s_1, IMM$$

- *opcode* = [100100]:
  - f — побитовое И
  - *IMM* — беззнаковое число
  - **op** = **andiu**
- *opcode* = [100101]:
  - f — побитовое И
  - *IMM* — знаковое число
  - **op** = **andi**

AI-команды



Семантика AI-команд:

$$\$t \leftarrow f(\$s, IMM);\ pc \leftarrow pc + 1;$$

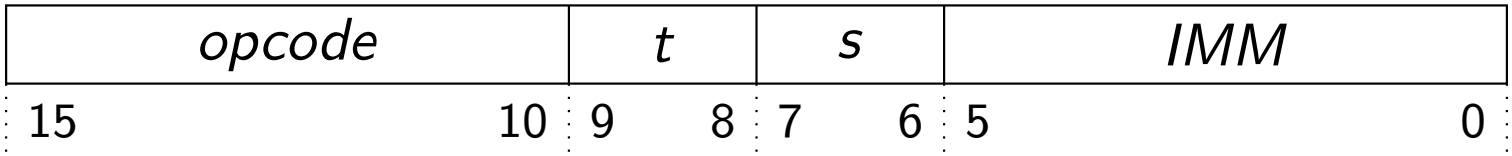
Функция f определяется значением *opcode*:

Ассемблерная запись:

$$\textbf{op}\ \$t, \$s_1, IMM$$

- *opcode* $= [100110]$:
  - f — побитовое ИЛИ
  - IMM — беззнаковое число
  - **op** $=$ **oriu**
- *opcode* $= [100111]$:
  - f — побитовое ИЛИ
  - IMM — знаковое число
  - **op** $=$ **ori**

AI-команды



Семантика AI-команд:

$$\$t \leftarrow f(\$s, IMM);\ pc \leftarrow pc + 1;$$

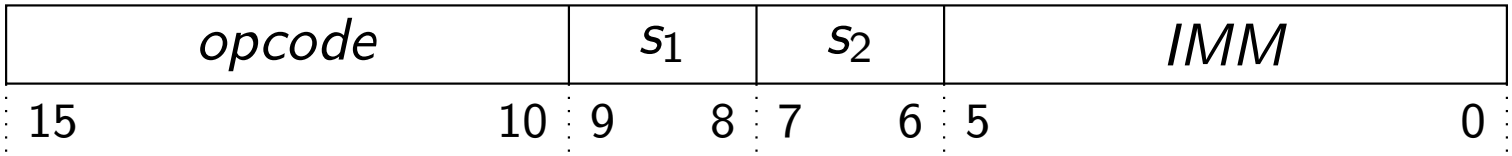
Функция f определяется значением *opcode*:

Ассемблерная запись:

$$\textbf{op}\ \$t, \$s_1, IMM$$

- *opcode* $= [100010]$:
  - $f(a, b) = 1$, если $a < b$, и $f(a, b) = 0$ иначе
  - *IMM* — беззнаковое число
  - **op** = **sltiu**
- *opcode* $= [100011]$:
  - $f(a, b) = 1$, если $a < b$, и $f(a, b) = 0$ иначе
  - *IMM* — знаковое число
  - **op** = **slti**

BI-команды



Семантика BI-команд:

$$pc \leftarrow cond(\$s_1, \$s_2)?pc + 1 + IMM : pc + 1$$

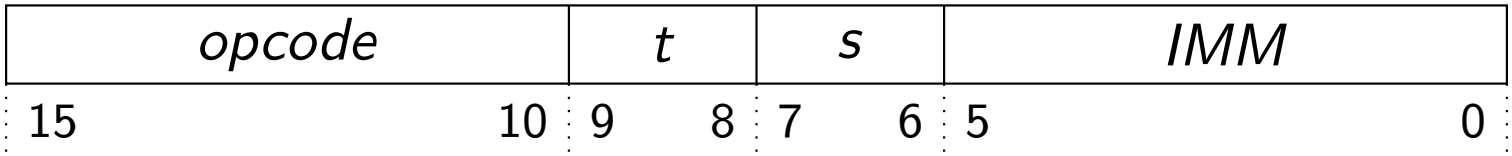
Ассемблерная запись:

$$\textbf{op}\ \$s_1, \$s_2, IMM$$

Условие *cond* определяется значением *opcode*:

- *opcode* = [000010]:
  - $cond(a, b) = 1 \quad \Leftrightarrow \quad a = b$
  - *IMM* — знаковое число
  - **op** = **beq**
- *opcode* = [000011]:
  - $cond(a, b) = 1 \quad \Leftrightarrow \quad a \neq b$
  - *IMM* — знаковое число
  - **op** = **bne**

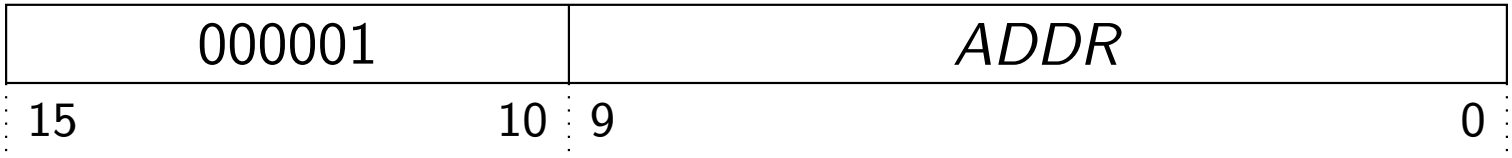
MI-команды



- $opcode = [000100]$:
  - $\$t \leftarrow M[\$s + IMM];\ pc \leftarrow pc + 1;$
  - IMM — знаковое число
  - ассемблерная запись: **lw** $\$t, IMM(\$s)$
- $opcode = [000101]$:
  - $M[\$s + IMM] \leftarrow \$t;\ pc \leftarrow pc + 1;$
  - IMM — знаковое число
  - ассемблерная запись: **sw** $\$t, IMM(\$s)$

J-команды

Нам понадобится только одна J-команда:



Семантика:

$$pc \leftarrow ADDR$$

ADDR трактуется как беззнаковое число

Ассемблерная запись: **j** *ADDR*