Архитектура ЭВМ

Лекция 2. GaLyA

Филонов Павел
filonovpv@gmail.com

Московский Государственный Технический Университет
Гражданской Авиации

2021 г.

О чём поговорим сегодня?

О чём поговорим сегодня?

1. О некоторых CC

О чём поговорим сегодня?

1. О некоторых CC
2. "Вычитать и умножать, малышей не обижать учат в школе, учат в школе, учат в школе"

О чём поговорим сегодня?

1. О некоторых CC
2. "Вычитать и умножать, малышей не обижать учат в школе, учат в школе, учат в школе"
3. KiD хорош, но GaLyA лучше

О чём поговорим сегодня?

1. О некоторых CC
2. "Вычитать и умножать, малышей не обижать учат в школе, учат в школе, учат в школе"
3. KiD хорош, но GaLyA лучше
4. Поднять флаги!

О чём поговорим сегодня?

1. О некоторых CC
2. "Вычитать и умножать, малышей не обижать учат в школе, учат в школе, учат в школе"
3. KiD хорош, но GaLyA лучше
4. Поднять флаги!
5. Прыг-скок и условные переходы

О чём поговорим сегодня?

1. О некоторых CC
2. "Вычитать и умножать, малышей не обижать учат в школе, учат в школе, учат в школе"
3. KiD хорош, но GaLyA лучше
4. Поднять флаги!
5. Прыг-скок и условные переходы
6. Никаких `if`, `for`, `while`, только `jump`, только hardcore!

СС бывают разные

Десятичная — наша любимая

No comments

Двоичная — любимица современных ЭВМ

$11001101_2 =$
$1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 205$
$75 = 64 + 8 + 2 + 1 = 2^6 + 2^3 + 2^1 + 2^0 = 1001011b$

Восьмеричная — встречается в кодах прав доступа UNIX

$$0755 = 0111101101b \text{ — rwxr-xr-x}$$

Шестнадцатеричная — «Компромисс» между десятичной и двоичной

0x3F = 00111111b = 077 = 63
A1h = 10100001b = 0241 = 161

Степени двойки

i	2^i	2^ib	02^i	$0x2^i$
0	1	1b	01	0x1
1	2	10b	02	0x2
2	4	100b	04	0x4
3	8	1000b	010	0x8
4	16	10000b	020	0x10
5	32	100000b	040	0x20
6	64	1000000b	0100	0x40
7	128	10000000b	0200	0x80
8	256	100000000b	0400	0x100
9	512	1000000000b	01000	0x200
10	1024	10000000000b	02000	0x400

Правило тетрад

N_{10}	N_2	N_{16}
0	0000b	0x0
1	0001b	0x1
2	0010b	0x2
3	0011b	0x3
4	0100b	0x4
5	0101b	0x5
6	0110b	0x6
7	0111b	0x7
8	1000b	0x8
9	1001b	0x9
10	1010b	0xA
11	1011b	0xB
12	1100b	0xC
13	1101b	0xD
14	1110b	0xE
15	1111b	0xF

$$0xA5 = \underbrace{1010}_{0xA}\underbrace{0101}_{0x5}\,b$$

$$\underbrace{0110}_{0x6}\underbrace{1101}_{0xD}\,b = 0x6D$$

Правило тетрад

N_{10}	N_2	N_{16}
0	0000b	0x0
1	0001b	0x1
2	0010b	0x2
3	0011b	0x3
4	0100b	0x4
5	0101b	0x5
6	0110b	0x6
7	0111b	0x7
8	1000b	0x8
9	1001b	0x9
10	1010b	0xA
11	1011b	0xB
12	1100b	0xC
13	1101b	0xD
14	1110b	0xE
15	1111b	0xF

$$0xA5 = \underbrace{1010}_{0xA}\underbrace{0101}_{0x5}\,b$$

$$\underbrace{0110}_{0x6}\underbrace{1101}_{0xD}\,b = 0x6D$$

$$0x3C =$$

Правило тетрад

N_{10}	N_2	N_{16}
0	0000b	0x0
1	0001b	0x1
2	0010b	0x2
3	0011b	0x3
4	0100b	0x4
5	0101b	0x5
6	0110b	0x6
7	0111b	0x7
8	1000b	0x8
9	1001b	0x9
10	1010b	0xA
11	1011b	0xB
12	1100b	0xC
13	1101b	0xD
14	1110b	0xE
15	1111b	0xF

$$0xA5 = \underbrace{1010}_{0xA}\underbrace{0101}_{0x5}\,b$$

$$\underbrace{0110}_{0x6}\underbrace{1101}_{0xD}\,b = 0x6D$$

$$0x3C = \underbrace{0011}_{0x3}\underbrace{1100}_{0xC}\,b$$

Правило тетрад

N_{10}	N_2	N_{16}
0	0000b	0x0
1	0001b	0x1
2	0010b	0x2
3	0011b	0x3
4	0100b	0x4
5	0101b	0x5
6	0110b	0x6
7	0111b	0x7
8	1000b	0x8
9	1001b	0x9
10	1010b	0xA
11	1011b	0xB
12	1100b	0xC
13	1101b	0xD
14	1110b	0xE
15	1111b	0xF

$$0xA5 = \underbrace{1010}_{0xA}\underbrace{0101}_{0x5}b$$

$$\underbrace{0110}_{0x6}\underbrace{1101}_{0xD}b = 0x6D$$

$$0x3C = \underbrace{0011}_{0x3}\underbrace{1100}_{0xC}b$$

$$01011011b =$$

Правило тетрад

N_{10}	N_2	N_{16}
0	0000b	0x0
1	0001b	0x1
2	0010b	0x2
3	0011b	0x3
4	0100b	0x4
5	0101b	0x5
6	0110b	0x6
7	0111b	0x7
8	1000b	0x8
9	1001b	0x9
10	1010b	0xA
11	1011b	0xB
12	1100b	0xC
13	1101b	0xD
14	1110b	0xE
15	1111b	0xF

$$0xA5 = \underbrace{1010}_{0xA}\underbrace{0101}_{0x5}\,b$$

$$\underbrace{0110}_{0x6}\underbrace{1101}_{0xD}\,b = 0x6D$$

$$0x3C = \underbrace{0011}_{0x3}\underbrace{1100}_{0xC}\,b$$

$$\underbrace{0101}_{0x5}\underbrace{1011}_{0xB}\,b = 0x5B$$

Машинная арифметика

Сначала всё в порядке

	0	0	1	0	1	0	1	1	= 43
+									
	0	0	0	1	1	1	1	0	= 30
	0	1	0	0	1	0	0	1	= 73

Машинная арифметика

Сначала всё в порядке

	0	0	1	0	1	0	1	1 = 43
+								
	0	0	0	1	1	1	1	0 = 30
	0	1	0	0	1	0	0	1 = 73

А теперь — сюрприз!

	0	1	1	0	1	0	1	1 = 107
+								
	1	0	0	1	1	1	1	0 = 158
1	0	0	0	0	1	0	0	1 = 9 !

Переполнение

Машинная арифметика

Сначала всё в порядке

	0	0	1	0	1	0	1	1	= 43
+									
	0	0	0	1	1	1	1	0	= 30
	0	1	0	0	1	0	0	1	= 73

А теперь — сюрприз!

	0	1	1	0	1	0	1	1	= 107
+									
	1	0	0	1	1	1	1	0	= 158
1	0	0	0	0	1	0	0	1	= 9 !

Переполнение

В качестве результата всех операций сложения берётся остаток от деления на 2^n, где n — число хранимых двоичных разрядов

$$a+b = (a+b) \mod 2^n$$

Машинная арифметика

Сначала всё в порядке

	0	0	1	0	1	0	1	1 = 43
+								
	0	0	0	1	1	1	1	0 = 30
	0	1	0	0	1	0	0	1 = 73

А теперь — сюрприз!

	0	1	1	0	1	0	1	1 = 107
+								
	1	0	0	1	1	1	1	0 = 158
1	0	0	0	0	1	0	0	1 = 9 !

Переполнение

В качестве результата всех операций сложения берётся остаток от деления на 2^n, где n — число хранимых двоичных разрядов

$$a+b = (a+b) \mod 2^n$$

$$107 + 158 = 265 \mod 2^8 = 9$$

Что сложнее — сложение или вычитание?

А можно ли упростить вычитание?

Что сложнее — сложение или вычитание?

А можно ли упростить вычитание?

$$245-109 = 245+1000-1000-109 = 245+999+1-109-1000 =$$
$$= 245 + (999 - 109) + 1 - 1000 = 245 + 890 + 1 - 1000 =$$
$$= 245 + 891 - 1000 = 1136 - 1000 = 136$$

Что сложнее — сложение или вычитание?

А можно ли упростить вычитание?

$$245-109 = 245+1000-1000-109 = 245+999+1-109-1000 =$$
$$= 245 + (999 - 109) + 1 - 1000 = 245 + 890 + 1 - 1000 =$$
$$= 245 + 891 - 1000 = 1136 - 1000 = 136$$

Что же мы сделали?

Дополнение разрядов 109 до 9 :$109 \to 999 - 109 = 890$

Дополнение числа 890 до 10 : $890 \to 890 + 1 = 891$

Сложение : $245 + 891 = 1136$

Простое вычитание : $1136 - 1000 = 136$

Дополнительный код

А как это выглядит в двоичном коде?

- **Прямой код** — $00110101b$
- Дополнение до 1 (инвертируются все биты) $00110101b \rightarrow 11001010b$ — **обратный код**
- Дополнение до 2 $(+1)$ $11001010b + 1 = 11001011b$ — **дополнительный код**
- Вычитание заменяется сложением с дополнительным кодом вычитаемого
- Переполнение срабатывает как вычитание 1 из старшего разряда (-1000 в предыдущем примере)

Дополнительный код

А как это выглядит в двоичном коде?

- **Прямой код** — $00110101b$
- Дополнение до 1 (инвертируются все биты) $00110101b \rightarrow 11001010b$ — **обратный код**
- Дополнение до 2 (+1) $11001010b + 1 = 11001011b$ — **дополнительный код**
- Вычитание заменяется сложением с дополнительным кодом вычитаемого
- Переполнение срабатывает как вычитание 1 из старшего разряда (-1000 в предыдущем примере)

Пример

$81 - 51 = 01010001b - 00110011b = 01010001b + 11001101b = (1)00011110b = 00011110b = 30$

Представление знаковых чисел

Правила для знаковых чисел

- Старший бит хранит знак числа (0 - '+', 1 - '-')
- Для положительных чисел дополнительный, обратный и прямой коды совпадают
- Для отрицательных чисел в дополнительном коде хранятся значащие разряды (все кроме старшего)

Пример

$-35 \rightarrow 10100011 \rightarrow 11011100 \rightarrow 11011101$
$101 \rightarrow 01100101$

Представление знаковых чисел

Десятичное представление	Прямой	Обратный	Дополнительный
127	01111111	01111111	01111111
1	00000001	00000001	00000001
0	00000000	00000000	00000000
-1	10000001	11111110	11111111
-2	10000010	11111101	11111110
-3	10000011	11111100	11111101
-4	10000100	11111011	11111100
-5	10000101	11111010	11111011
-6	10000110	11111001	11111010
-7	10000111	11111000	11111001
-8	10001000	11110111	11111000
-9	10001001	11110110	11110111
-10	10001010	11110101	11110110
-11	10001011	11110100	11110101
-127	11111111	10000000	10000001
-128	10000000	11111111	10000000

Диапазоны представления целых чисел

Размер (байт)	Беззнаковые	Знаковые
1	0..255	-128..127
2	0..65535	-32768..32767
3	0..16777215	-8366608..8366607
4	0..4294967295	-2147483648..2147483647

Грубое правило для оценки $2^{10} \sim 10^3$

$2^{16} \sim 10^3 * 2^6 = 64000$

$2^{24} \sim 10^6 * 2^4 = 16000000$

$2^{32} \sim 10^9 * 2^2 = 4000000000$

$2^{64} \sim 10^{18} * 2^4 = 16 * 10^{18}$

Разминка (кодирование со знаком)

1. $99 - 105 =$

Разминка (кодирование со знаком)

1. $99 - 105 = -6 = 11111010b = 0xFA$
2. $45 - 75 =$

Разминка (кодирование со знаком)

1. $99 - 105 = -6 = 11111010b = 0xFA$
2. $45 - 75 = -30 = 11100010b = 0xE2$
3. $10 - 111 =$

Разминка (кодирование со знаком)

1. $99 - 105 = -6 = 11111010b = 0xFA$
2. $45 - 75 = -30 = 11100010b = 0xE2$
3. $10 - 111 = -101 = 10011011b = 0x9B$
4. $120 + 35 =$

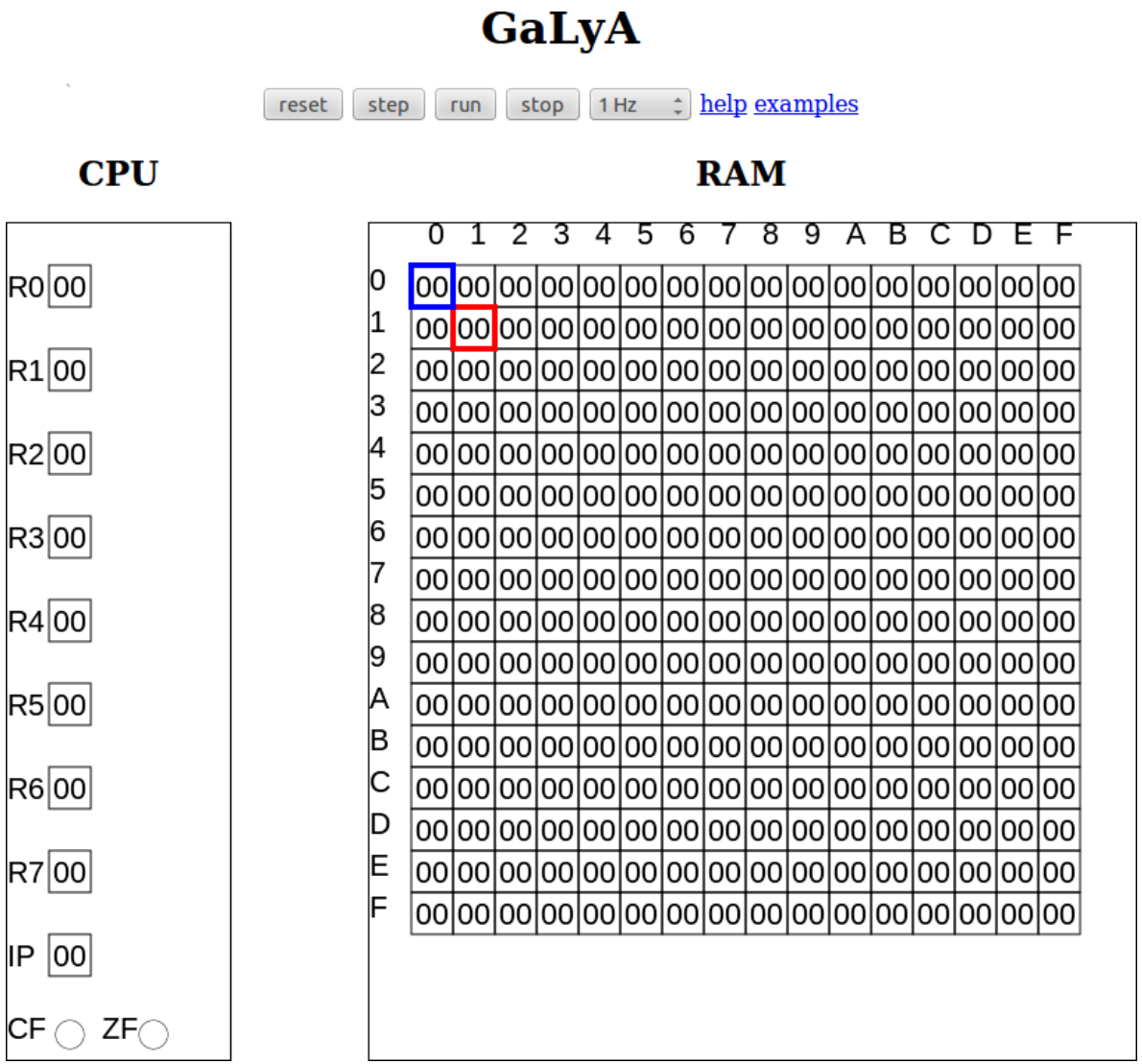
Разминка (кодирование со знаком)

1. $99 - 105 = -6 = 11111010b = 0xFA$
2. $45 - 75 = -30 = 11100010b = 0xE2$
3. $10 - 111 = -101 = 10011011b = 0x9B$
4. $120 + 35 = 155 = 10011011 = 0x9B = -101$
5. $-100 - 79 =$

Разминка (кодирование со знаком)

1. $99 - 105 = -6 = 11111010b = 0xFA$
2. $45 - 75 = -30 = 11100010b = 0xE2$
3. $10 - 111 = -101 = 10011011b = 0x9B$
4. $120 + 35 = 155 = 10011011 = 0x9B = -101$
5. $-100 - 79 = -179 = 77 \mod 256 = 01001101b = 0x4D$

GaLyA (http://cleric.su/galya)



Описание архитектуры

1. Система счисления — шестнадцатеричная
2. Разрядность регистров — 8 бит
3. Регистры общего назначения — R0-R7
4. Счётчик команд — IP
5. Объём оперативной памяти — 256 байт
6. Тактовая частота (число операций в секунду) — 1, 2, 5, 10, 100 Гц
7. Флаги: ZF — флаг нуля, CF — флаг переноса
8. Система команд — 28 команд
9. Все арифметические операции производятся только над содержимым регистров!

Пример (арифметические операции)

$0xA6 - 0x23 + 0x45 * 0x02 =$

Мнемокод	Машинный код
R0 := 0xA6	0x09 0x00 0xA6
R1 := 0x23	0x09 0x01 0x23
R0 := R0 - R1	0x21 0x00 0x01
R2 := 0x45	0x09 0x02 0x45
R3 := 0x02	0x09 0x03 0x02
R2 := R2 * R3	0x26 0x02 0x03
R0 := R0 + R2	0x20 0x00 0x02
HLT	0xFF

Пример (адреса памяти)

Метка — именованный адрес памяти, используемый в мнемокоде

$a - b + c * d =$

```
 Мнемокод          Адрес    Машинный код
R0 := RAM[a]       0x00:   0x0A 0x00 0x16
R1 := RAM[b]       0x03:   0x0A 0x01 0x17
R0 := R0 - R1      0x06:   0x21 0x00 0x01
R2 := RAM[c]       0x09    0x0A 0x02 0x18
R3 := RAM[d]       0x0C:   0x0A 0x03 0x19
R2 := R2 * R3      0x0F:   0x26 0x02 0x03
R0 := R0 + R2      0x12:   0x20 0x00 0x02
HLT                0x15:   0xFF
a: 0xA6            0x16:   0xA6
b: 0x23            0x17:   0x23
c: 0x45            0x18:   0x45
d: 0x02            0x19:   0x02
```

Пример (безусловный переход)

Команда JMP (jump) может записывать адрес в IP и изменять порядок команд

```
JMP start          0x00:  0x30 0x06
a: 0xA6            0x02:  0xA6
b: 0x23            0x03:  0x23
c: 0x45            0x04:  0x45
d: 0x02            0x05:  0x02
start:
R0 := RAM[a]       0x06:  0x0A 0x00 0x02
R1 := RAM[b]       0x09:  0x0A 0x01 0x03
R0 := R0 - R1      0x0C:  0x21 0x00 0x01
R2 := RAM[c]       0x0F   0x0A 0x02 0x04
R3 := RAM[d]       0x12:  0x0A 0x03 0x05
R2 := R2 * R3      0x15:  0x26 0x02 0x03
R0 := R0 + R2      0x18:  0x20 0x00 0x02
HLT                0x1B:  0xFF
```

Флаги и условные переходы

Флаг нуля (ZF – zero flag) устанавливается в 1, если результат последней арифметической операции равен нулю. Иначе ZF устанавливается в 0.
Флаг переноса (CF - carry flag) устанавливается в 1, если в результате последней арифметической операции был перенос старшего разряда. Иначе CF устанавливается в 0.

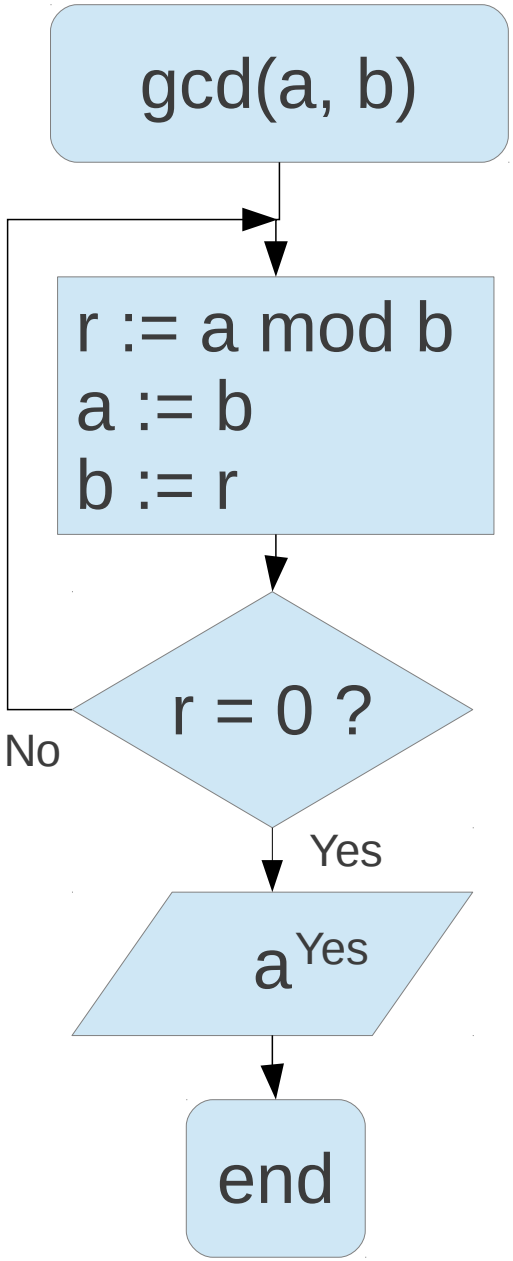
Команда JZ (jump zero) – прыгнуть, если нуль

JZ label
if ZF = 1 then IP := label

Команда JNZ (jump nonzero) – прыгнуть, если не нуль

JNZ label
if ZF = 0 then IP := label

Пример (алгоритм Евклида)




```
JMP start                0x00: 0x30 0x04
a: 0x0F                  0x02: 0x0F
b: 0x0A                  0x03: 0x0A
start:
  R0 := RAM[a]           0x04: 0x0A 0x00 0x02
  R1 := RAM[b]           0x07: 0x0A 0x01 0x03
loop:
  R2 := R0               0x0A: 0x0C 0x02 0x00
  R2 := R2 mod R1        0x0C: 0x28 0x02 0x01
  R0 := R1               0x10: 0x0C 0x00 0x01
  R1 := R2               0x13: 0x0C 0x01 0x02
  JNZ loop               0x16: 0x32 0x0A
HLT                      0x18: 0xFF
```

Флаги и команда сравнения CMP

Команда сравнения CMP RX, RY(compare) вычитает регистр RY из RX, но не сохраняет результат. Зато устанавливает флаги.

Рассмотрим выражение $a - b$ и три возможных случая

$a > b$: $a = 20, b = 10$
$a = 20 = 00010100b, -b = -10 = 11110110b$
$a + (-b) = (1)00001010b = 00001010b$
mod 256

$a < b$: $a = 10, b = 20$
$a = 10 = 00001010b, -b = -20 = 11101100b$
$a + (-b) = 00001010b + 11101100b =$
$11110110b$

$a = b$: $a = b = 10$
$a = 00001010b, -b = -10 = 11110110$
$a + (-b) = (1)00000000b = 00000000b$

a,b	ZF	CF
$a > b$	0	1
$a < b$	0	0
$a = b$	1	1

Команды условного перехода

Мнемоника	Машинный код	Описание
CMP RX, RY	0x22 0x0X 0x0Y	RX - RY
JZ XY	0x31 0xXY	if ZF then IP := 0xXY
JNZ XY	0x32 0xXY	if not ZF then IP := 0xXY
JE XY	0x31 0xXY	if ZF then IP := 0xXY
JNE XY	0x32 0xXY	if not ZF then IP := 0xXY
JL XY	0x33 0xXY	if not ZF and not CF then IP := 0xXY
JG XY	0x34 0xXY	if not ZF and CF then IP := 0xXY
JLE XY	0x35 0xXY	
JGE XY	0x36 0xXY	

Команды условного перехода

Мнемоника	Машинный код	Описание
CMP RX, RY	0x22 0x0X 0x0Y	RX - RY
JZ XY	0x31 0xXY	if ZF then IP := 0xXY
JNZ XY	0x32 0xXY	if not ZF then IP := 0xXY
JE XY	0x31 0xXY	if ZF then IP := 0xXY
JNE XY	0x32 0xXY	if not ZF then IP := 0xXY
JL XY	0x33 0xXY	if not ZF and not CF then IP := 0xXY
JG XY	0x34 0xXY	if not ZF and CF then IP := 0xXY
JLE XY	0x35 0xXY	if CF = ZF then IP := 0xXY
JGE XY	0x36 0xXY	

Команды условного перехода

Мнемоника	Машинный код	Описание
CMP RX, RY	0x22 0x0X 0x0Y	RX - RY
JZ XY	0x31 0xXY	if ZF then IP := 0xXY
JNZ XY	0x32 0xXY	if not ZF then IP := 0xXY
JE XY	0x31 0xXY	if ZF then IP := 0xXY
JNE XY	0x32 0xXY	if not ZF then IP := 0xXY
JL XY	0x33 0xXY	if not ZF and not CF then IP := 0xXY
JG XY	0x34 0xXY	if not ZF and CF then IP := 0xXY
JLE XY	0x35 0xXY	if CF = ZF then IP := 0xXY
JGE XY	0x36 0xXY	if CF then IP := 0xXY

Пример — max(a,b)

```
if a < b then
    a := b;
```

```
Мнемоника              Адрес Машинный код
JMP start          0x00: 0x30 0x04
a:0x34             0x02: 0x34
b:0xA4             0x03: 0xA4
start:
  R0 := RAM[a]     0x04: 0x0A 0x00 0x02
  R1 := RAM[b]     0x07: 0x0A 0x01 0x03
  CMP R0, R1       0x0A: 0x22 0x00 0x01 ; установить ZF,CF
  JGE end          0x0D: 0x36 0x12      ; посмотреть флаги
  MOV R0, R1       0x0F: 0x0C 0x00 0x01
end:
  HLT              0x12: 0xFF
```

Пример — abs(a - b)

```
if a > b then
    abs := a - b
else
    abs := b - a;
```

```
Мнемоника              Адрес Машинный код
JMP start              0x00: 0x30 0x04
a:0x34                 0x02: 0x34
b:0xA4                 0x03: 0xA4
start:
    R0 := RAM[a]       0x04: 0x0A 0x00 0x02
    R1 := RAM[b]       0x07: 0x0A 0x01 0x03
    CMP R0, R1         0x0A: 0x22 0x00 0x01 ; установить ZF,CF
    JLE else           0x0D: 0x35 0x14      ; посмотреть флаги
then:
    R0 := R0 - R1      0x0F: 0x21 0x0X 0x01
    JMP end            0x12: 0x30 0x1A      ; безусловный переход
else:
    R1 := R1 - R0      0x14: 0x21 0x01 0x00
    R0 := R1           0x17: 0x0C 0x00 0x01
end:
    HLT                0x1A: 0xFF
```

Пример — простое число

```
    Псевдокод
i := 2;
while i < n do begin
    if n mod i = 0 then
        return False
    i := i + 1
end;
return True
```

```
Мнемоника            Адрес Машинный код
JMP start            0x00: 0x30 0x03
n:0x33               0x02: 0x33
start:
  R0 := RAM[n]       0x03: 0x0A 0x00 0x02
  R1 := 2            0x06: 0x09 0x01 0x02
while:
  CMP R1, R0         0x09: 0x22 0x01 0x00
  JGE wend           0x0C: 0x36 0x1F
  if:
    R2 := R0         0x0D: 0x0C 0x02 0x00
    R2 := R2 mod R1  0x11: 0x28 0x02 0x01
    JNZ next         0x14: 0x32 0x1B
  then:
    R0 := 0          0x16: 0x09 0x00 0x00 ; False
    JMP end          0x19: 0x30 0x22
next:
  R1 := R1 + 1       0x1B: 0x24 0x01
  JMP while          0x1D: 0x30 0x09
wend:
  R0 := 1            0x1F: 0x09 0x00 0x01 ; True
end:
    HLT              0x22: 0xFF
```

Краткие итоги

- Мы знаем как работать с разными СС
- Понимаем, в чём особенность машинной арифметики
- Можем работать с дополнительными кодами
- Придумать свой процессор не очень сложно
- Что такое флаги, и как они работают
- Что такое команда jump
- Как с помощью прыжков имитировать управляющие конструкции (if, for, while)