

# 2

## Регистровый файл. Память. Программируемое устройство

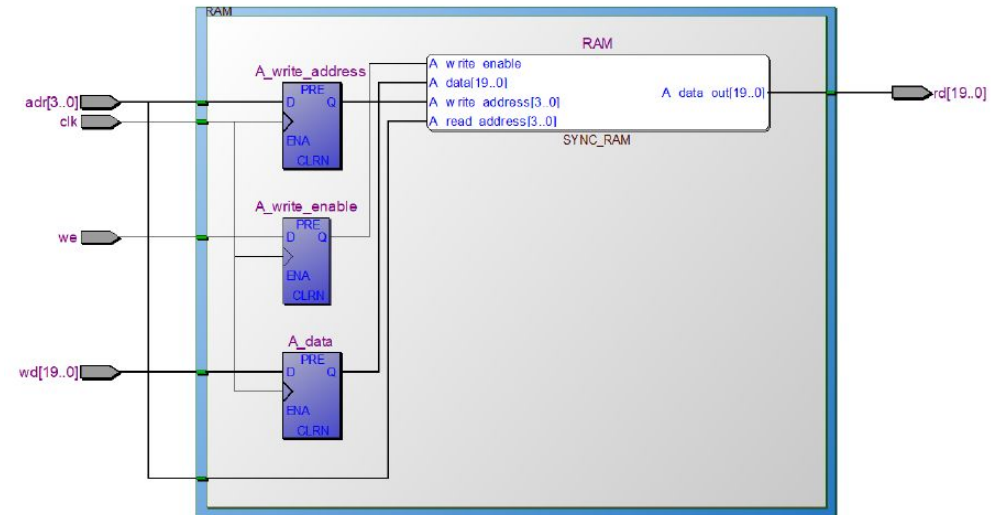
Архитектуры микропроцессорных систем и средств

# План лабораторной работы

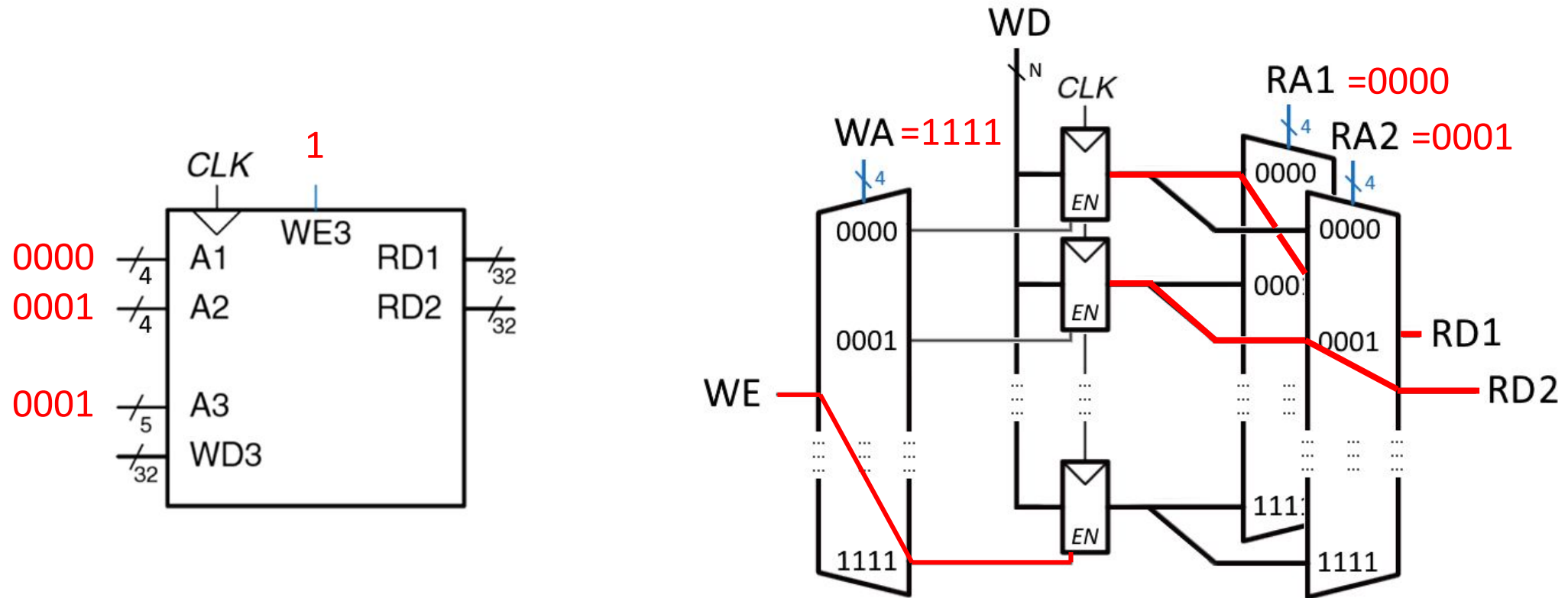
- 1 пара
  - Пример синтеза памяти на языке Verilog HDL (**T**)
  - Синхронная память (**T**)
  - Синтез и верификация трехпортового регистрового файла (**S**)
- 2 пара
  - Архитектура и микроархитектура программируемого устройства (**T**)
  - Пример программы (**T**)
  - Реализация микроархитектуры. Программирование (**S**)
  - Проверка на отладочном стенде (**S**)

# Пример синтеза памяти на языке Verilog HDL

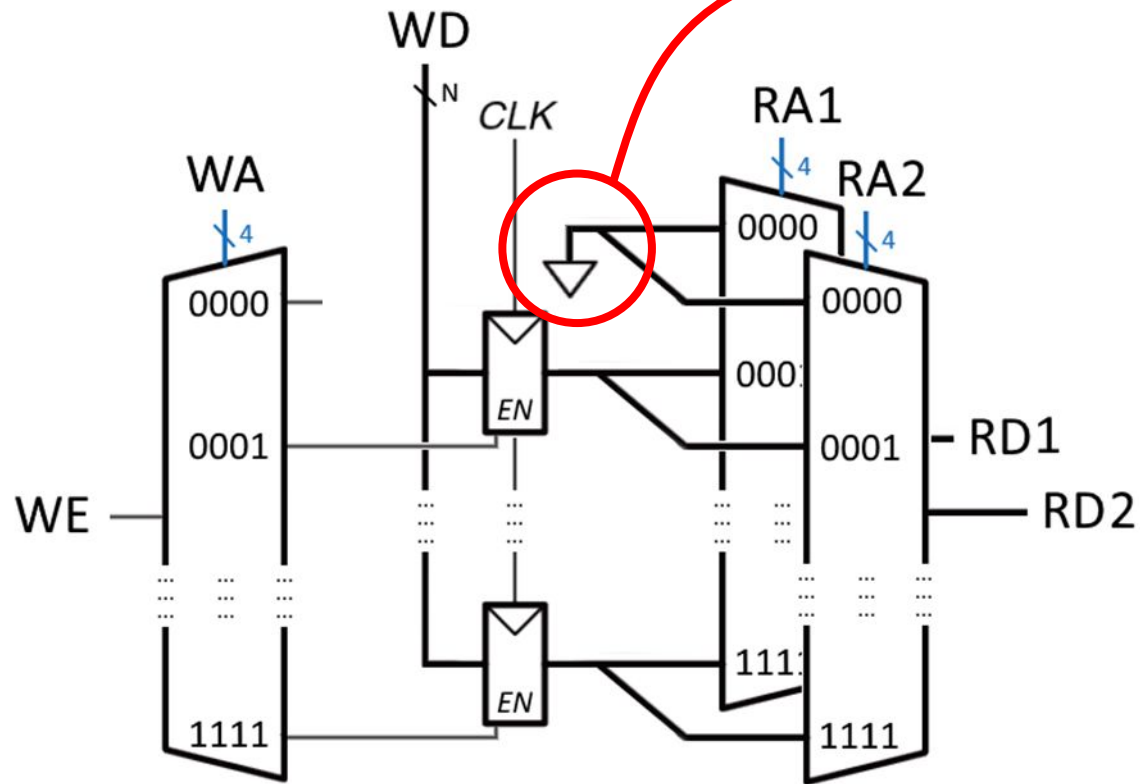
```
1 | module mem16_20 (  
2 |     input          clk,  
3 |     input   [3:0]  adr, // address  
4 |     input   [19:0] wd, // Write Data  
5 |     input          we, // Write Enable  
6 |     output  [19:0] rd  // Read Data  
7 | );  
8 | reg [19:0] RAM [0:15]; // создать память из 16-ти 20-битных ячеек  
9 |  
10 | assign rd = RAM[adr]; // подключение выхода rd к  
11 |                      // ячейке памяти с адресом adr  
12 | always @ (posedge clk) // запись данных wd  
13 |     if (we) RAM[adr] <= wd; // в ячейку по адресу adr,  
14 |                               // если we == 1  
15 | endmodule
```



# Трехпортовый регистровый файл



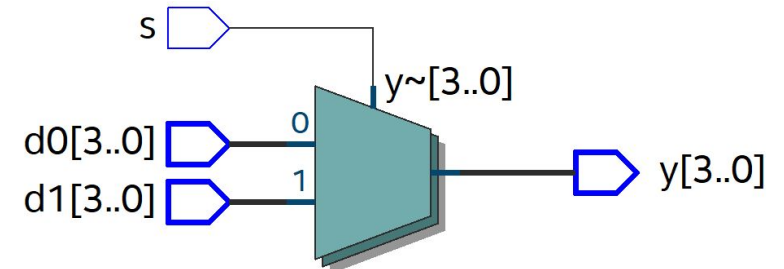
# Трехпортовый регистровый файл



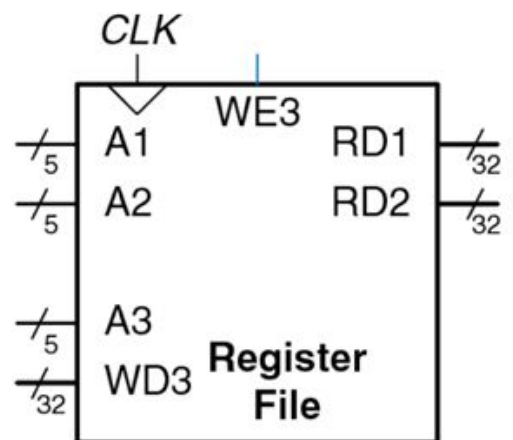
```
module ternary (  
    input    [3:0] d0, d1,  
    input    s,  
    output   [3:0] y  
);
```

```
    assign y = s ? d1 : d0;
```

```
endmodule
```



# Задание



# План лабораторной работы

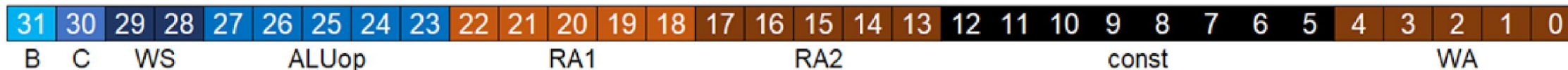
## ~~• 1 пара~~

- ~~• Пример синтеза памяти на языке Verilog HDL (T)~~
- ~~• Синхронная память (T)~~
- ~~• Синтез и верификация трехпортового регистрового файла (S)~~

## • 2 пара

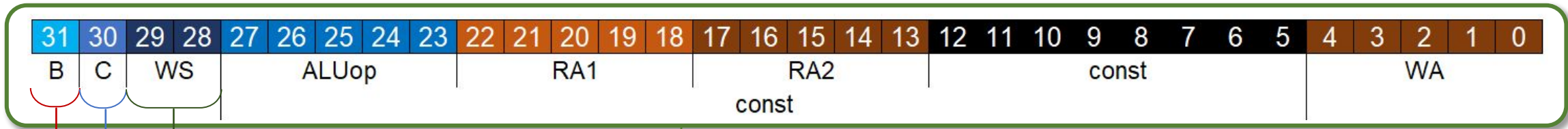
- Архитектура и микроархитектура программируемого устройства (T)
- Пример программы (T)
- Реализация микроархитектуры. Программирование (S)
- Проверка на отладочном стенде (S)

# Архитектура



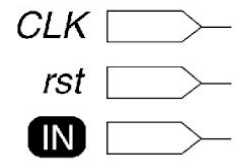
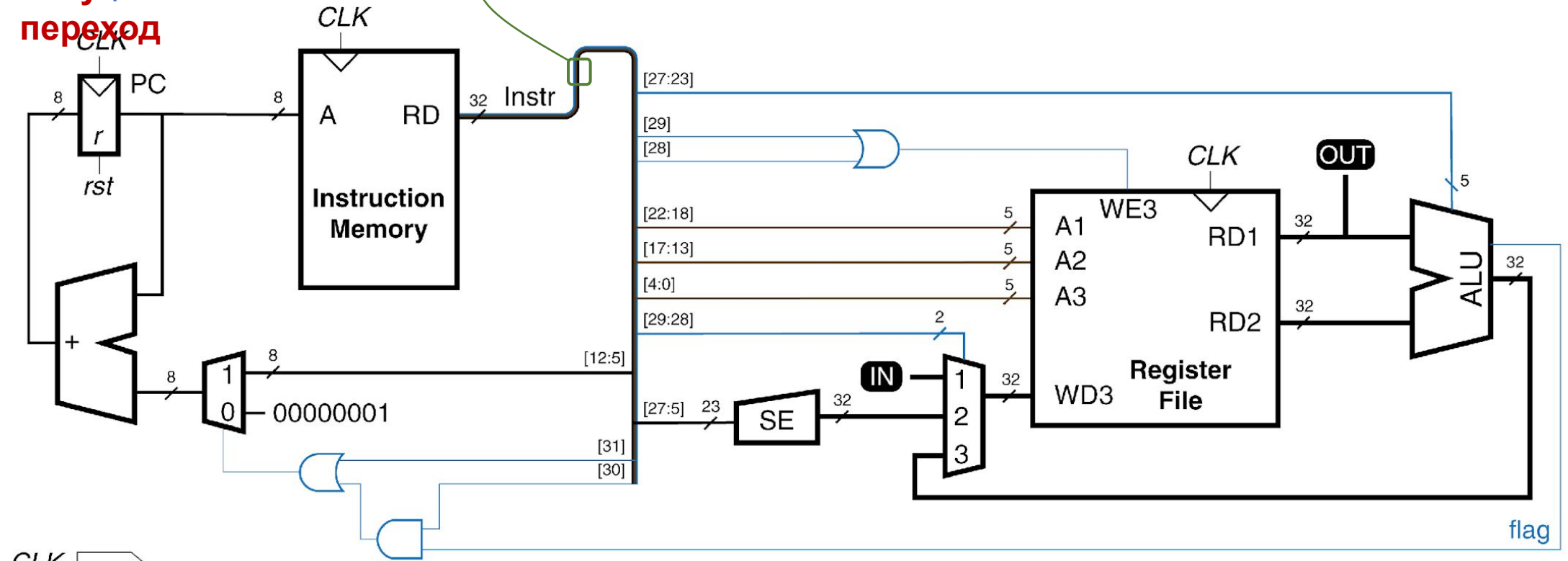
Сигнал	Описание сигнала
B	выполнить безусловный переход
C	выполнить условный переход
WS[1:0]	источник для записи в регистровый файл со следующим выбором источников: 0 — нет записи; 1 — данные с переключателей; 2 — константа из инструкции; 3 — запись результата АЛУ;
ALUop[4:0]	код операции, которую надо выполнить АЛУ
RA1[4:0]	адрес первого операнда АЛУ
RA2[4:0]	адрес второго операнда АЛУ
const[7:0]	8-битное значение константы
WA[4:0]	адрес регистра в регистровом файле, куда будет производиться запись





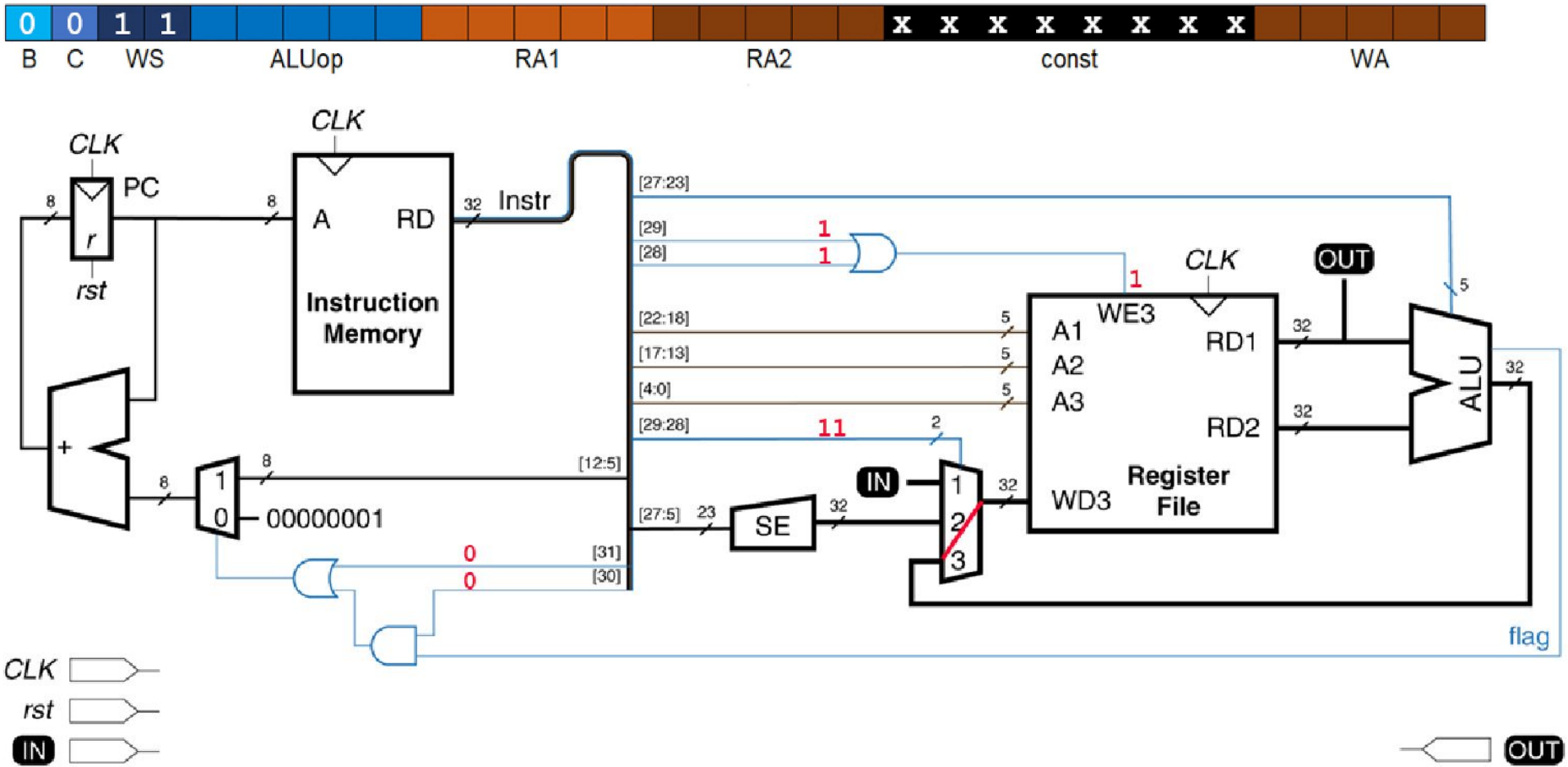
Откуда пишем в регистровый файл, и пишем ли вообще

Условный переход  
 Безусловный переход



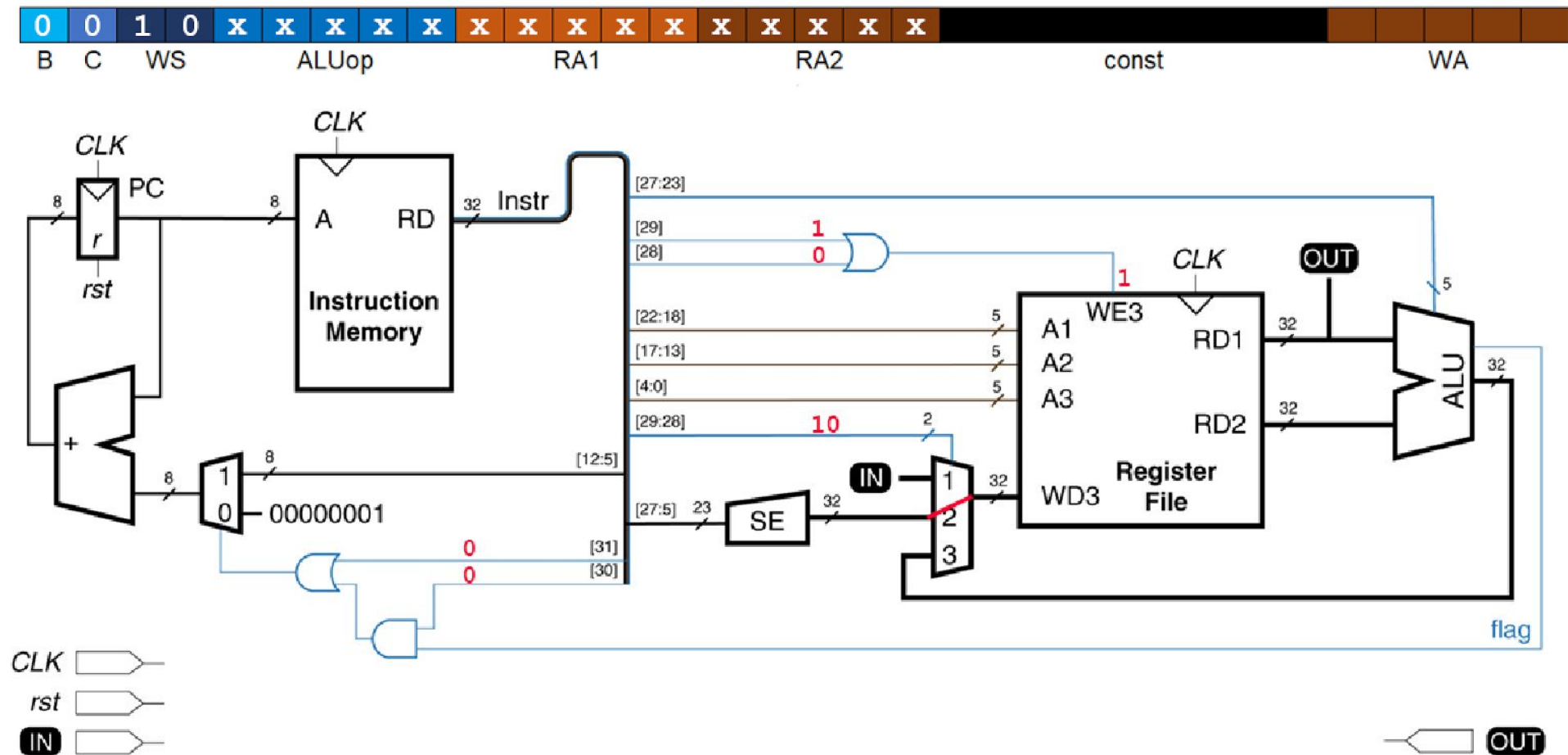
# 1. Операция на АЛУ

$$\text{reg}[WA] \leftarrow \text{reg}[RA1] \text{ (ALUop) } \text{reg}[RA2]$$



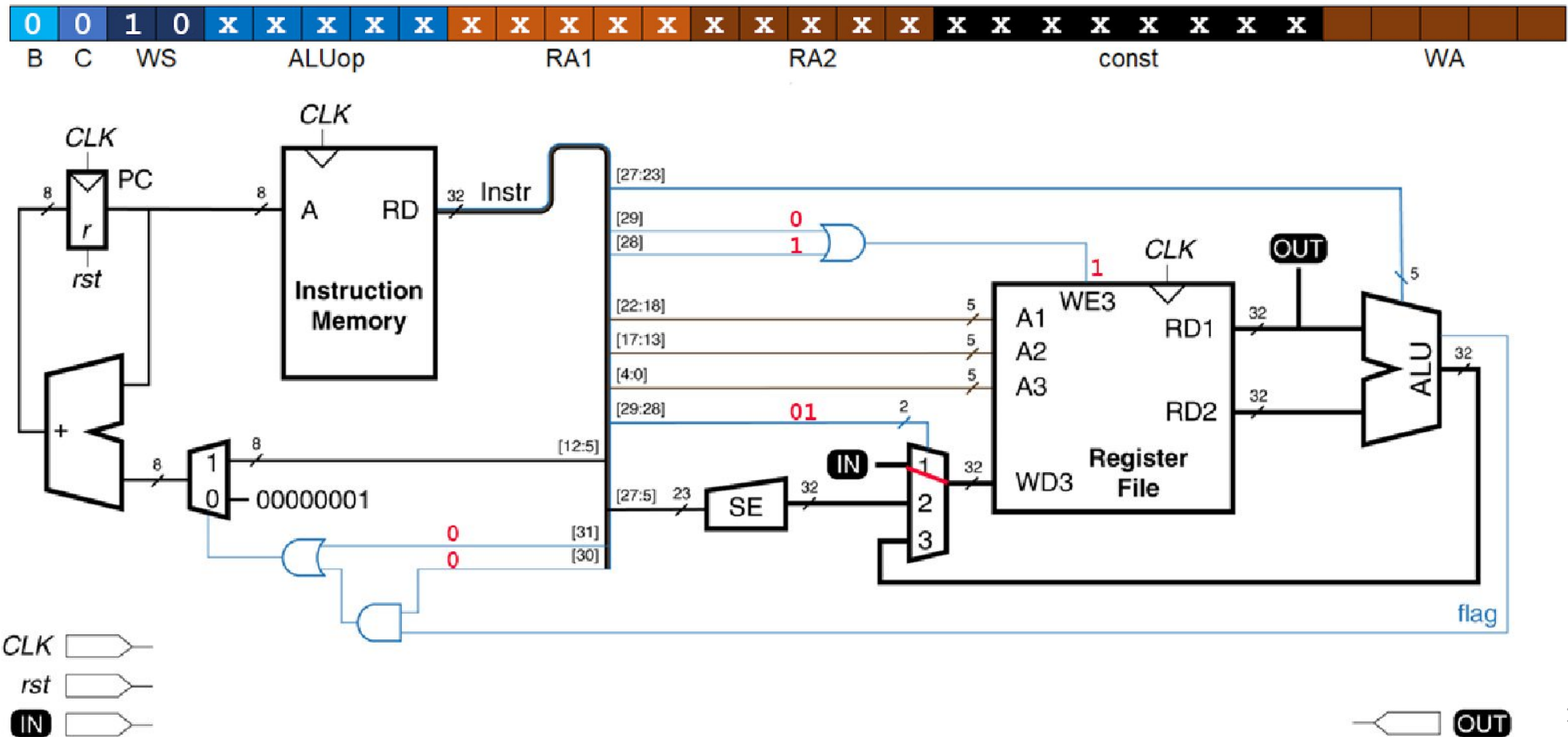
# 2. Загрузка константы

reg[WA] ← const



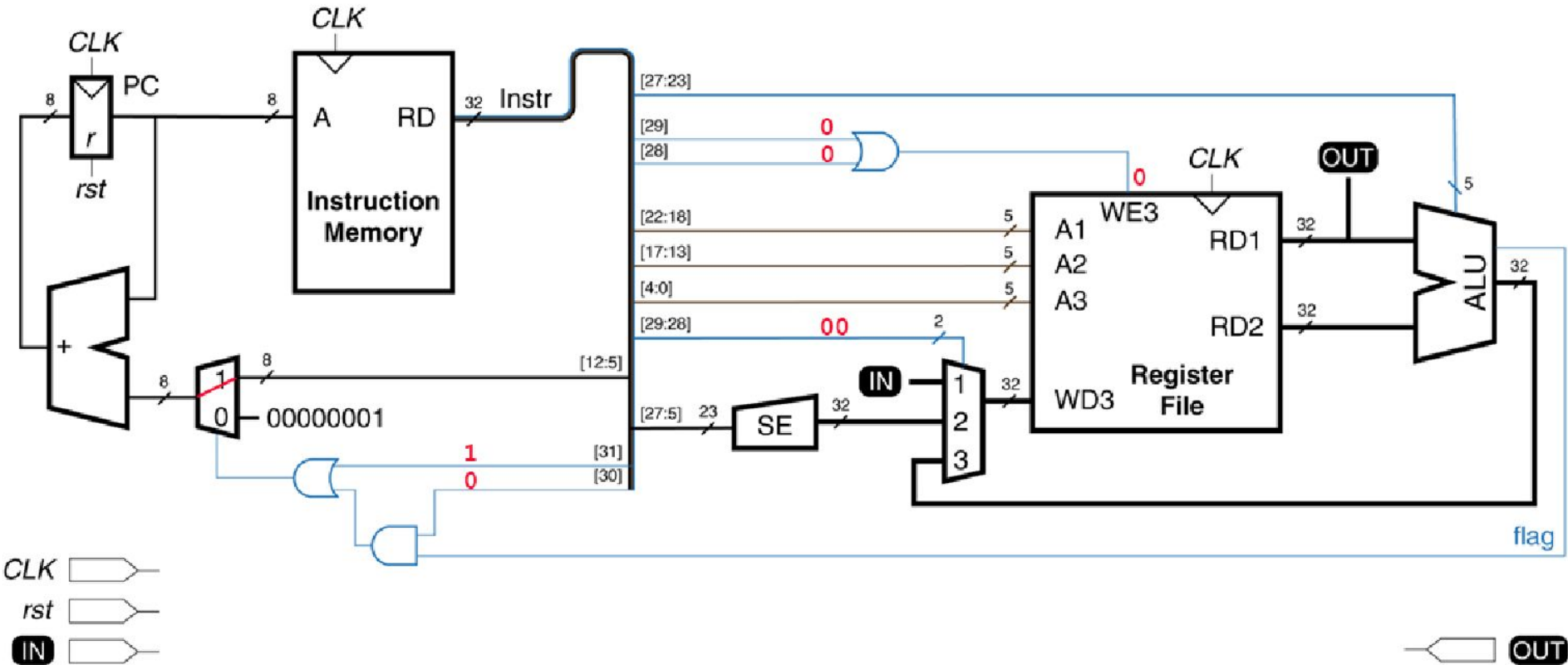
# 3. Загрузка с внешних устройств

reg[WA] ← switches



# 4. Безусловный переход

$PC \leftarrow PC + const$



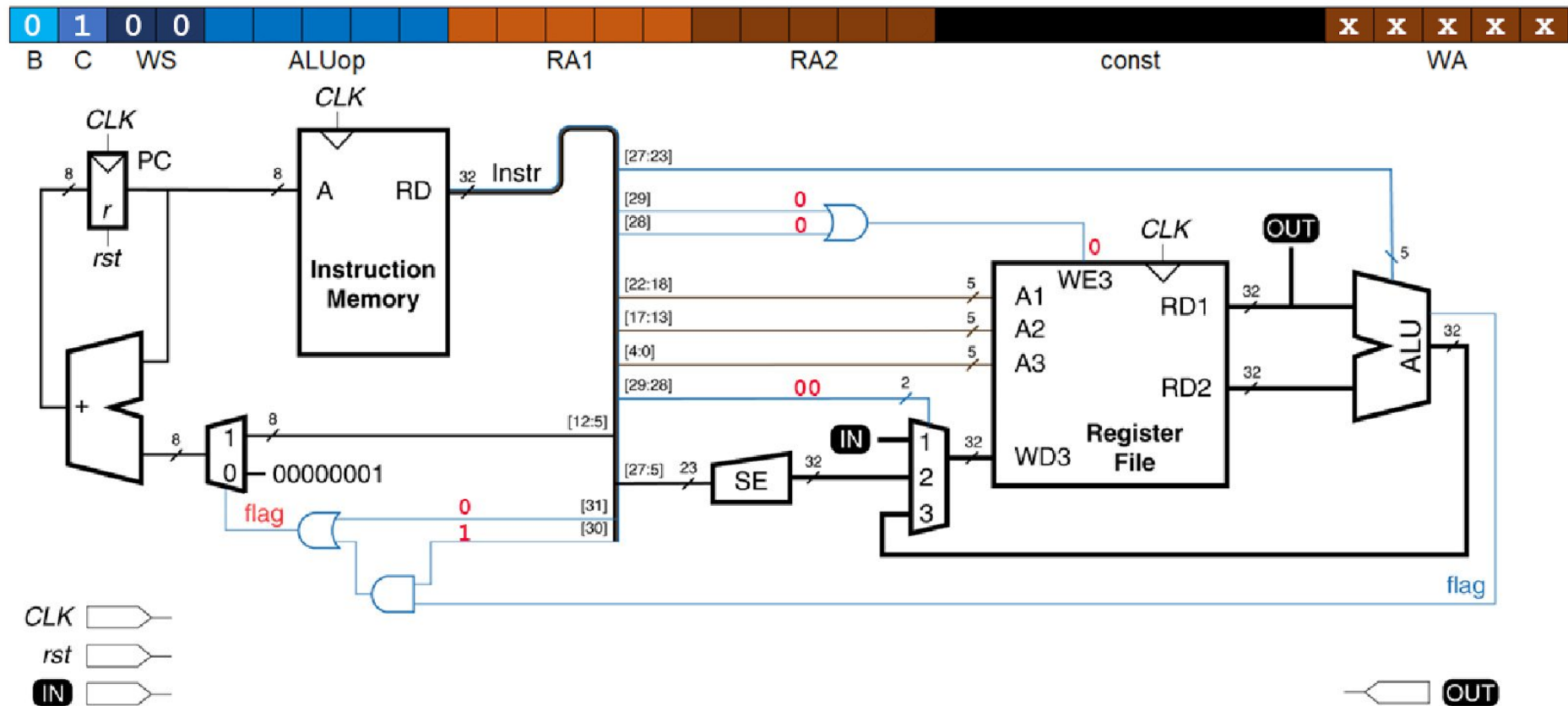
# 5. Условный переход

```
if (reg[RA1] (ALUop) reg[RA2]) then
```

```
    PC ← PC + const
```

```
else
```

```
    PC ← PC + 1
```



# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

`reg[3] ← 1`

`if (reg[2] == reg[0]) PC ← PC + (4)`

`reg[4] ← reg[4] + reg[1]`

`reg[2] ← reg[2] - reg[3]`

`PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
0	0	0	0

# Пример программы (13 \* switches)

```
➔ reg[1] ← 13
  reg[2] ← switches
  reg[3] ← 1
  if (reg[2] == reg[0]) PC ← PC + (4)
  reg[4] ← reg[4] + reg[1]
  reg[2] ← reg[2] - reg[3]
  PC ← PC + (-3)
  PC ← PC + (0)
```

reg[1]	reg[2]	reg[3]	reg[4]
13	0	0	0



# Пример программы (13 \* switches)

`reg[1] ← 13`

➔ `reg[2] ← switches`

`reg[3] ← 1`

`if (reg[2] == reg[0]) PC ← PC + (4)`

`reg[4] ← reg[4] + reg[1]`

`reg[2] ← reg[2] - reg[3]`

`PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	2	0	0

# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

➔ `reg[3] ← 1`

`if (reg[2] == reg[0]) PC ← PC + (4)`

`reg[4] ← reg[4] + reg[1]`

`reg[2] ← reg[2] - reg[3]`

`PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	2	1	0

# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

`reg[3] ← 1`

➔ `if (reg[2] == reg[0]) PC ← PC + (4)`

`reg[4] ← reg[4] + reg[1]`

`reg[2] ← reg[2] - reg[3]`

`PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	2	1	0

# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

`reg[3] ← 1`

`if (reg[2] == reg[0]) PC ← PC + (4)`

➔ `reg[4] ← reg[4] + reg[1]`

`reg[2] ← reg[2] - reg[3]`

`PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	2	1	13

# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

`reg[3] ← 1`

`if (reg[2] == reg[0]) PC ← PC + (4)`

`reg[4] ← reg[4] + reg[1]`

➔ `reg[2] ← reg[2] - reg[3]`

`PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	1	1	13

# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

`reg[3] ← 1`

`if (reg[2] == reg[0]) PC ← PC + (4)`

`reg[4] ← reg[4] + reg[1]`

`reg[2] ← reg[2] - reg[3]`

➔ `PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	1	1	13

# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

`reg[3] ← 1`

➔ `if (reg[2] == reg[0]) PC ← PC + (4)`

`reg[4] ← reg[4] + reg[1]`

`reg[2] ← reg[2] - reg[3]`

`PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	1	1	13

# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

`reg[3] ← 1`

`if (reg[2] == reg[0]) PC ← PC + (4)`

➔ `reg[4] ← reg[4] + reg[1]`

`reg[2] ← reg[2] - reg[3]`

`PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	1	1	26



# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

`reg[3] ← 1`

`if (reg[2] == reg[0]) PC ← PC + (4)`

`reg[4] ← reg[4] + reg[1]`

➔ `reg[2] ← reg[2] - reg[3]`

`PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	0	1	26

# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

`reg[3] ← 1`

`if (reg[2] == reg[0]) PC ← PC + (4)`

`reg[4] ← reg[4] + reg[1]`

`reg[2] ← reg[2] - reg[3]`

➔ `PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	0	1	26

# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

`reg[3] ← 1`

➔ `if (reg[2] == reg[0]) PC ← PC + (4)`

`reg[4] ← reg[4] + reg[1]`

`reg[2] ← reg[2] - reg[3]`

`PC ← PC + (-3)`

`PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	0	1	26

# Пример программы (13 \* switches)

`reg[1] ← 13`

`reg[2] ← switches`

`reg[3] ← 1`

`if (reg[2] == reg[0]) PC ← PC + (4)`

`reg[4] ← reg[4] + reg[1]`

`reg[2] ← reg[2] - reg[3]`

`PC ← PC + (-3)`

➔ `PC ← PC + (0)`

reg[1]	reg[2]	reg[3]	reg[4]
13	0	1	26

# Пример программы

	B	C	WS	ALUop				RA1				RA2				const				WA											
0x00	0	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	1	1	0	1	0	0	0	0	1	
0x04	0	0	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	1	0	
0x08	0	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	1	0	0	0	1	1	
0x0C	0	1	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	x	x	x	x	x	
0x10	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	x	x	x	x	x	x	x	0	0	1	0	0	
0x14	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	1	x	x	x	x	x	x	x	0	0	0	1	0
0x18	1	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	1	1	1	1	1	1	0	1	x	x	x	x	x	
0x1C	1	0	0	0	x	x	x	x	x	0	0	1	0	0	x	x	x	x	x	0	0	0	0	0	0	0	x	x	x	x	x

# План лабораторной работы

## ~~• 1 пара~~

- ~~• Пример синтеза памяти на языке Verilog HDL (T)~~
- ~~• Синхронная память (T)~~
- ~~• Синтез и верификация трехпортового регистрового файла (S)~~

## ~~• 2 пара~~

- ~~• Архитектура и микроархитектура программируемого устройства (T)~~
- ~~• Пример программы (T)~~
- Реализация микроархитектуры. Программирование (S)
- Проверка на отладочном стенде (S)