Есть ли у вас вопросы?

Краткое содержание предыдущей серии

- Что такое ассемблер?
- Что такое процессорные регистры и зачем они нужны?
- Как организована память?
- Где хранится код?
- А как он при этом выглядит?

Краткое содержание сегодняшней серии

- Системы счисления
- Что такое «архитектура» компьютера
- Регистры в ARM Cortex M3
- Набор команд Thumb 2
- Способы адресации в ARM Cortex M3

Системы счисления

Что такое система счисления?

Метод записи чисел.

В зависимости от основания системы одна и та же запись означает разное число:

$100_5 = 25_{10}$

$100_2 = 4_{10}$

Двоичная система

Широко применяется в информатике, программировании и т.д.

Но почему?

Потому что компьютеры очень удобно делать из двоичных электронных компонентов.

Двоичная система

Почему люди не пользуются двоичной системой?

- По историческим причинам
- Относительно небольшие числа в ней записываются очень длинно:

$10\ 0111\ 0000\ 1111_2 = 9999_{10}$

А программисту зачем двоичная система?

В низкоуровневом программировании бывает так, что число не несет количественного смысла.

Вместо этого, каждый *бит в двоичном представлении числа* имеет свой смысл.

Что делать, если часто нужно считать в двоичной системе?

Нужно научиться быстро переводить числа из одной системы в другую.

Как?

- Быстро делить в уме
- Использовать специальный софт (напр., калькулятор windows)
- Использовать шестнадцатеричную или восьмеричную систему

Шестнадцатеричная система (hexadecimal)

В ней 16 цифр, 0..9, A,B,C,D,E и F.

В языке C такие числа записываются с префиксом 0x:

0xFF

Но что все это дает?

Шестнадцатеричная система (hexadecimal)

16 – это 2^4, поэтому каждая цифра в hex'е – это 4 цифры в двоичной системе!

И переводить числа очень легко:

0x1532

0001 0101 0011 0010

5426_{10} перевести в двоичную систему в уме сложнее

Если вам не легко

То нужно научится считать в двоичной системе от 0 до F:

binary	decimal	hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

binary	decimal	hex
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

В языке C

В языке C можно использовать три системы счисления:

- Десятичную – просто числа, без префиксов
- Шестнадцатеричную – числа с префиксом `0x`
- Восьмеричную – с префиксом `0`
- В некоторых компиляторах есть нестандартный префикс `0b` для двоичной системы

Подвох

```
static const struct _tag_cpltbl
{
  unsigned cp;
  const char* mimecp;
} cptbl[] =
{
  {   037, "IBM037" },     // IBM EBCDIC US-Canada
  {   437, "IBM437" },     // OEM United States
  {   500, "IBM500" },     // IBM EBCDIC International
  {   708, "ASMO-708" },   // Arabic (ASMO 708)
  ...
}
```

Программист для красоты выровнял колонку цифр.

И десятичное 37 превратилось в восьмеричное 037 == 31.

ARM

- Advanced RISC Machines – британская компания
- ARMv1.. ARMv9 – архитектуры ЭВМ
- ARM2...ARM11 и Cortex – микропроцессорные ядра

ARM не производит физических устройств, только спецификации

Мы изучаем микроконтроллер STM32F103

- ST Microelectronics – производитель чипа
- 32 – «битность»
- F103 - серия

Cortex M3 – ядро (M – означает микроконтроллер) с архитектурой ARMv7

Что же такое «архитектура компьютера»?

Это сочетание многих системных решений об устройстве компьютера, концептуальная структура, которая включает в себя

- Набор ассемблерных команд (instruction set)
- «Битность»
- Тип памяти (Гарвард или фон Нейман)
- Количество и назначение шин
- Общие принципы работы
- И т. д.

Что такое «битность»?

32-битный компьютер обладает:

- 32-битными регистрами
- 32-битной шиной адреса
- 32-битной шиной данных

Возможно, не все сразу – т.е. понятие это несколько расплывчато.

Типы архитектур

CISC – Complex Instruction Set Computer

RISC – Restricted Instruction Set Computer

Типы архитектур

Название	CISC	RISC
Пример	Intel x86,	ARM, MIPS, AVR, Nios, Blackfin...
Длина команд	Очень разная	Одинаковая или близкая
Время выполнения команд	Очень разное	Одинаковое или близкое
Кол-во команд	любое	любое
Регистров общего назначения	Мало	Много
Специализированных регистров	Много	Мало
Работа с памятью	Разнообразная	Только записать/прочитать
Встроенных типов данных	Много	Мало
Методов адресации	Много	Мало

Архитектура ARMv7

- RISC-подобная
- Набор команд Thumb-2 (совместим с Thumb из ARMv4)
- 32 бита
- фон Неймановская память (единое адресное пространство)
- 13 регистров общего назначения
- Endianness (и еще многое) на выбор производителя

Набор команд Thumb 2

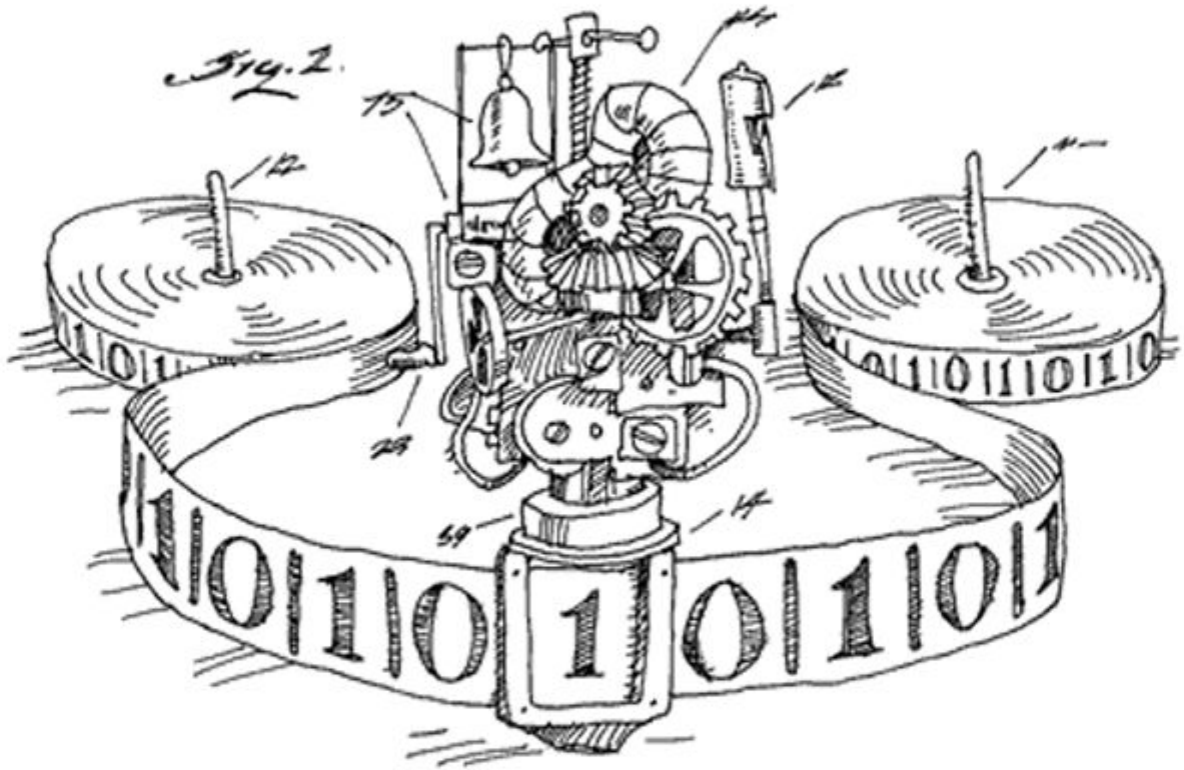
- Большая часть команд длиной 2 байта, есть команды в 4 байта (мнемоники единообразные)
- Большая часть команд выполняется за 2 такта (дольше – умножение, деление, множественная загрузка/сохранение и т.д.)
- Только целочисленная арифметика

Подробности - ARMv7-M Architecture Reference Manual

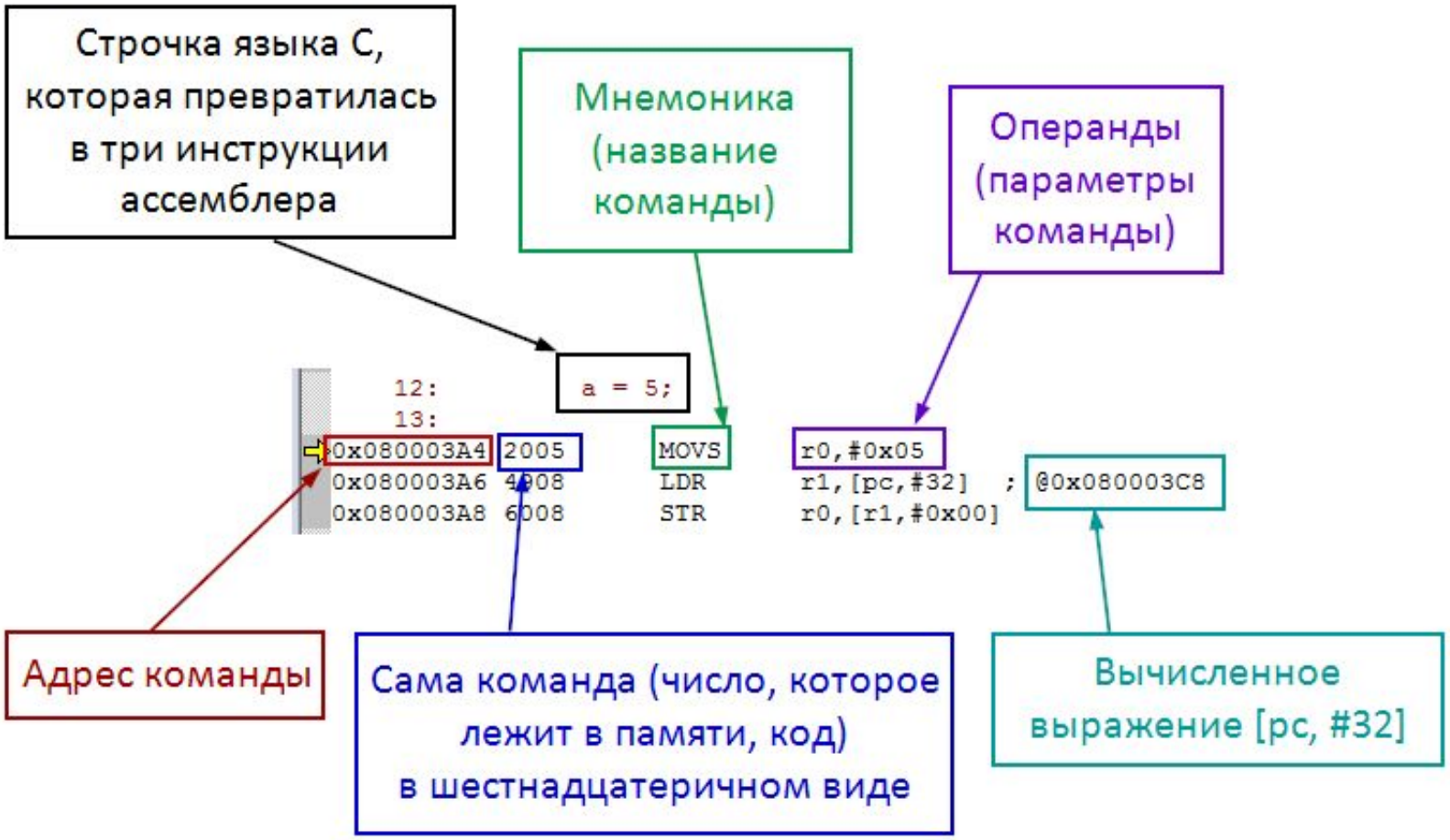
Регистры в ARM Cortex M3

Регистр	Название	Зачем он нужен
Регистры общего назначения	R0, R1, R2...R12	Как заранее созданные переменные для ассемблера
Указатель стека - Stack Pointer	R13 (SP)	Хранит адрес вершины стека
Регистр связи – Link Register	R14 (LR)	Хранит адрес возврата при вызове функции
Счетчик команд – Program Counter	R15 (PC)	Хранит адрес текущей команды
Регистры состояния	PSR, ASPR, IPSR, EPSR, PRIMASK, FAULTMASK, BASEPRI, CONTROL	Каждый бит в этих регистрах указывает на какое-нибудь событие (произошло или нет)

Помните машину Тьюринга?



Как выглядит ассемблер в Keil



Команда movs r0, #0x05 – поместить (от слова move) в регистр R0 число 5

Подробнее о команде

Итак, `0x080003A4 2005 MOVS r0,#0x05`

означает «по адресу `0x8003A4` хранится: положить в регистр `R0` число 5»

А откуда берется число пять?

Оно лежит прямо в коде команды! 20**05**

Это называется «непосредственная» адресация (immediate)

Подробнее о команде

Итак, `0x080003A4 2005 MOVS r0,#0x05`
означает «по адресу `0x8003A4` хранится: положить в регистр `R0` число 5»

А где написано, что `R0`?
Тоже в команде! 2**0**05

Это «регистровая адресация» – один из операндов – регистр и его номер указан прямо в команде.

Структура команды (на примере 16-битной mov)

MOV (1) (Move) moves a large immediate value to a register.

- 0x2005 = 0010 0000 0000 0101
- Пять старших бит показывают что это, собственно, команда mov
- Биты 10, 9 и 8 задают номер регистра
- Биты с 7 по 0 задают непосредственный операнд

Поразительным образом все, что делает команда, указано прямо в команде!

А какая вообще бывает адресация?

Название	Суть
Непосредственная	в команде лежит сам операнд
Регистровая	в команде указан регистр, в котором лежит операнд
Абсолютная (прямая)	в команде лежит адрес переменной
Косвенно-регистровая (со смещением)	в команде указан регистр, в котором лежит адрес переменной (+/- смещение)
Косвенно-регистровая с автомодификацией	в регистре лежит адрес, который автоматически меняется после выполнения команды

Косвенность, теоретически, может наращиваться бесконечно - по адресу лежит адрес, по которому лежит адрес, по которому...

(указатель на указатель на указатель...)

Абсолютная адресация в Cortex M3

Команды имеют длину 2 или 4 байта.

Адреса имеют длину 4 байта.

Как же положить адрес прямо в команду?

А никак. В Cortex M3 такого способа адресации нет!

Что же делать?

Косвенно-регистровая адресация

- Синтаксис – квадратные скобки

LDR R0, [R1,#0x00] – считать в регистр R0 значение, лежащее по адресу «то, что в R1 + 0»

- Смещение лежит прямо в команде и может быть от 0 до 4095.

- Для адресации глобальных переменных часто используется регистр PC. Но почему?

Немножко о компиляции языка C

- Каждый файл .c компилируется отдельно от остальных и превращается в файл .o – «объектный файл»

- Линкер разрешает зависимости между файлами, проставляет вызовы функций

- Линкер размещает объектные файлы в памяти

Вывод?

Линкеру удобно положить адрес переменной рядом с кодом, который к ней обращается. Если они в одном файле.

А где в ассемблере учитывается тип?

У команд есть вариации:

- W – word – слово – 4 байта
- H – halfword – полуслово – 2 байта
- B – byte – байт – 1 байт
- S – signed – знаковый