Кодирование команд. Вещественная арифметика

АКОС ФИВТ ПМИ - Лекция №3

RISC v.s. CISC

- CISC: переменное количество байт на команду
- RISC: размер команды - машинное слово
- Для гарвардиких архитектур может быть два размера машинных слов (AVR)

Кодирование инструкций AVR

ADD – Add without Carry

Description

Adds two registers without the C Flag and places the result in the destination register Rd.

	Operation:		
(i)	(i) Rd ← Rd + Rr		
	Syntax:	Operands:	Program Counter:
(i)	ADD Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0000	11rd	dddd	rrrr

Кодирование инструкций AVR

LDI – Load Immediate

Description

Loads an 8-bit constant directly to register 16 to 31.

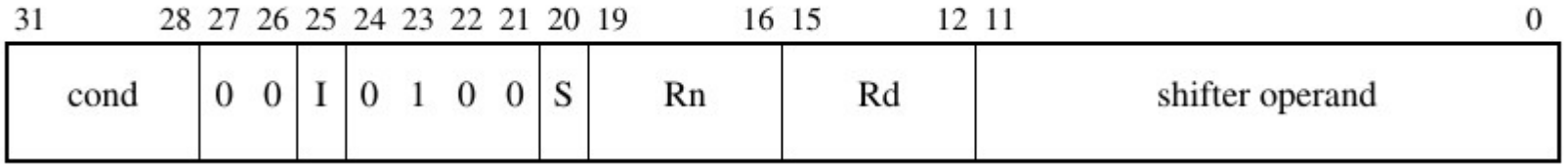
	Operation:		
(i)	Rd ← K		
	Syntax:	**Operands:**	**Program Counter:**
(i)	LDI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

1110	KKKK	dddd	KKKK

Кодирование ARM-32

ADD



- **cond** - условие выполнения команды
- **I** - флаг, определяющий, что закодировано в **shifter_operand**
- **S** - флаг, определяющий, нужно ли обновлять регистр статуса
- **Rn**, **Rd** - первый аргумент операции и куда записать результат

Примеры:

```
ADD r0, r1, r2  // r0 ← r1 + r2
ADDS    r0, r1, r2  // r0 ← r1 + r2, обновить флаги Z,V,C,N
ADDEQ   r0, r1, r2  // Если Z, то r0 ← r1 + r2
ADD r0, r1, r2, lsl #3  // r0 ← r1 + (r2 << 3)    [5bit shift; 2bit type; 0; 4bit reg]
ADD r0, r1, #0xFF  // r0 ← r1 + 0b0000'1111'1111
ADD r0, r1, #0x200 // r0 ← r1 + (0b0000'0010 ROR 0b1100)
```

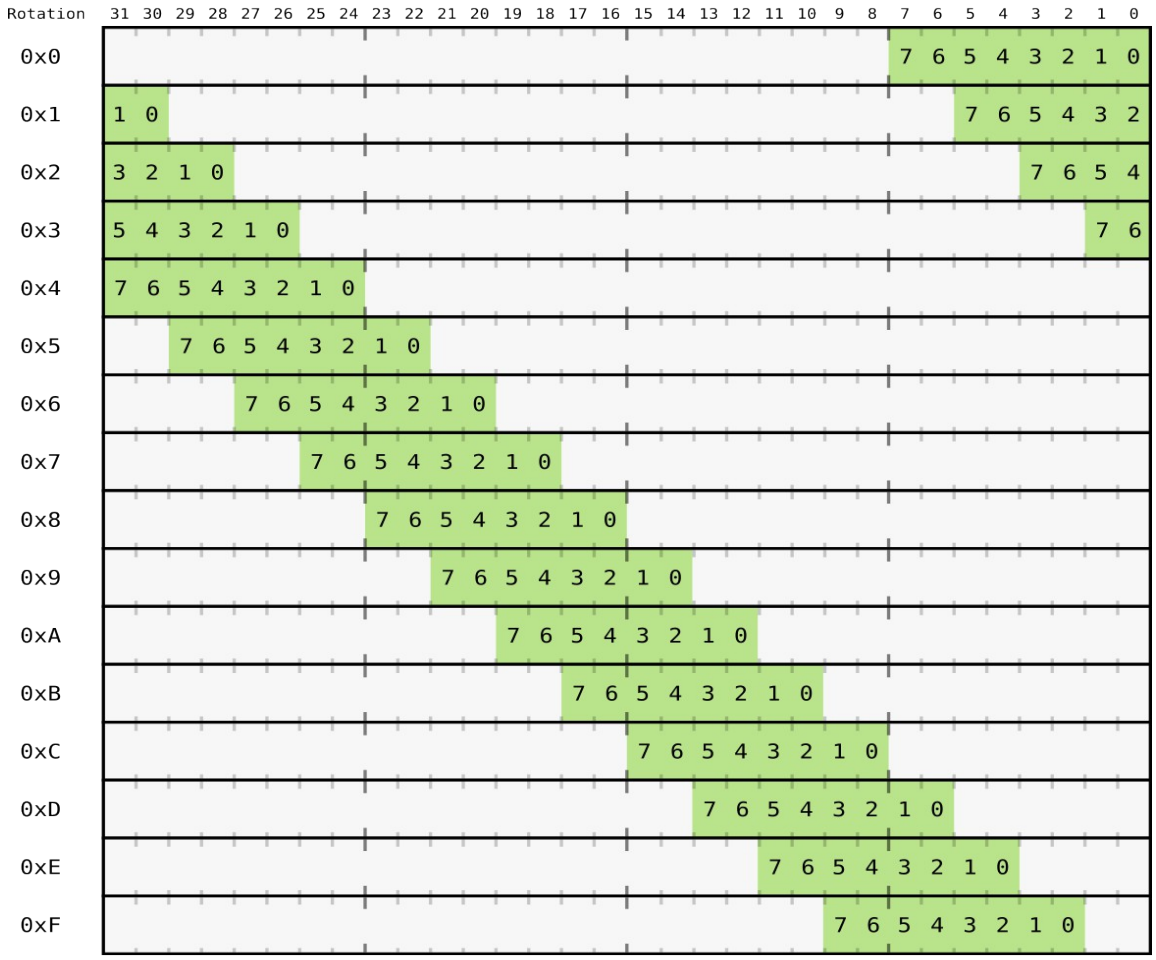
Циклический сдвиг

ADD r0, r1, #0x200

```
0x200 = 512 =
0b10'0000'0000

Lo(Instr) =
0b1100'0000'0010
  rot  immediate

0b1100 = 0xC
0b0000'0010
```

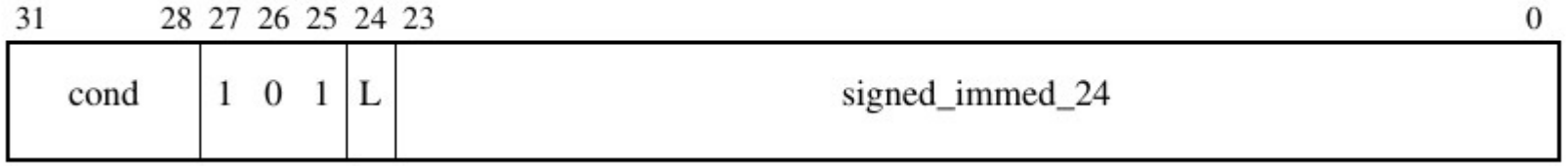


Условное выполнение команд

Opcode [31:28]	Mnemonic extension	Meaning	Condition flag state
0000	EQ	Equal	Z set
0001	NE	Not equal	Z clear
0010	CS/HS	Carry set/unsigned higher or same	C set
0011	CC/LO	Carry clear/unsigned lower	C clear
0100	MI	Minus/negative	N set
0101	PL	Plus/positive or zero	N clear
0110	VS	Overflow	V set
0111	VC	No overflow	V clear
1000	HI	Unsigned higher	C set and Z clear
1001	LS	Unsigned lower or same	C clear or Z set
1010	GE	Signed greater than or equal	N set and V set, or N clear and V clear (N == V)
1011	LT	Signed less than	N set and V clear, or N clear and V set (N != V)
1100	GT	Signed greater than	Z clear, and either N set and V set, or N clear and V clear (Z == 0,N == V)
1101	LE	Signed less than or equal	Z set, or N set and V clear, or N clear and V set (Z == 1 or N != V)
1110	AL	Always (unconditional)	-

Кодирование ARM-32

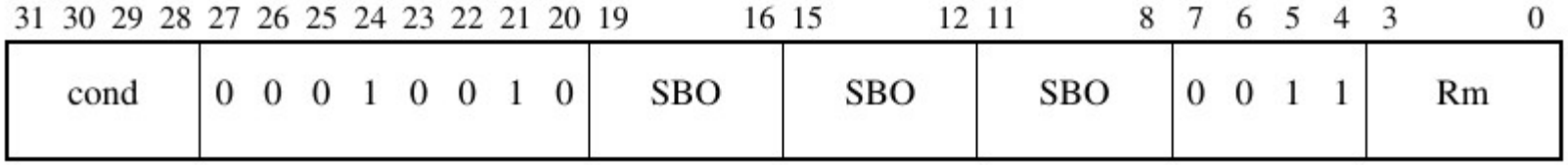
B, BL



+/- 2^{23} - это +/- 8Мб

Branch with Link and eXchange

BLX (2)



BLX (2) calls an ARM or Thumb subroutine from the ARM instruction set, at an address specified in a register.

It sets the CPSR T bit to bit[0] of Rm. This selects the instruction set to be used in the subroutine.

The branch target address is the value of register Rm, with its bit[0] forced to zero.

It sets R14 to a return address. To return from the subroutine, use a `BX R14` instruction, or store R14 on the stack and reload the stored value into the PC.

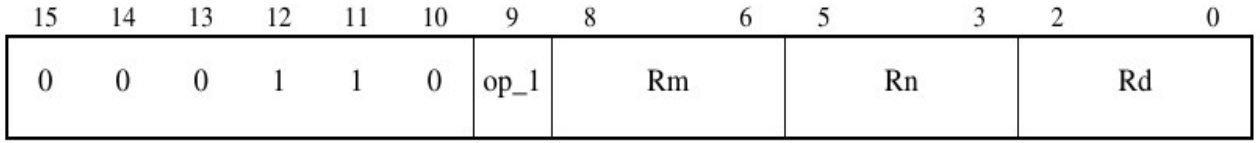
Расширения ARM

- Thumb -- 16 битные инструкции
- Jazelle -- декодирование байткода Java

Кодирование команд Thumb

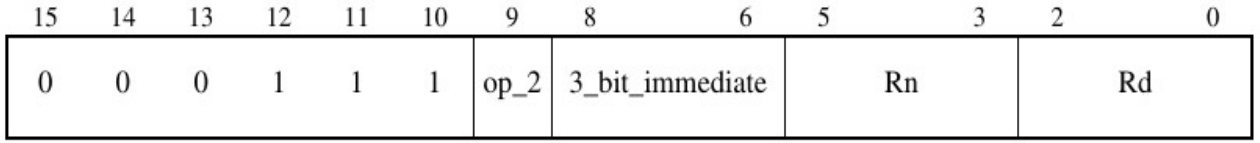
Format 1

```
<opcode1>  <Rd>, <Rn>, <Rm>
<opcode1> := ADD | SUB
```



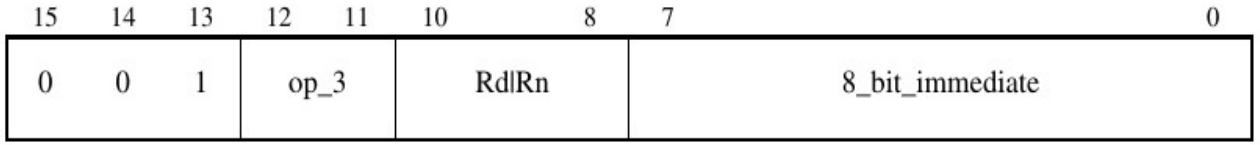
Format 2

```
<opcode2>  <Rd>, <Rn>, #<3_bit_immed>
<opcode2> := ADD | SUB
```



Format 3

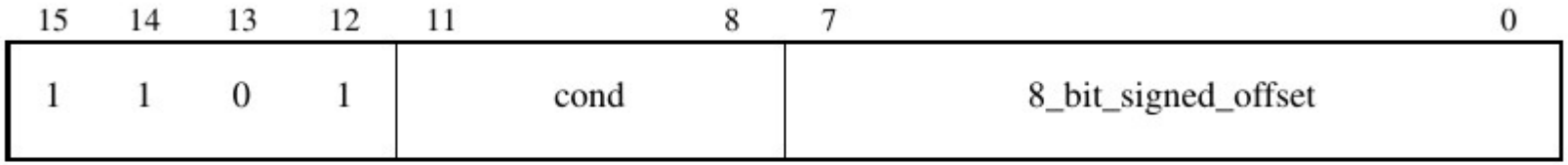
```
<opcode3>  <Rd>|<Rn>, #<8_bit_immed>
<opcode3> := ADD | SUB | MOV | CMP
```



Кодирование команд Thumb

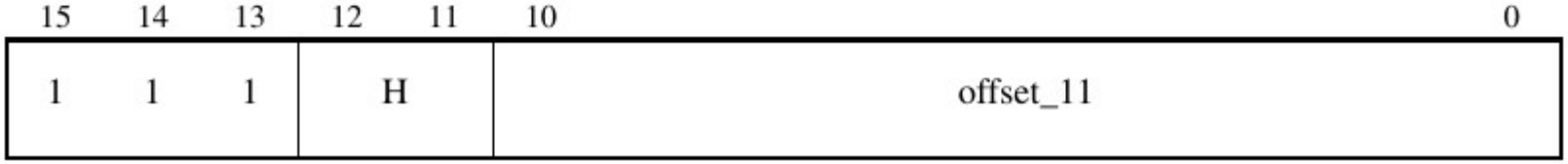
Conditional branch

```
B<cond>  <target_address>
```

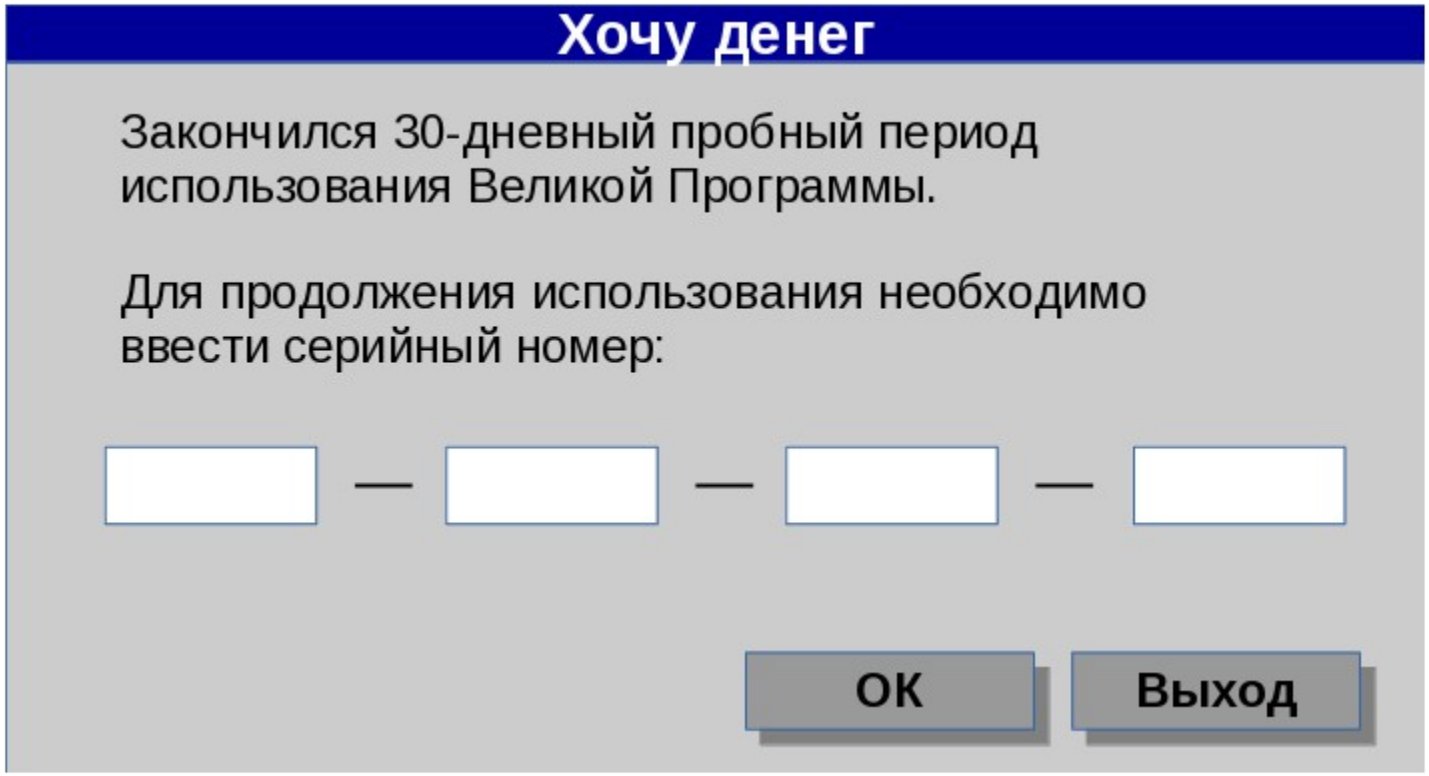


Unconditional branch

```
B    <target_address>
BL   <target_address>          ; Produces two 16-bit instructions
BLX  <target_address>          ; Produces two 16-bit instructions
```



Зачем понимать кодирование команд



Уголовный кодекс РФ, глава 21 «Преступления против собственности»

Статья 159.6. Мошенничество в сфере компьютерной информации

1. Мошенничество в сфере компьютерной информации, то есть хищение чужого имущества или приобретение права на чужое имущество путем ввода, удаления, блокирования, модификации компьютерной информации либо иного вмешательства в функционирование средств хранения, обработки или передачи компьютерной информации или информационно-телекоммуникационных сетей, - наказывается штрафом в размере до ста двадцати тысяч рублей или в размере заработной платы или иного дохода осужденного за период до одного года, либо обязательными работами на срок до трехсот шестидесяти часов, либо исправительными работами на срок до одного года, либо ограничением свободы на срок до двух лет, либо принудительными работами на срок до двух лет, либо арестом на срок до четырех месяцев.

Как ломают

```
void TrialEndDialog_OkPressed()
{
  const char * sn =
Dialog_GetSerialNumber();
  if (CheckSerialNo(sn)) {
    ContinueLaunch();
  }
  else {
    ShowErrorMessage();
    ExitProgram();
  }
}
```

Как ломают

```
. . . . . .
01: callq   CheckSerialNo    ;; вызов [bool CheckSerialNo]
02: testb   $1, %al          ;; сравнение результата с 1
03: jejmp   $05              ;; если OK, то переход к 05
04: jmp     $08              ;; переход к 08
05: movb$0, %al              ;; очистка регистра AL
06: callq   ContinueLaunch   ;; вызов [bool ContinueLaunch]
07: jmp     $0C              ;; переход к строке после if {}
08: movb$0, %al              ;; очистка регистра AL
09: callq   ShowErrorMessage ;; вызов [void ShowErrorMessage]
0A: movb$0, %al              ;; очистка регистра AL
0B: callq   ExitProgram      ;; вызов [void ExitProgram]
. . . . . .
```

Как ломают

```
. . . . . .
01: callq    CheckSerialNo     ;; вызов [bool CheckSerialNo]
02: testb    $1, %al           ;; сравнение результата с 1
03: je       $05               ;; если OK, то переход к 05
04: jmp      $08               ;; переход к 08
05: movb$0, %al                ;; очистка регистра AL
06: callq    ContinueLaunch    ;; вызов [bool ContinueLaunch]
07: jmp      $0C               ;; переход к строке после if {}
08: movb$0, %al                ;; очистка регистра AL
09: callq    ShowErrorMessage  ;; вызов [void ShowErrorMessage]
0A: movb$0, %al                ;; очистка регистра AL
0B: callq    ExitProgram       ;; вызов [void ExitProgram]
. . . . . .
```

Вещественная арифметика

IEEE 754

Представление IEEE754

Single-Precision (32 бит); B = 127		
S	E (8 бит)	M (23 бит)

Double-Precision (64 бит); B = 1023		
S	E (11 бит)	M (52 бит)

$$Value = (-1)^{S} \cdot 2^{E-B} \cdot (1 + M / 2^{52})$$

Специальные значения

S	E	M	Значение
0	0	0	+0
1	0	0	−0
0	11...11	0	+ ∞
1	11...11	0	− ∞
0	11...11	≠ 0	Signaling NaN
1	11...11	≠ 0	Quiet NaN
0	0	≠ 0	Денормализованные значения
1	0	≠ 0	

Денормализованные числа

Single-Precision (32 бит)		
S	0000 0000	M (23 бит)

Double-Precision (64 бит)		
S	000 0000 0000	M (52 бит)

$$Value = (-1)^S \cdot M / 2^{52}$$

Операции: умножение

$<S,E,M> = <S_1,E_1,M_1> \cdot <S_1,E_1,M_2>$

1. Вычислить

$S = S_1 \text{ ^ } S_2$

$E = E_1+E_2$

$M = 1.M_1 \cdot 1.M_2$

2. { M>>=1 ; E++ }пока переполнение M

Операции: сложение

$<S,E,M> = <S_1,E_1,M_1> \cdot <S_1,E_1,M_2>$

1. Вычислить $E_{diff}=E_1-E_2$
2. Нормализовать M_2 на E_{diff} бит
3. Значения:

   $E = E1$

   $M = M1 +/- M2$

   $S = sign(-1^{S1}M_1 + -1^{S2}M_2)$

Реализации FPU

- Расширенный набор команд (ARM VFP):
  - Дополнительные команды
  - Дополнительные 32 регистра
- Сопроцессор (x86):
  `(gcc -mfpmath=387)`
  - Команды, которые CPU делегирует FPU
  - Взаимодействие через стек
- Команды SSE (Pentium-III+, x86-64):
  `(gcc -msse -mfpmath=sse)`
  - Используются регистры xmm
  - Используются скалярные команды SSE

Точность

Floating Point

- Single Precision - 32bit ≈ 10^{37}
- Double Precision - 64bit ≈ 10^{307}

Fixed Point