Лекция 3
ARM

История

- Acorn Computers (Cambridge) начала разработку в 1983
- Acorn Archimede — персональный компьютер 1987 — архитектура ARM2
- ~30000 транзисторов
- «hardwired» декодер инструкций — без микрокода
- Быстрее и проще i286

ARM holdings

- Основан в 1990 году как Advanced RISC Machines Ltd
- Учредители: Acorn Computers, VLSI Technology, (кто третий учредитель?)
- Основной бизнес: разработка архитектуры и лицензирование архитектуры и интеллектуальной собственности
- Не выпускает процессоры

ARM holdings

- Основан в 1990 году как Advanced RISC Machines Ltd
- Учредители: Acorn Computers, VLSI Technology, Apple Computer
- Основной бизнес: разработка архитектуры и лицензирование архитектуры и интеллектуальной собственности
- Не выпускает процессоры

Производители ARM-процессоров

- Samsung
- Apple
- Broadcom
- Qualcomm
- AMD
- ...

Ядра ARM

- Профиль A (application): Cortex-A5, Cortex-A7, … Cortex-A17 (32 bit); Cortex-A53… (64 bit)
- Профиль R (real-time): Cortex-R4, …, Cortex-R7
- Профиль M (microcontroller): Cortex-M0, … Cortex-M7

ARM Examples

- Raspberry Pi 2
  - Cortex-A7 (Broadcom BCM2836), 900 MHz, 1GiB RAM
- Arduino Due
  - Cortex-M3 (Atmel SAM3X8E), 84 MHz, 512 KiB Program Flash, 96 KiB SRAM
- Teensy 3.2
  - Cortex-M4 (Freescale MK20DX256VLH7), 72 MHz, 256 KiB Program Flash, 64KiB SRAM
- Samsung Galaxy S6
  - Cortex-A57 (Exynos 5422), Quad Core, 2.1 GHz, 3 GiB RAM

ARM Instruction Sets

- По мере развития: …, ARMv5, ARMv6, …
- Текущий 32-битный набор: ARMv7
- Расширения:
  - NEON (SIMD аналогичный SSE)
  - Jazelle (исполнение java byte code)
  - FPU
  - Large Physical Address Extension (LPAE)
  - Vector FPU (vfpv4)
- ARMv8-A — 64 битный набор инструкций

Программная модель процессора

- Организация ОЗУ
- Организация ввода-вывода
- Регистры процессора (общего назначения — РОН (GPR), специальные, управляющие)
- Флаги состояния процессора

ARMv7

- ОЗУ — фон-Неймановское, т. е. 32-битное пространство адресов, общее для программ и данных
- Адресация байтовая
  - Byte — 8 bit
  - Half-word — 16 bit
  - Word — 32 bit
  - Dword — 64 bit
- Данные должны выравниваться

Выравнивание

- Выравнивание — гарантирует размещение переменной (простого или сложного типа) так, чтобы адрес размещения был кратен размеру выравнивания
- Дополнение — добавление в структуру скрытых полей так, чтобы поля структуры были правильно выровнены

Невыровненные данные

- Недопустимы на некоторых платформах (попытка обращения вызовет Bus Error)
- На других платформах (x86, ARM) обращение к невыровненным данным требует два цикла обращения к памяти вместо одного
- Работа с невыровненными данными **не атомарна**

Базовые типы и их свойства

type	X86 Linux		ARM Linux (gnueabihf)	
	size	alignment	size	alignment
`char`	1	1	1	1
`short`	2	2	2	2
`int`	4	4	4	4
`long`	4	4	4	4
`long long`	8	4	8	8
`void *`	4	4	4	4
`float`	4	4	4	4
`double`	8	4	8	8
`long double`	12	4	8	8

Пример:

```
struct s {
    char f1;
    long long f2;
    char f3;
};
```

- X86: sizeof(s) == 16
- X64: sizeof(s) == 24
- ARM: sizeof(s)==24

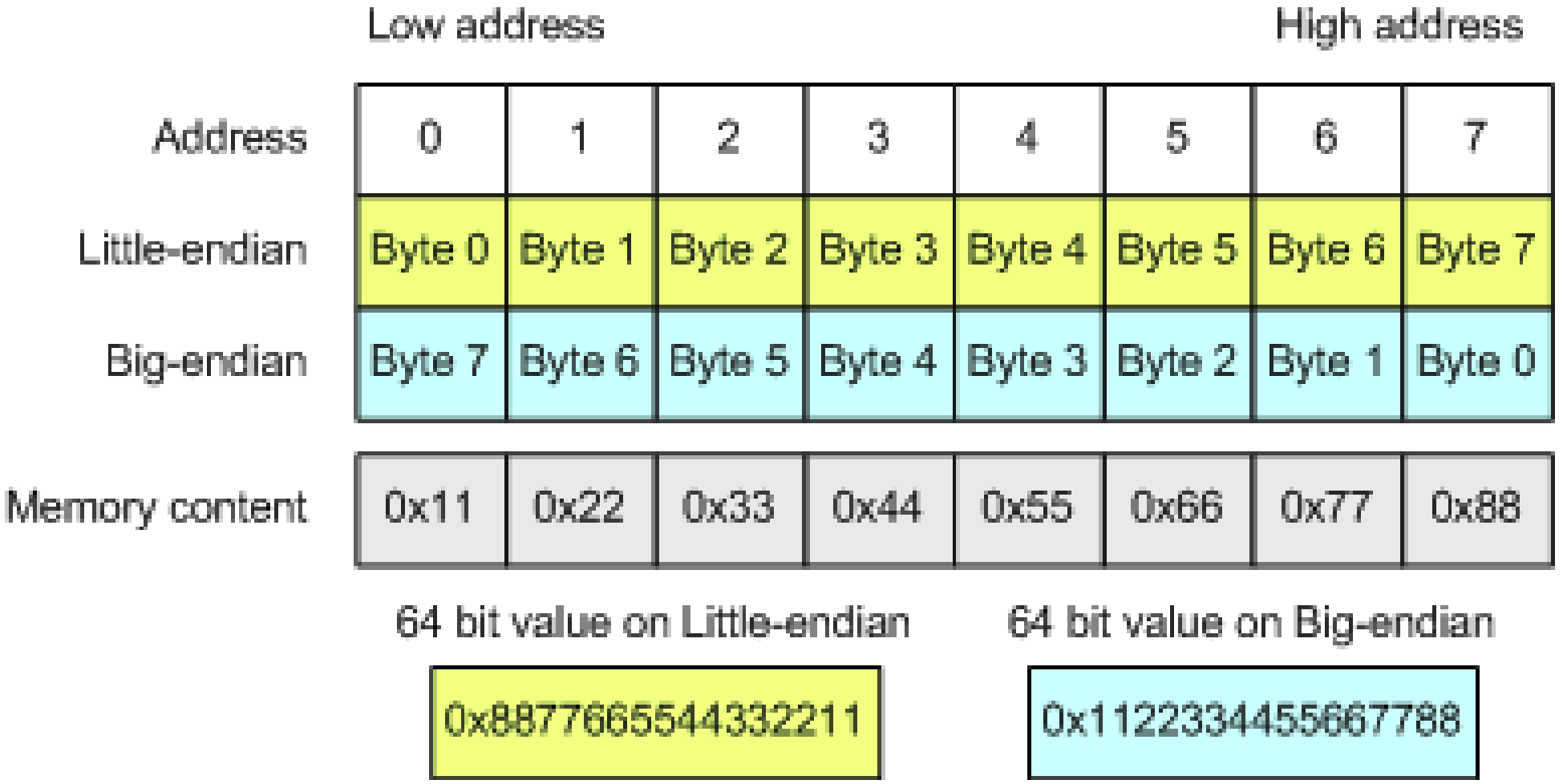
```
struct s {
    long long f2;
    char f1;
    char f3;
};
```

- X86: sizeof(s) == 12
- X64: sizeof(s) == 16
- ARM: sizeof(s) == 16

Byte order

- Память адресуется побайтно
- Целые числа большей длины могут размещаться в памяти по-разному
- Преобразование прозначно для программиста
- Little-endian: x86
- Big-endian: SPARC
- Переключаемые: ARM, PPC (Android — LE, iOS - LE)

Byte order


Low address
High address
Address
0
1
2
3
4
5
6
7
Little-endian
Byte 0
Byte 1
Byte 2
Byte 3
Byte 4
Byte 5
Byte 6
Byte 7
Big-endian
Byte 7
Byte 6
Byte 5
Byte 4
Byte 3
Byte 2
Byte 1
Byte 0
Memory content
0x11
0x22
0x33
0x44
0x55
0x66
0x77
0x88
64 bit value on Little-endian
0x8877665544332211
64 bit value on Big-endian
0x1122334455667788

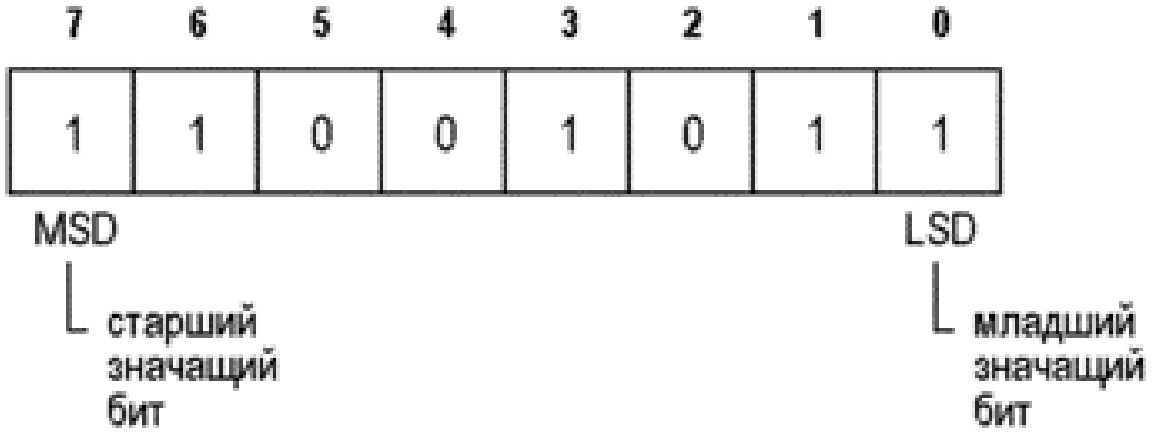
Типы данных

- Беззнаковые целые
- Знаковые целые
- Адреса — беззнаковые целые
- Смещения — знаковые целые
- Вещественные

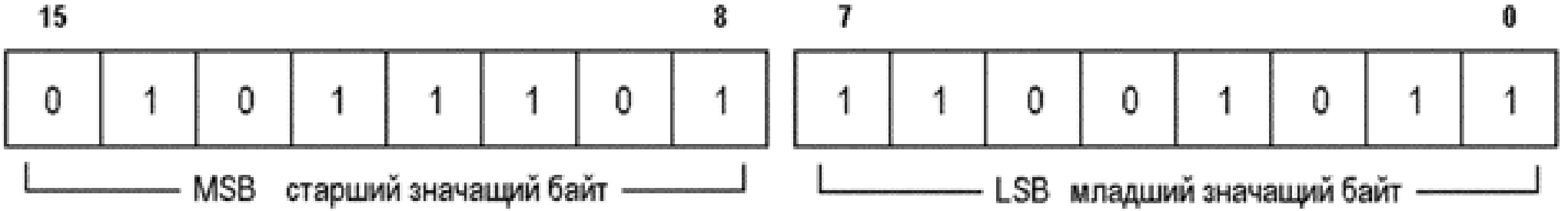
Беззнаковые типы

- Биты нумеруются от младшего к старшему

Байт (8 бит)



Слово (16 бит)

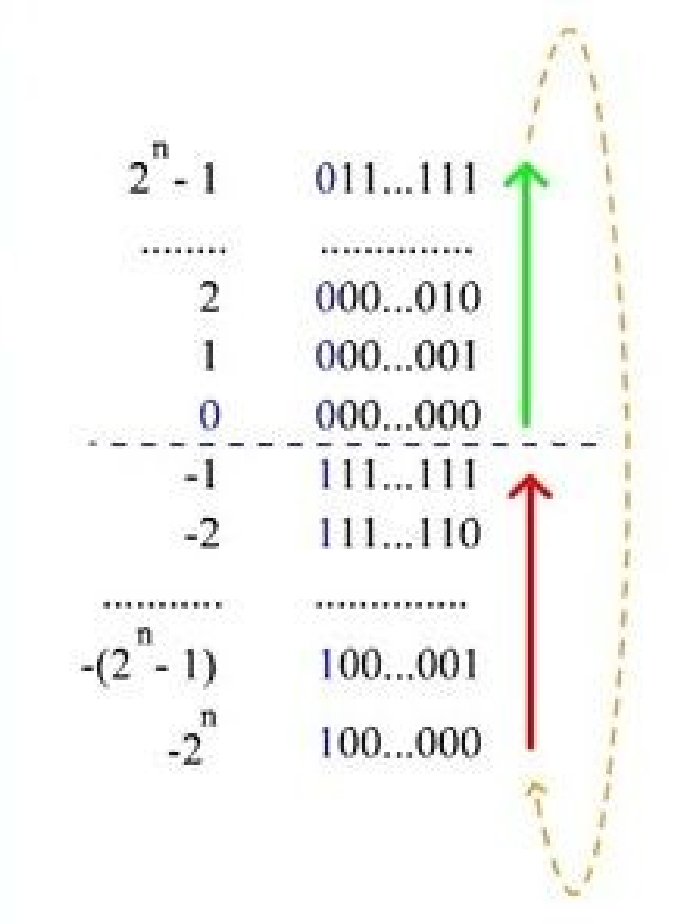


Беззнаковые типы (ARM)

- Byte (0 .. 255) — unsigned char
- Half-Word (0 .. 65536) — unsigned short
- Word (0 .. 4294967296) — unsigned int
- Dword (0 .. 18446744073709551616) — unsigned long long

Знаковые целые числа

- Могут иметь длину 8, 16, 32, 64 бита
- Представляются в дополнительном коде
- -x == ~x + 1
- Сложение, вычитание, сдвиг влево выполняются одинаково для знаковых и беззнаковых целых



Регистры общего назначения

- Используются для
  - Хранения значений наиболее часто используемых локальных переменных
  - Временного размещения аргументов и результатов арифметических, логических инструкций
  - Временного размещения параметров косвенного доступа к памяти

ARM GPR

- 16 32-битных регистров R0 .. R15
- R15 — PC — адрес инструкции для выполнения (почти, но не совсем)
- R14 — LR — регистр связи
- R13 — SP — указатель стека

Структура программы

- Программа — последовательность директив и инструкций процессора
- Директива или инструкция могут быть помечены
  main:
- Метка — символическое обозначение адреса в памяти или константы, то есть значение всех меток известно на момент загрузки программы на выполнение

Директивы

- Начинаются с точки (.)
- Общий вид:
  [LABEL:] DIRECTIVE OPERANDS
- Если директива задает данные, они размещаются в памяти по «текущему» адресу

Секции файла

- Объектный и исполняемый файл состоит из секций
- Самые базовые секции:
  - .text — секция кода (read-only)
  - .data — секция инициализированных данных, начальные значения явно задаются в программе
  - .bss — секция данных, инициализированных нулями — в исполняемом файле место не требуется

Точка входа

- Для freestanding-программ точка входа обычно называется _start
- Если мы используем драйвер gcc для компиляции, то точка входа — main
- Метки должны быть объявлены глобальными, чтобы быть доступными извне
  .global main

Данные

- .byte, .short (.hword), .int (.word, .long), .quad, .float, .double — объявление данных, например
  ```
  var:  .int   10
  ```
- .ascii, .asciz — строки (\0-terminated), например
  ```
  str:   .asciz   «Hello\n»
  ```
- .space — резервирование данных
  ```
  arr:   .space  64, 0
  ```
- .align — выравнивание данных
  ```
      .align   8
  ```

Вызовы подпрограмм

- Инструкция вызова подпрограмм:
  bl LABEL
- Значение PC, на которое нужно вернуться, копируется в LR, адрес подпрограммы загружается в PC
- Поскольку старое значение LR затирается, любая функция (кроме листовых) должна сохранить LR на входе и восстановить на выходе