Архитектуры процессорных систем

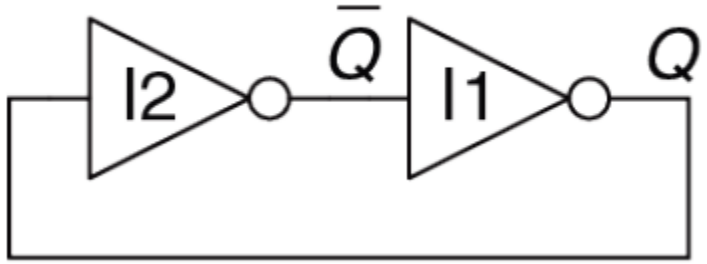
Лекция 4. Последовательностная логика. Память

Цикл из 16 лекций о цифровой схемотехнике, способах построения и архитектуре компьютеров

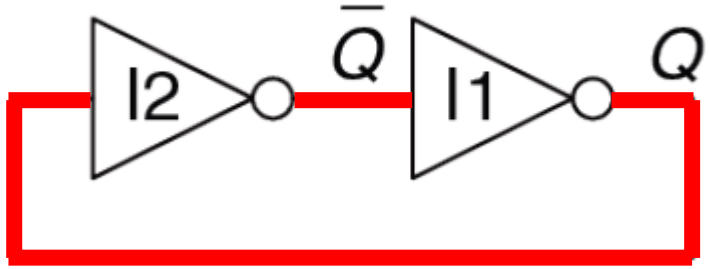
План лекции

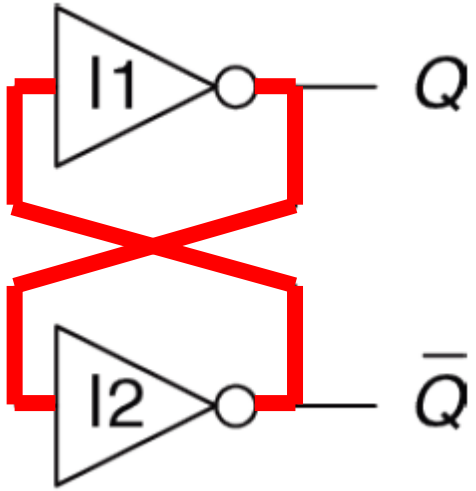
- Защелка, триггер, регистр
- Регистровый файл
- Автомат состояний (конечный автомат)
- Временные характеристики цифровых устройств
- Примитивное программируемое устройство

Бистабильное устройство

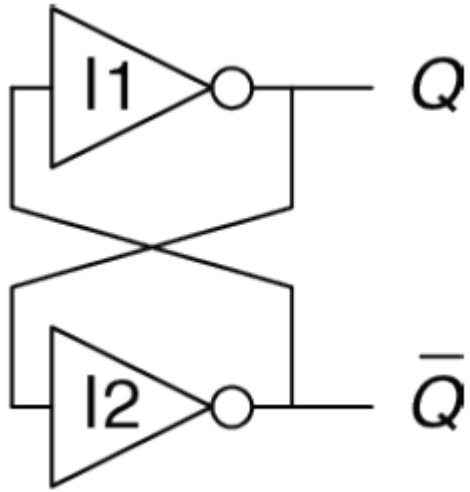


Бистабильное устройство

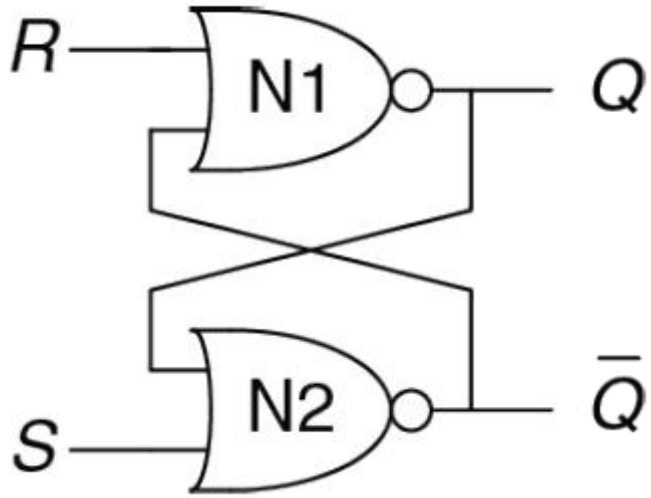




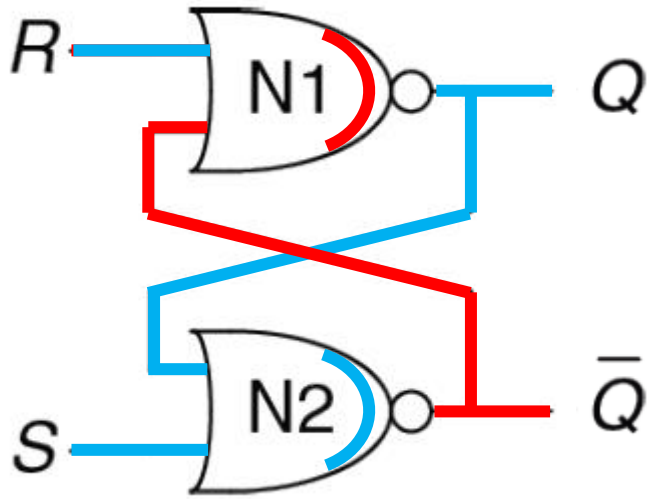
Бистабильное устройство



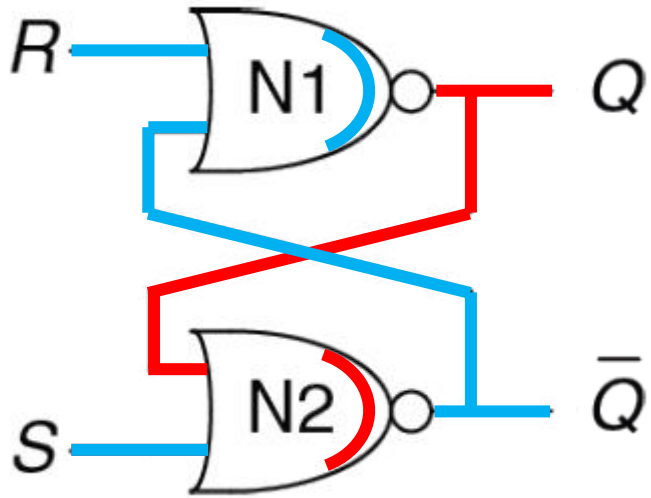
RS-триггер



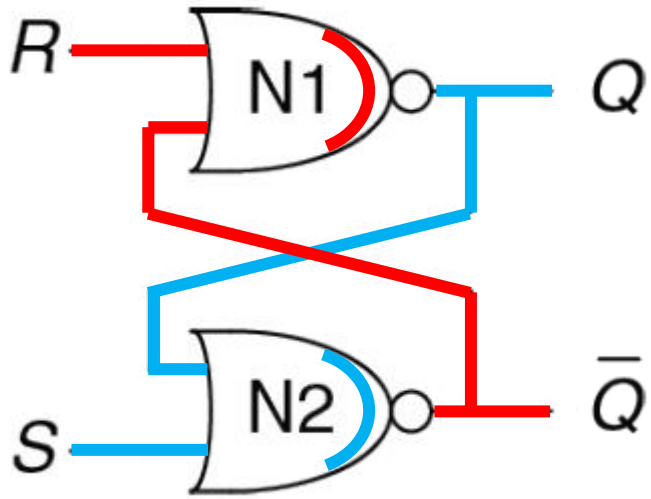
RS-триггер



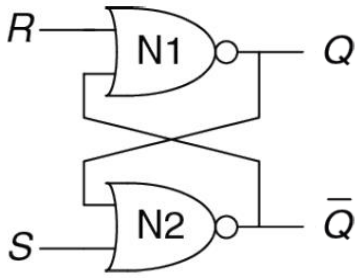
RS-триггер

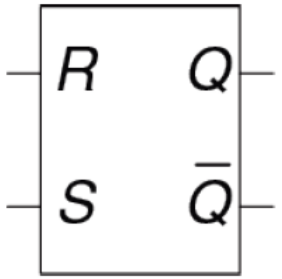


RS-триггер



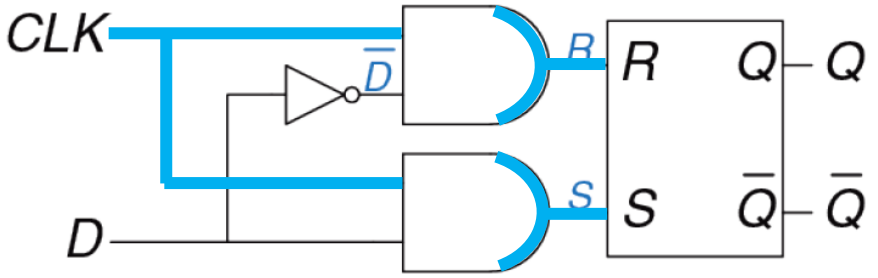
RS-триггер





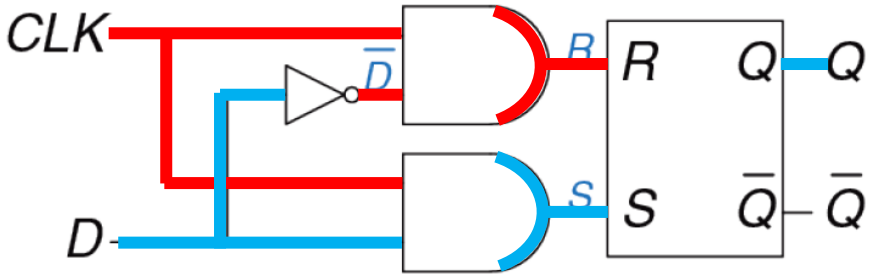
S	R	Q	$\bar{Q}$
0	0	Q_{prev}	$\bar{Q}_{prev}$
0	1	0	1
1	0	1	0
1	1	0	0

D-защелка (latch)



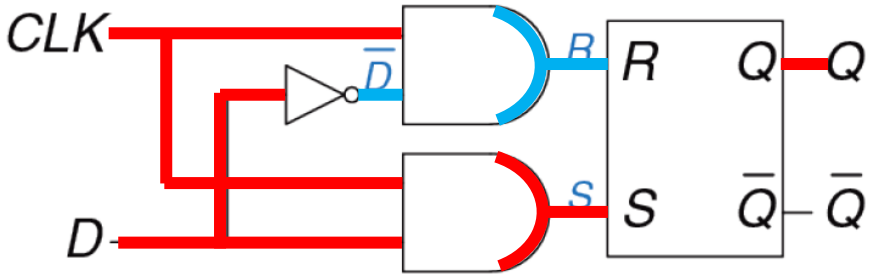
CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

D-защелка (latch)



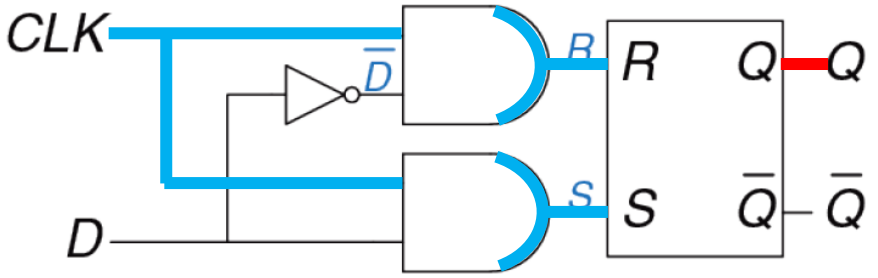
CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

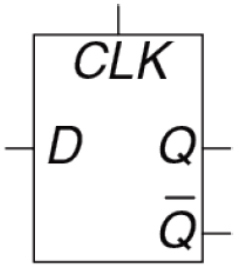
D-защелка (latch)



CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

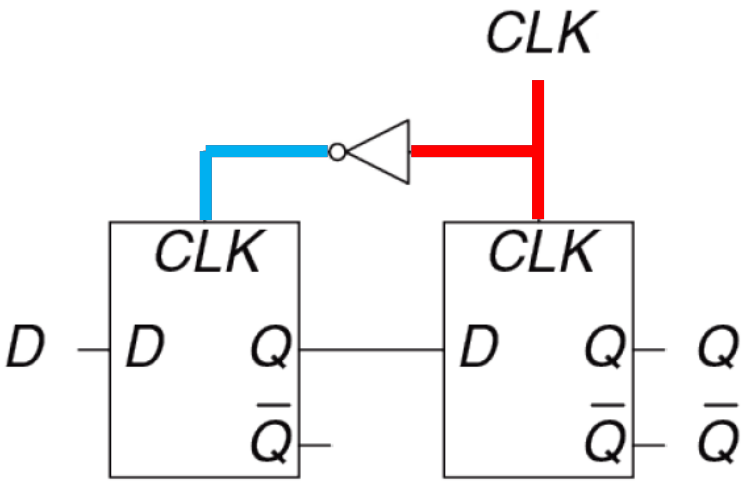
D-защелка (latch)

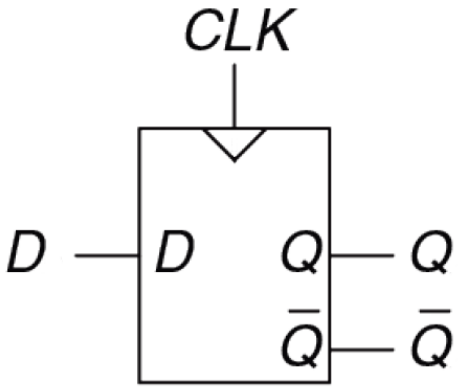




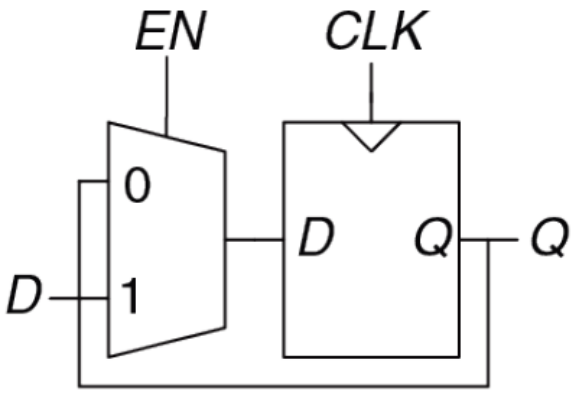
CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

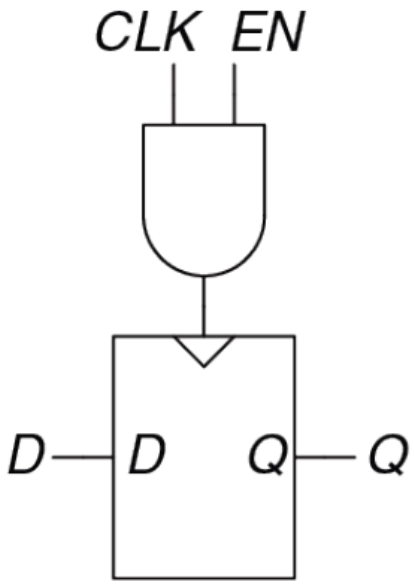
D-триггер

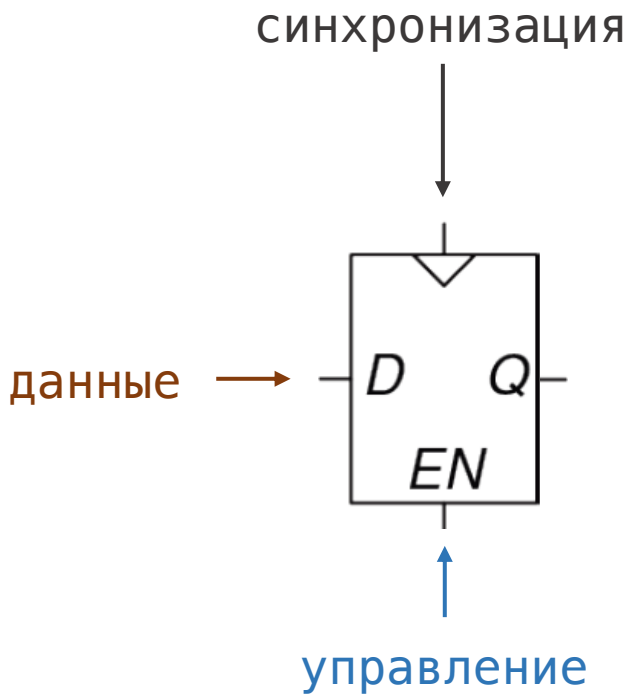




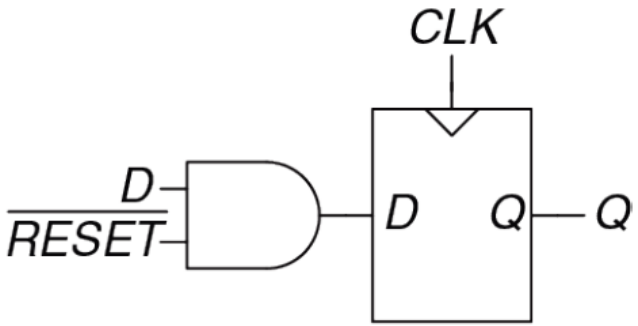
D-триггер с сигналом разрешения

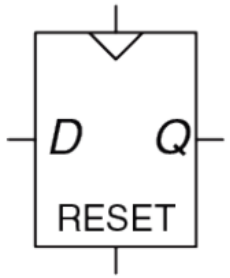


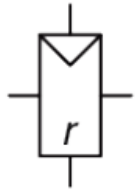




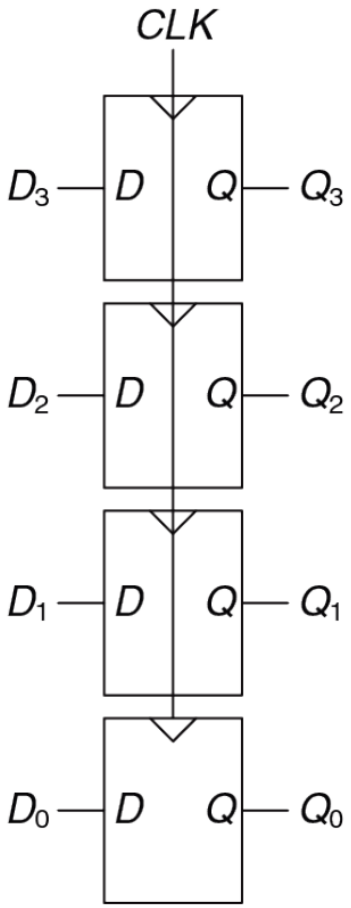
D-триггер с сигналом сброса

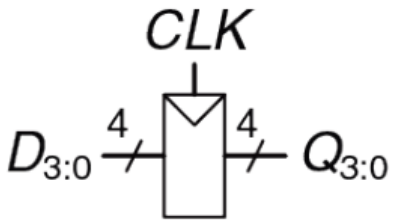




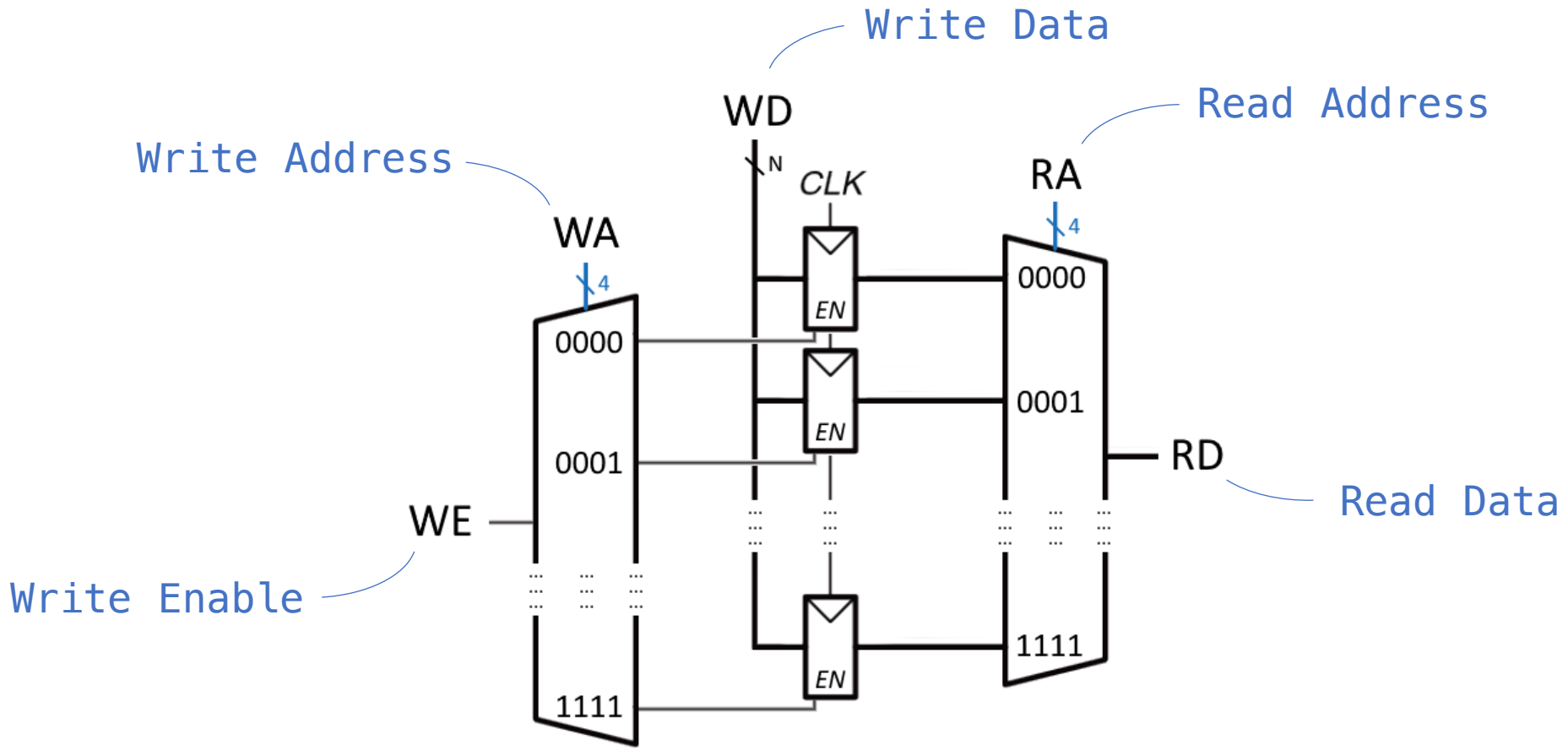


Регистр

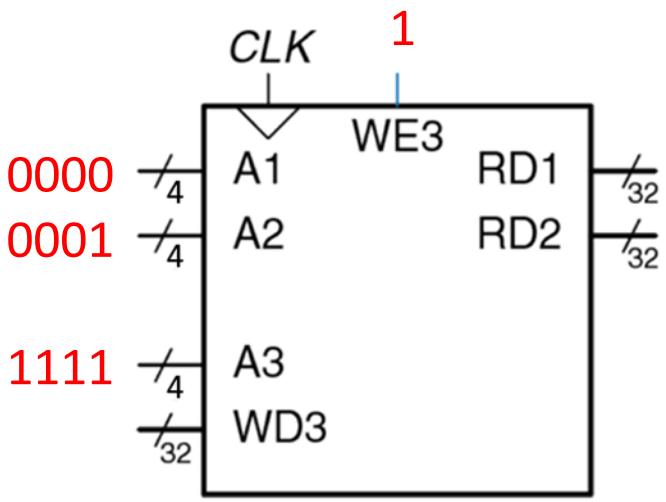


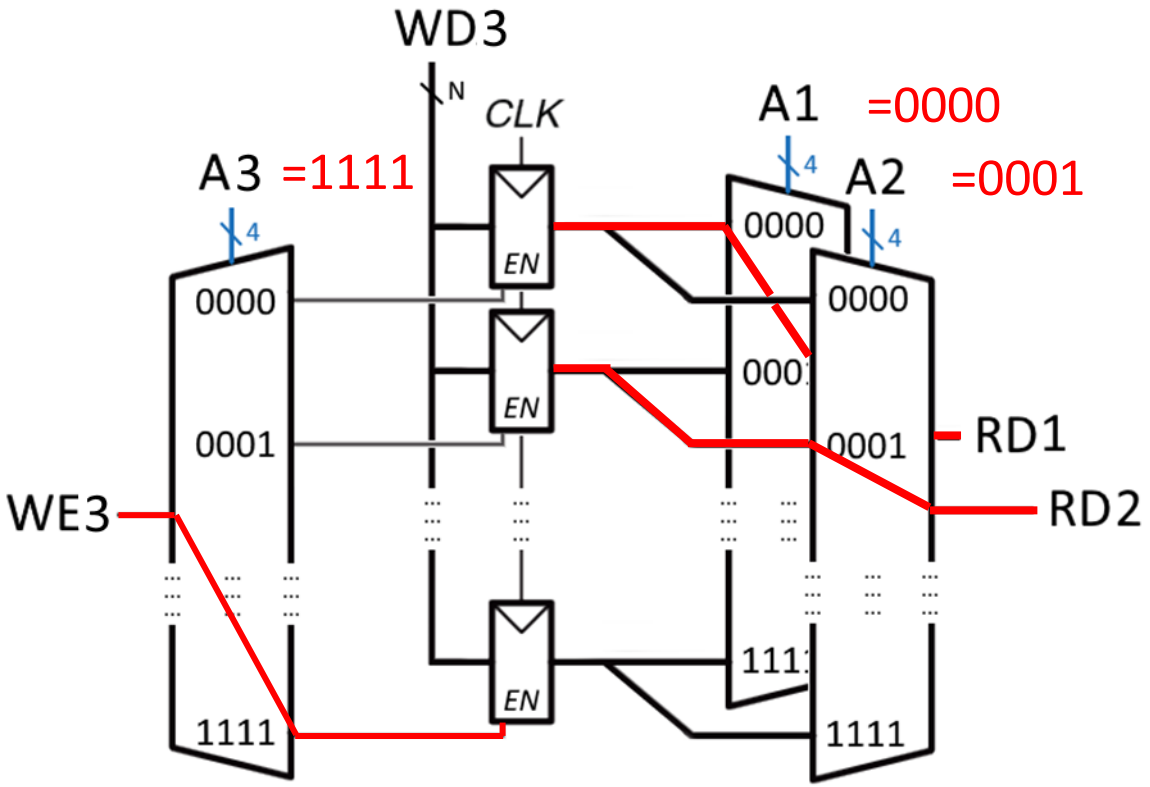


Адресуемые регистры

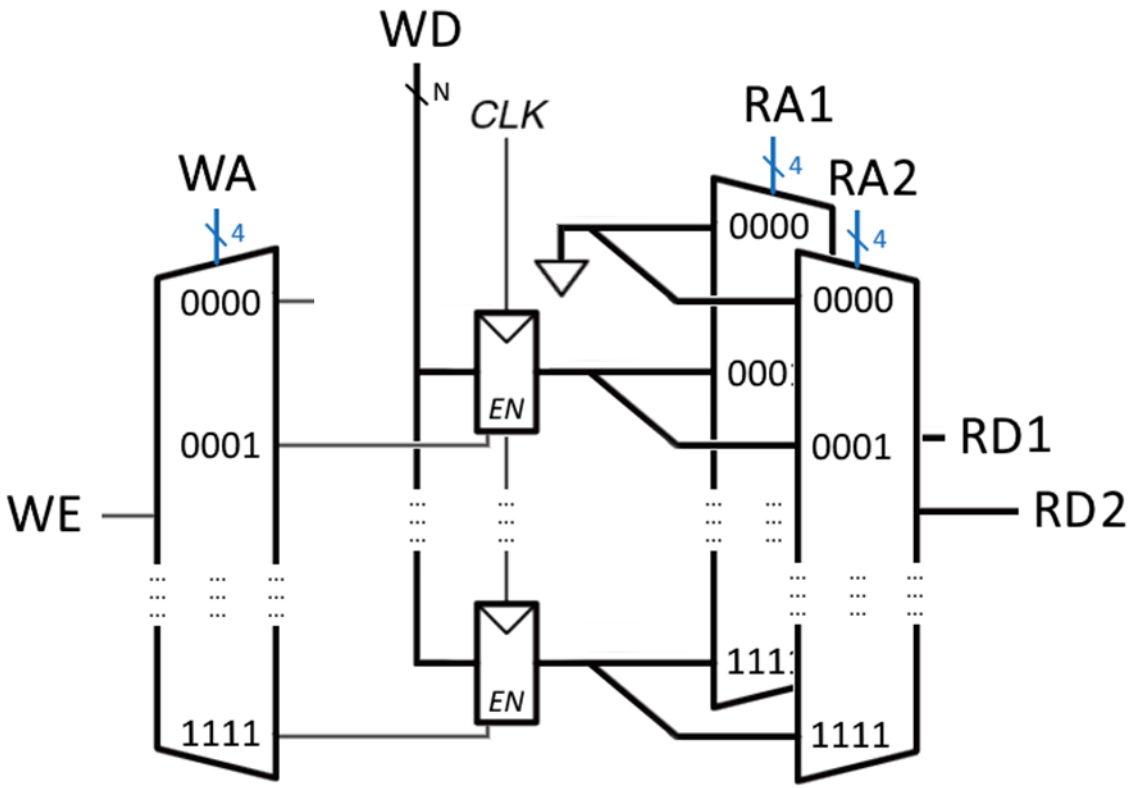


Трехпортовый регистровый файл

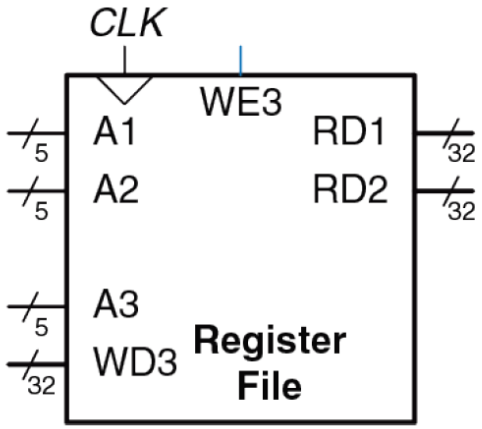


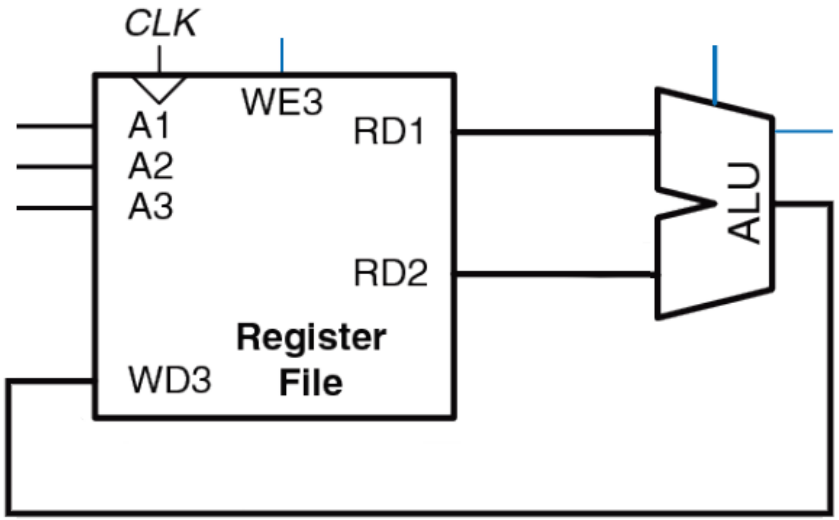


Трехпортовый регистровый файл

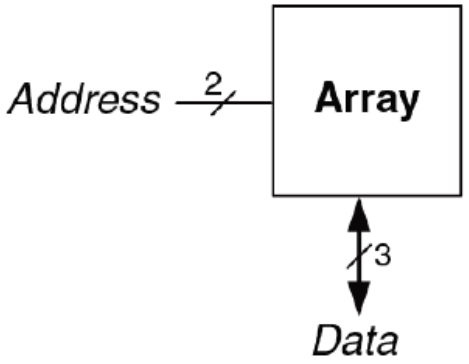


Регистровый файл для RISC-V

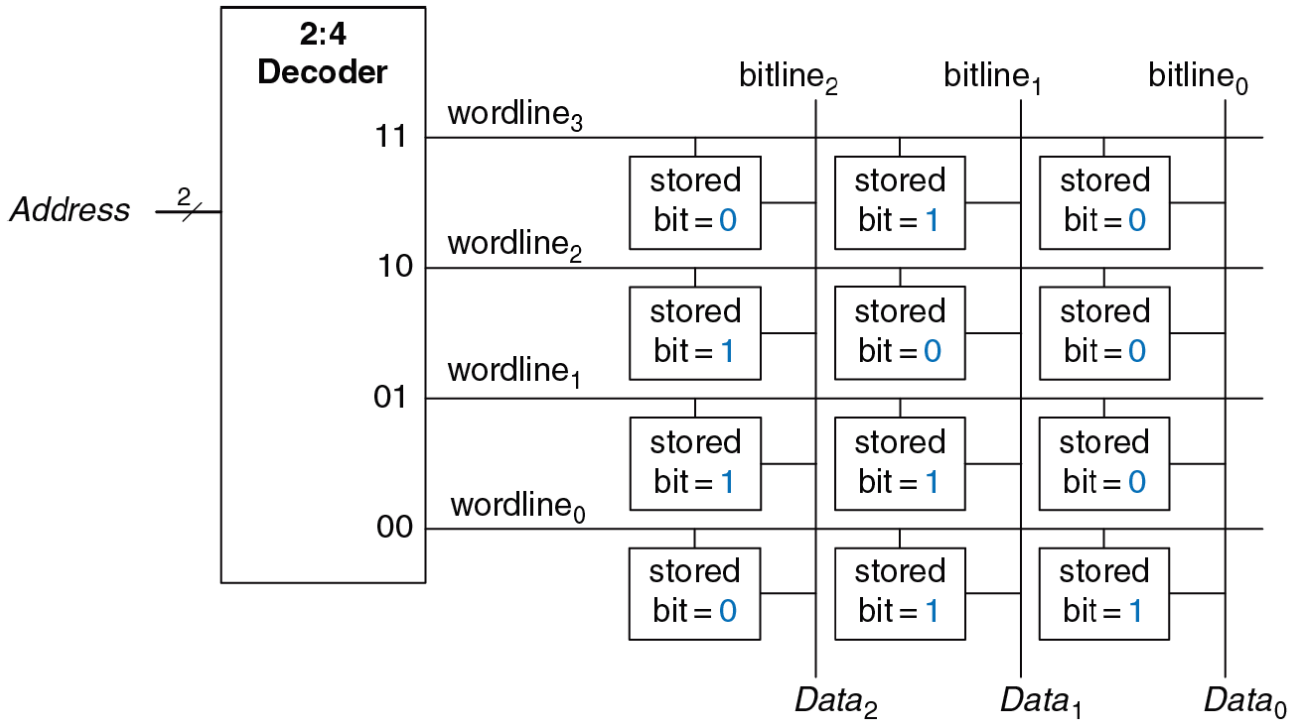




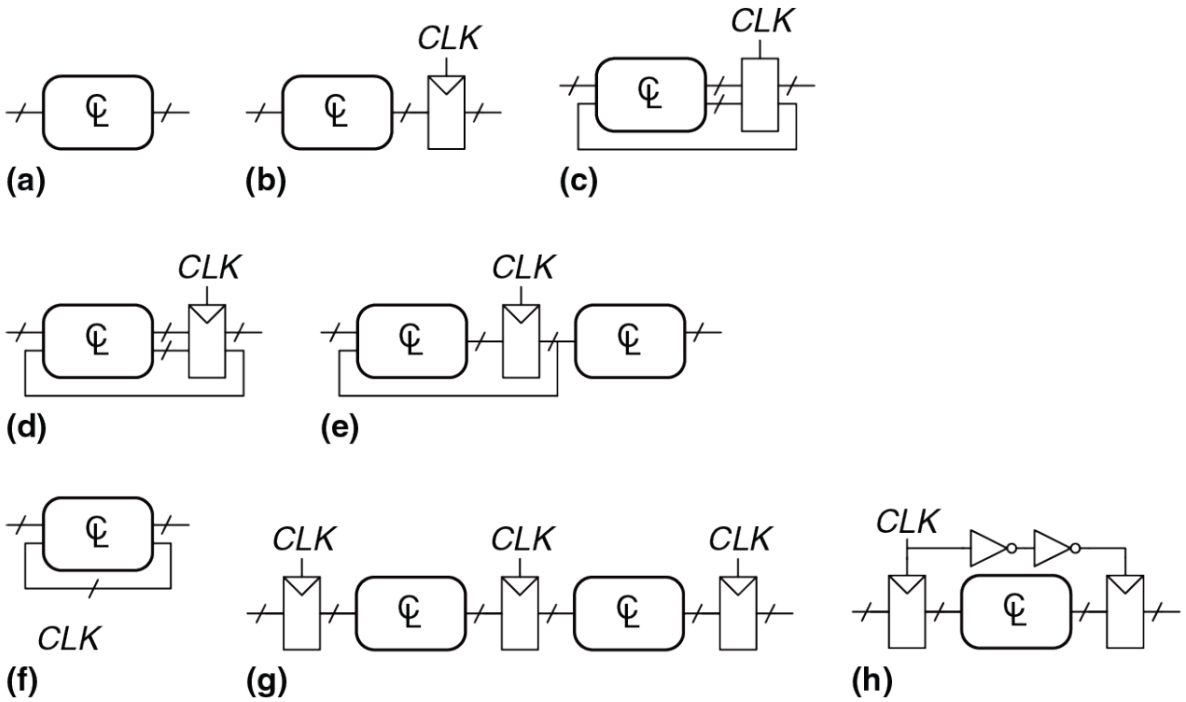
Память


Address
2
Array
3
Data

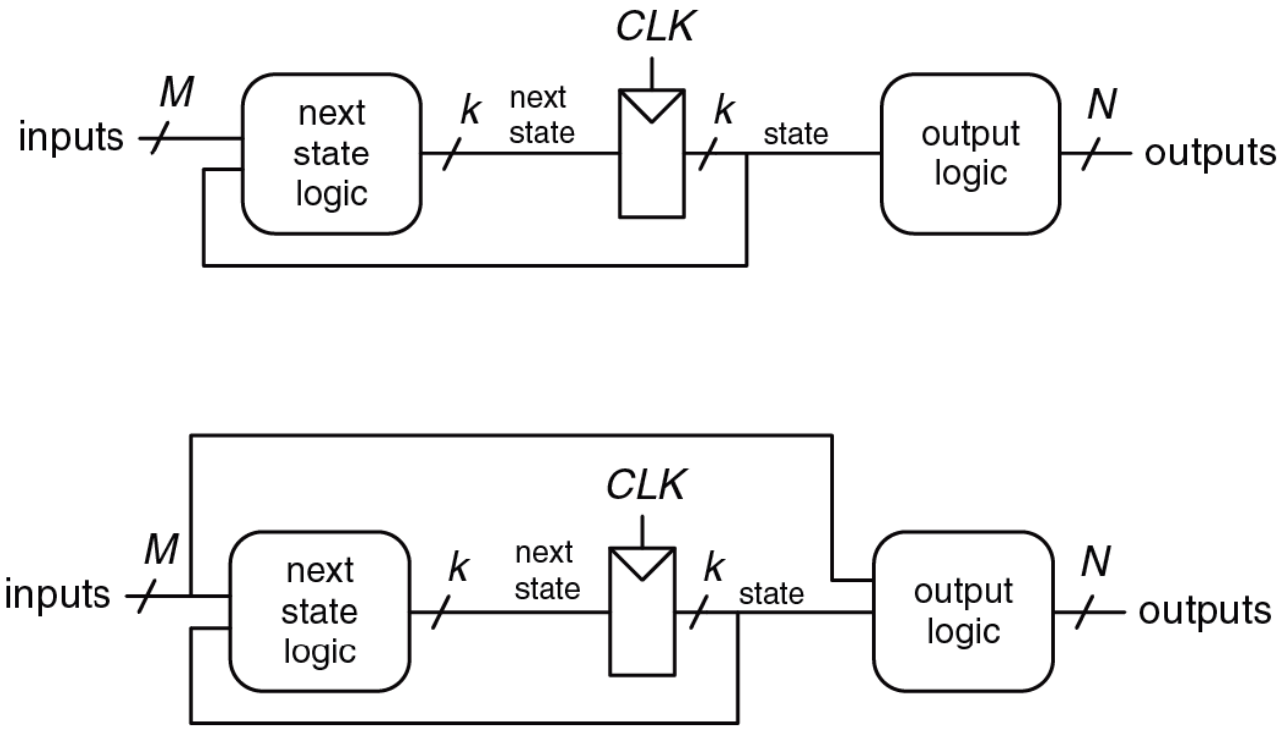

Address
Data
11
10
01
00
0 1 0
1 0 0
1 1 0
0 1 1
depth
width


2:4
Decoder
Address
2
11
10
01
00
wordline3
wordline2
wordline1
wordline0
bitline2
bitline1
bitline0
stored bit = 0
stored bit = 1
stored bit = 0
stored bit = 0
stored bit = 0
stored bit = 0
stored bit = 1
stored bit = 1
stored bit = 0
stored bit = 0
stored bit = 1
stored bit = 1
Data2
Data1
Data0

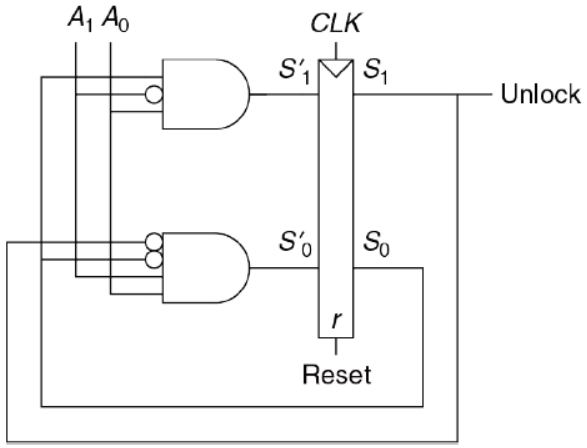
Последовательностные схемы



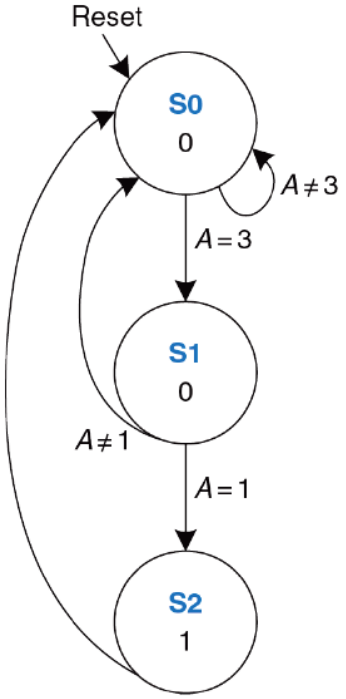
Конечные автоматы



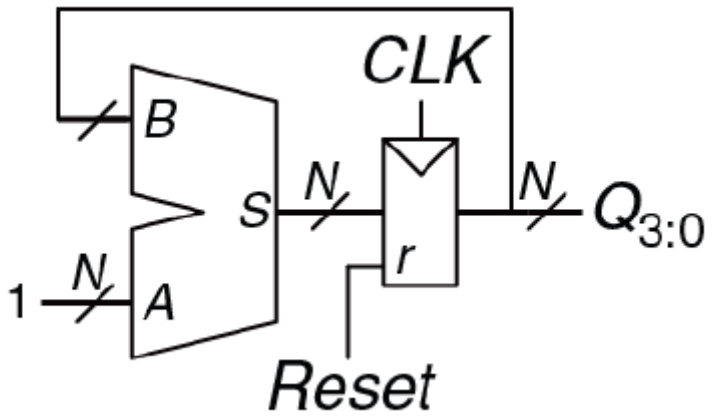
Конечные автоматы

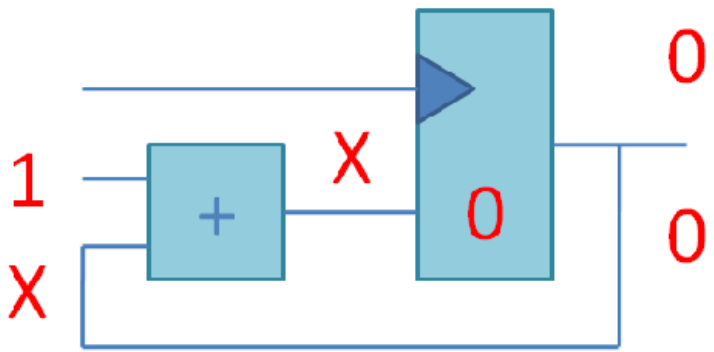


Current State *S*	Input *A*	Next State *S'*
S0	0	S0
S0	1	S0
S0	2	S0
S0	3	S1
S1	0	S0
S1	1	S2
S1	2	S0
S1	3	S0
S2	X	S0

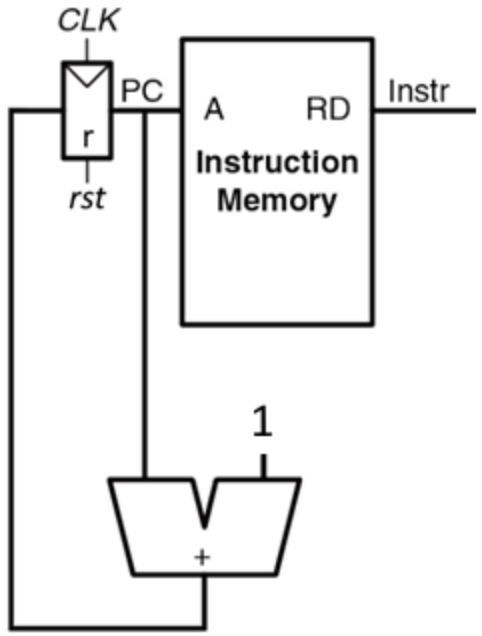


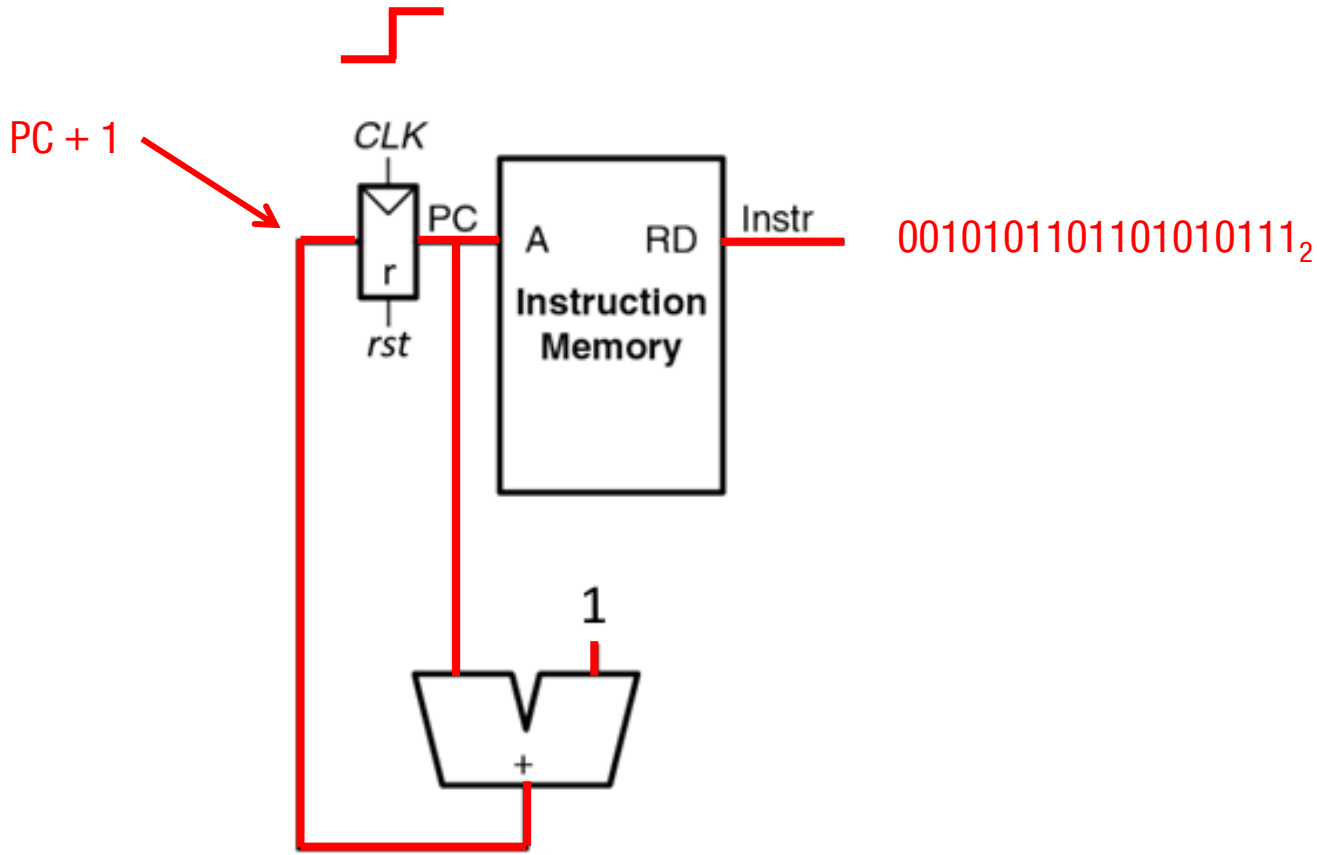
Счетчик



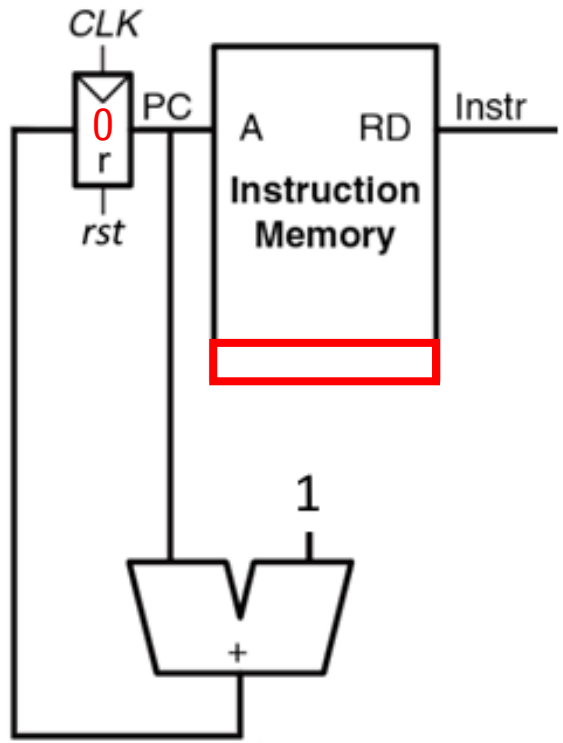


Последовательное считывание из памяти

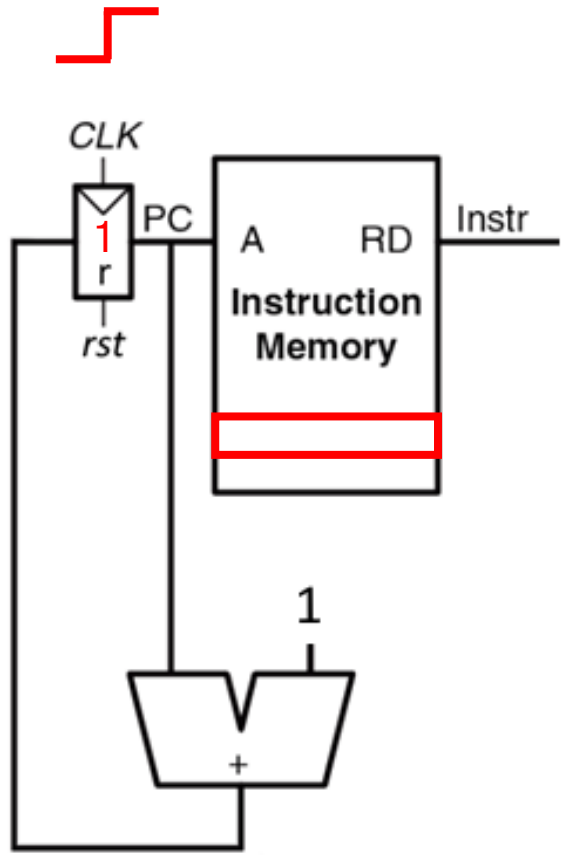




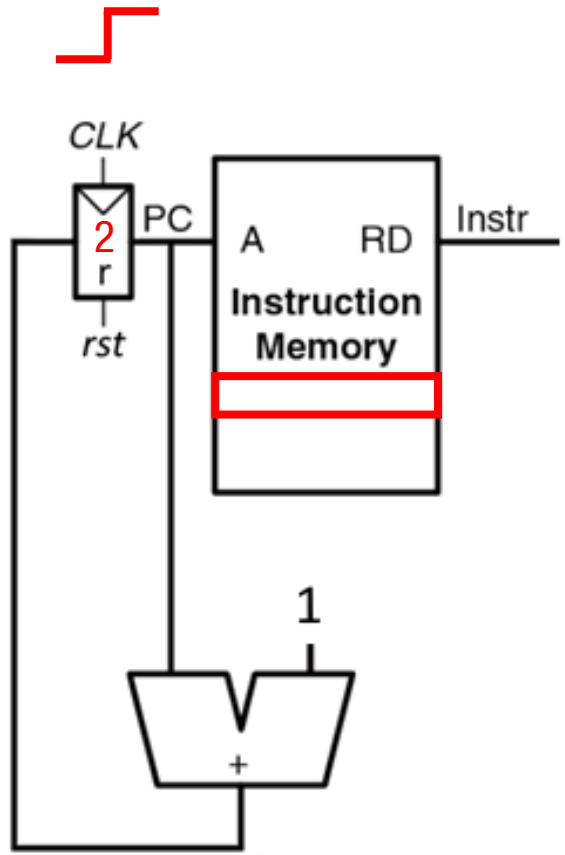
0010101101101010111_2



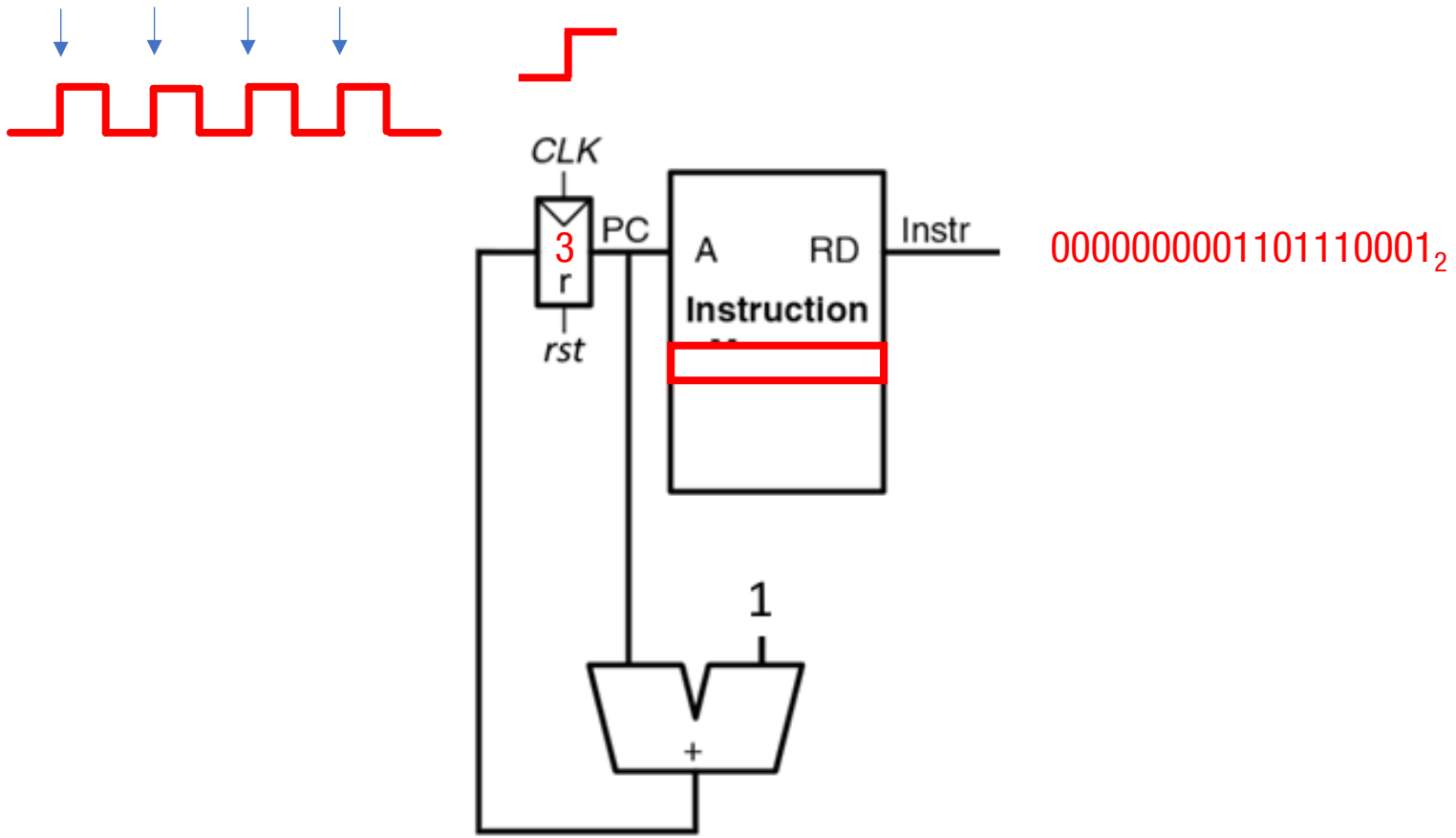
1111111001101010111_2



0000000001101110001_2

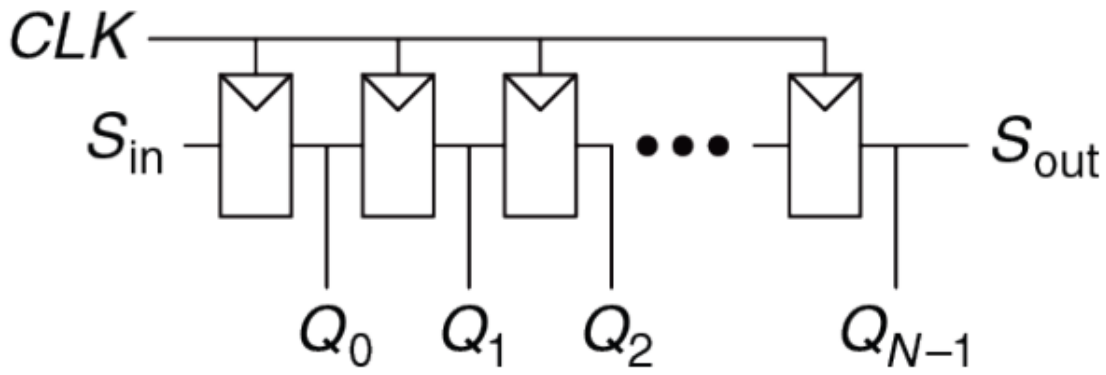


1110110000000010010_2

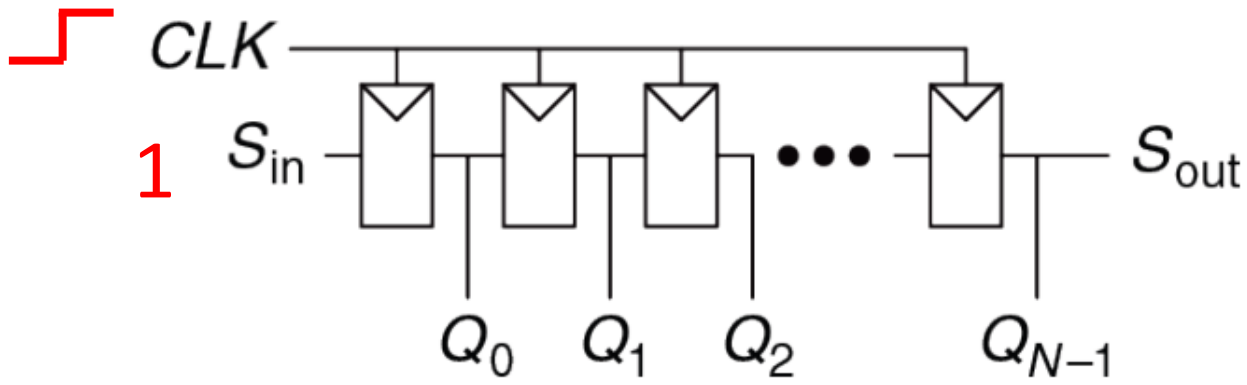


0000000001101110001_2

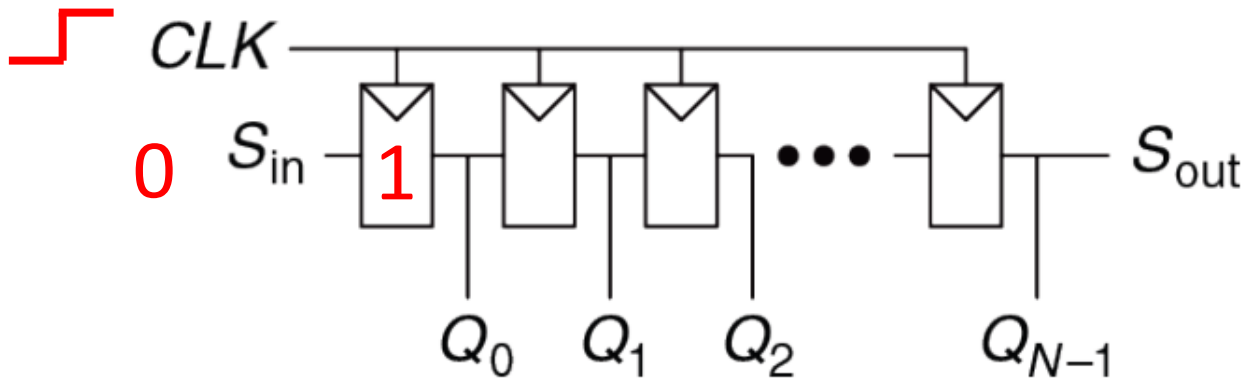
Сдвиговый регистр



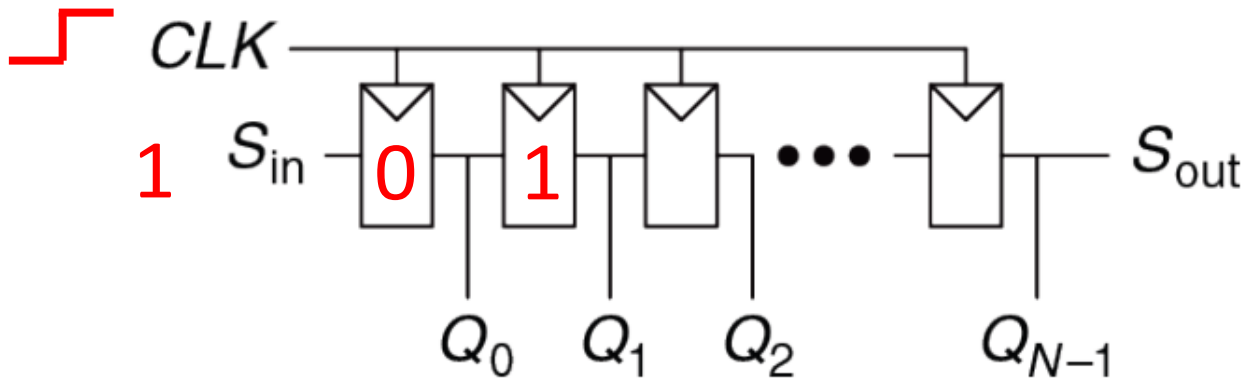
Сдвиговый регистр



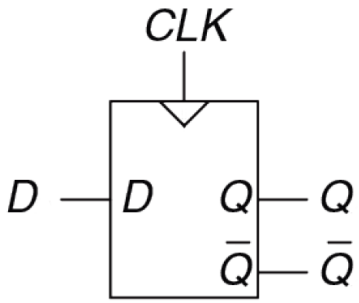
Сдвиговый регистр

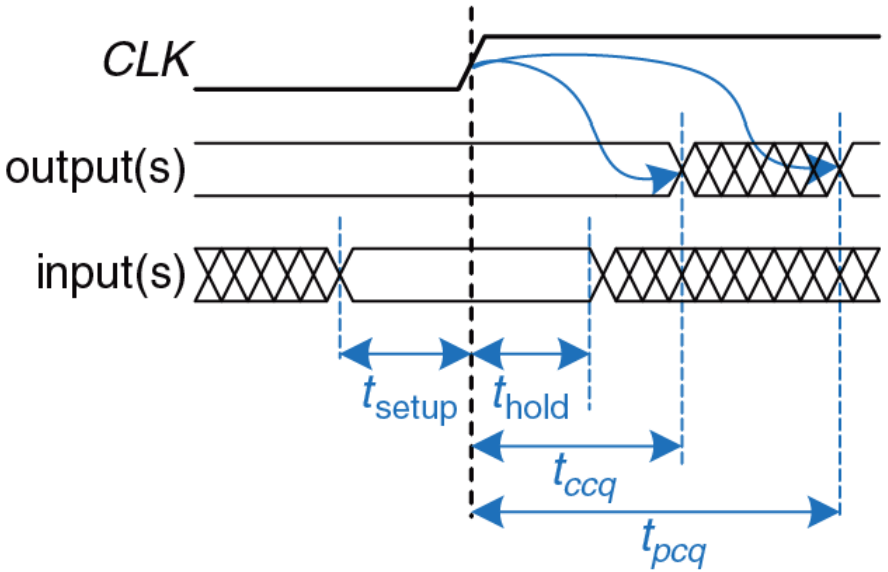


Сдвиговый регистр

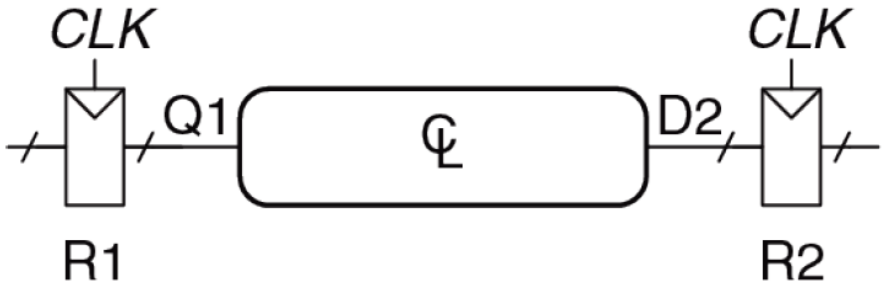


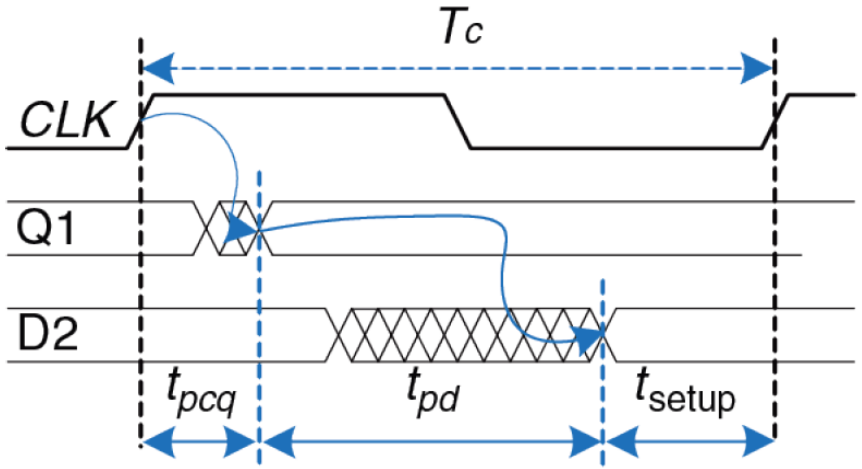
Временные характеристики



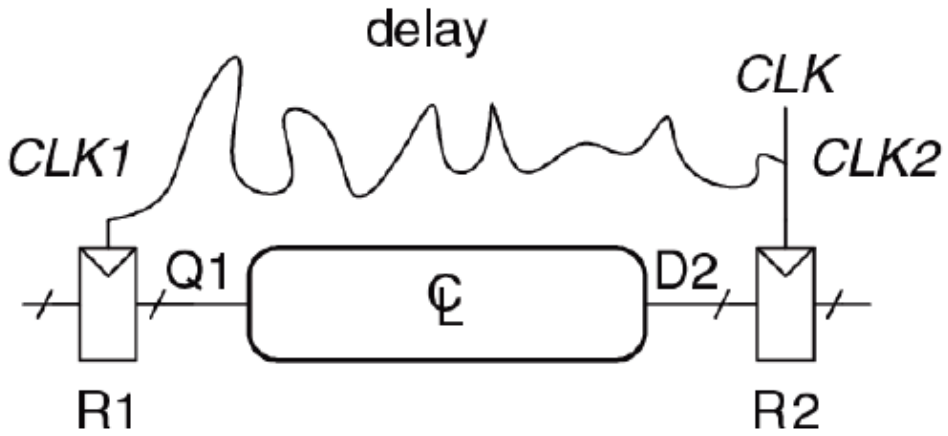


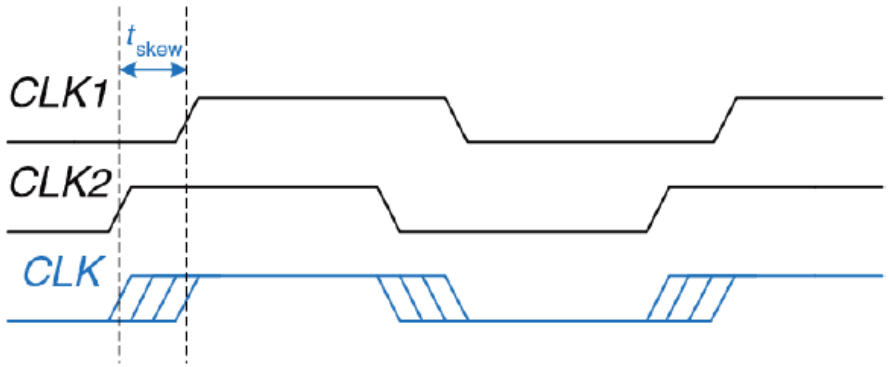
Временные характеристики



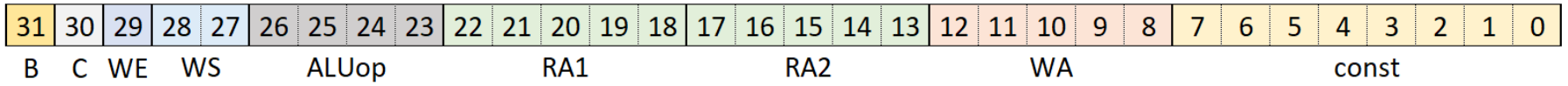


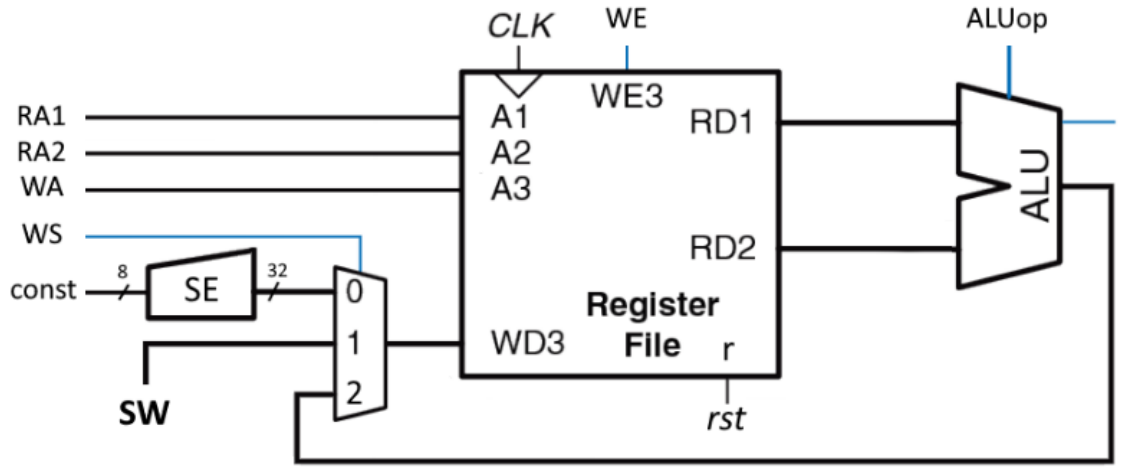
Расфазировка

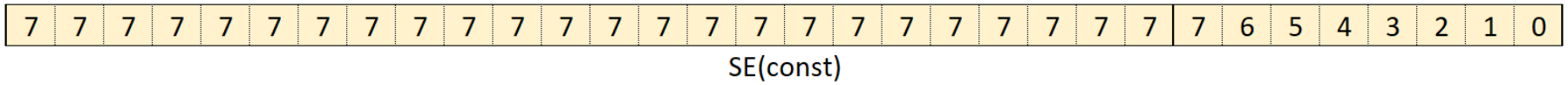




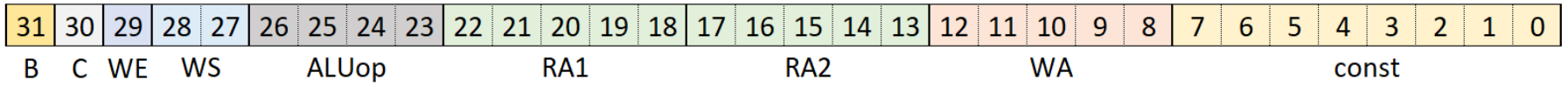
Программируемое устройство

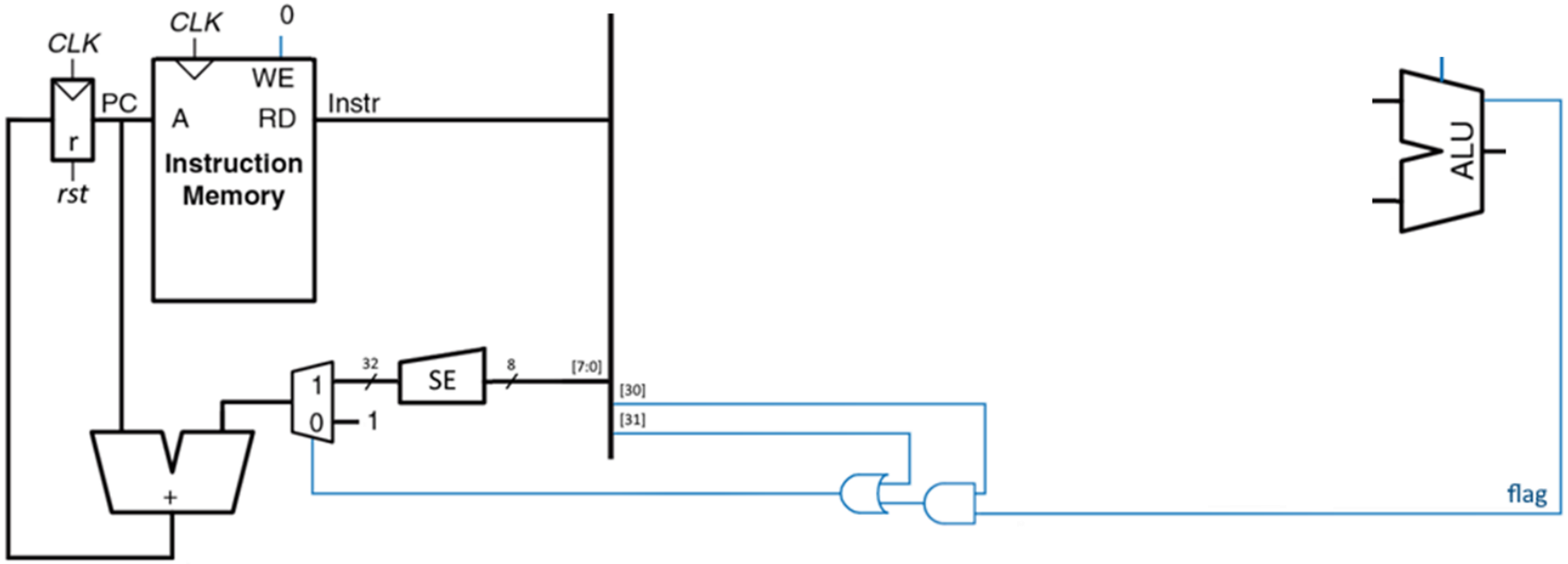




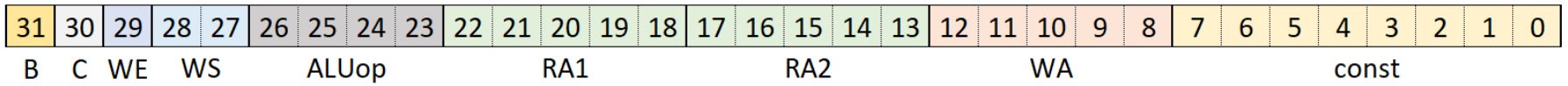


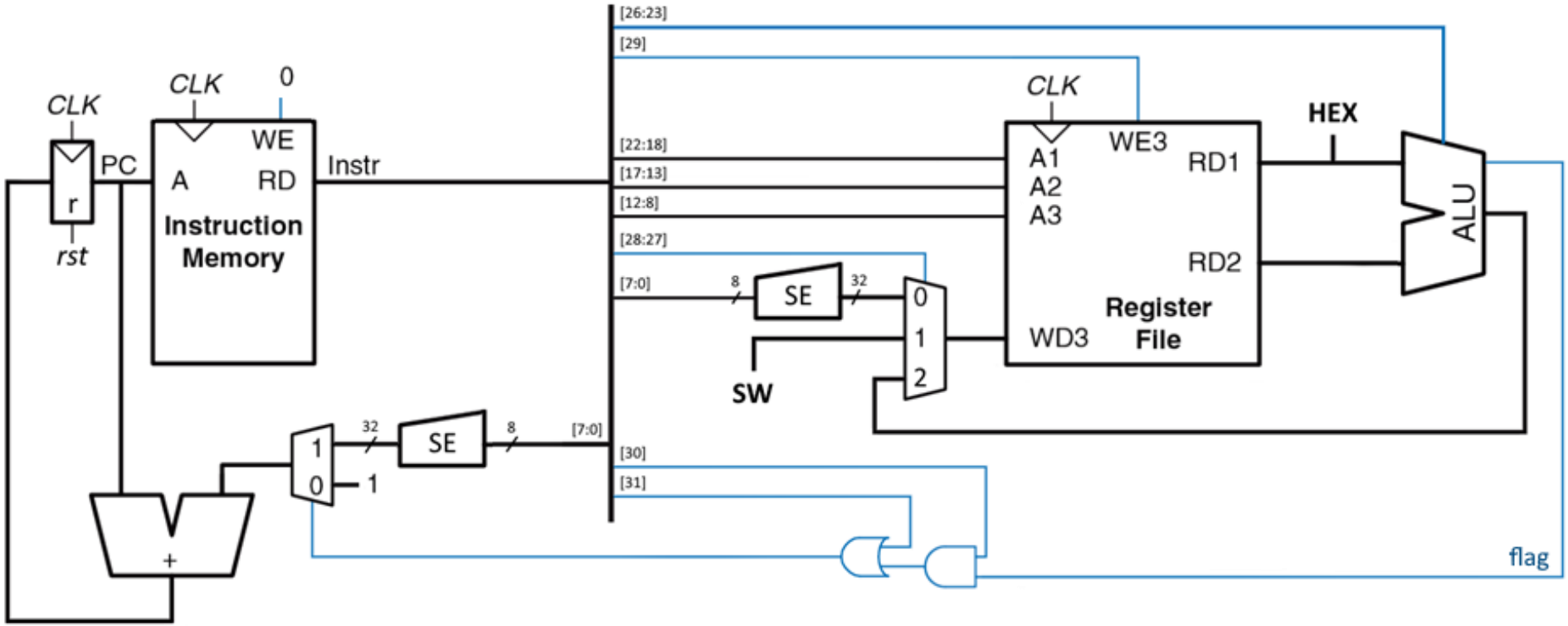
Программируемое устройство





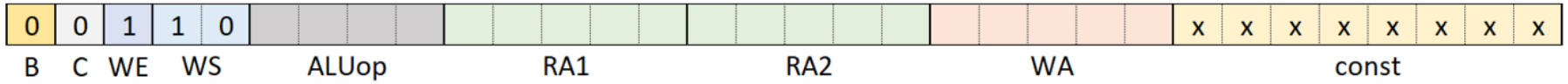
Программируемое устройство

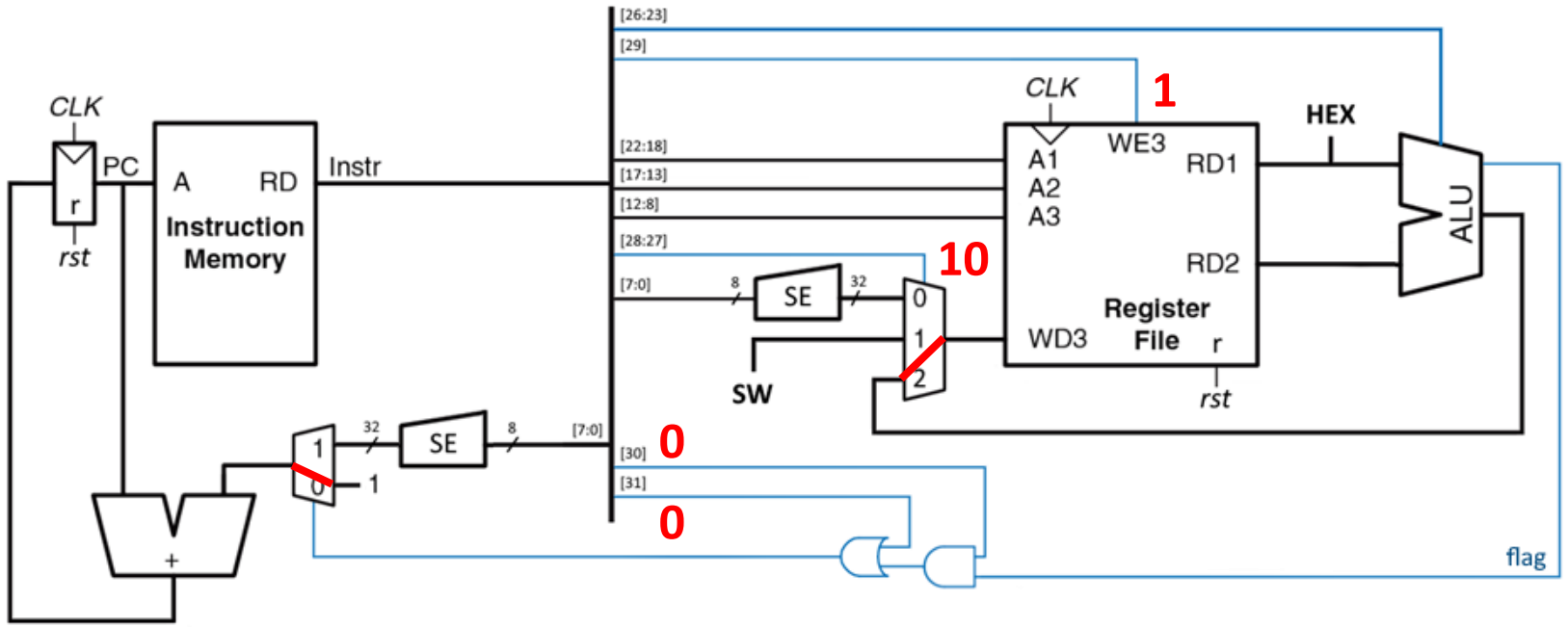




1. Операция на АЛУ

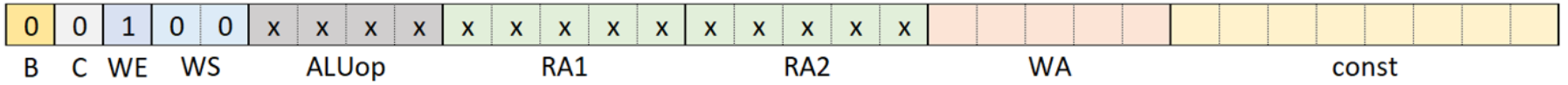
reg[WA] ← **reg**[RA1] (*ALUop*) **reg**[RA2]

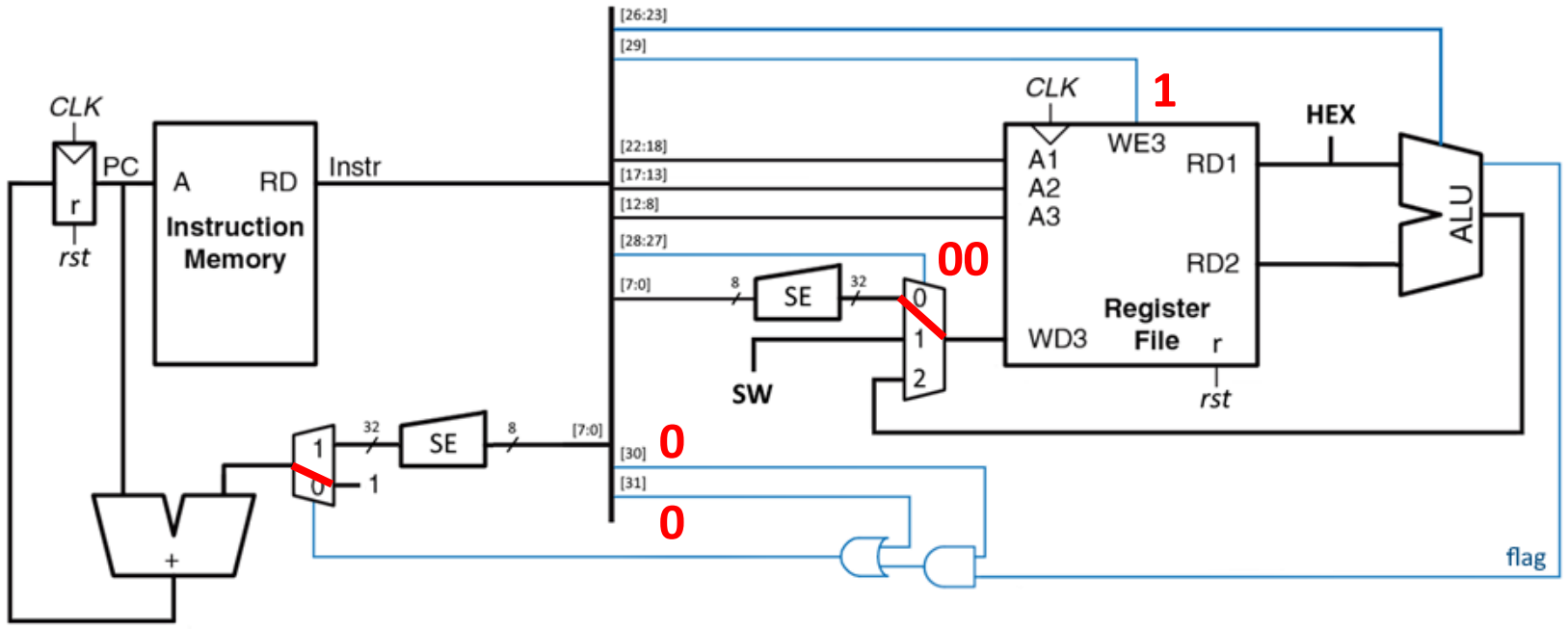




2. Загрузка константы

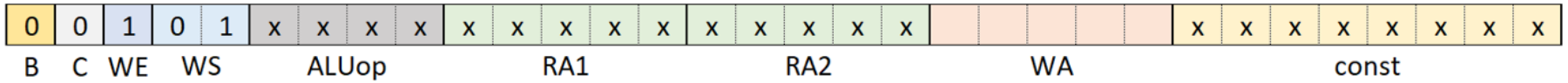
reg[WA] ← const

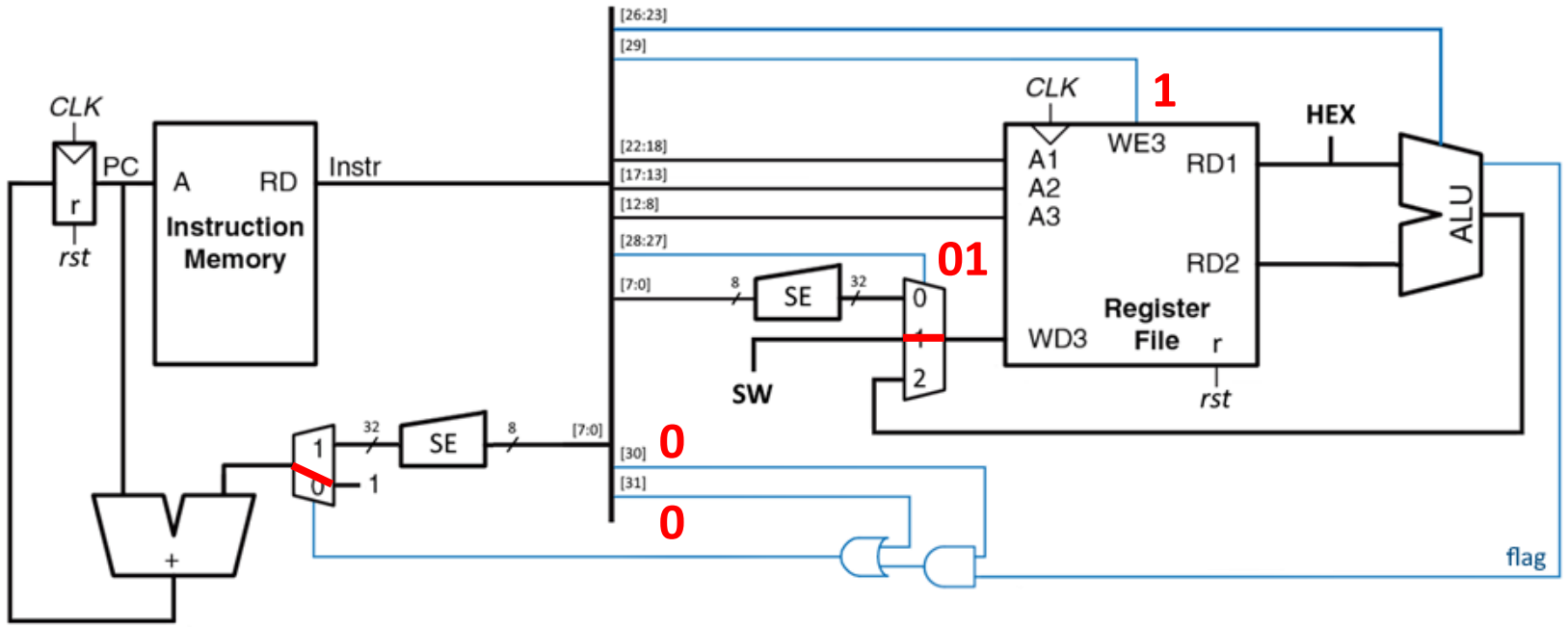




3. Загрузка с внешних устройств

reg[WA] ← switches

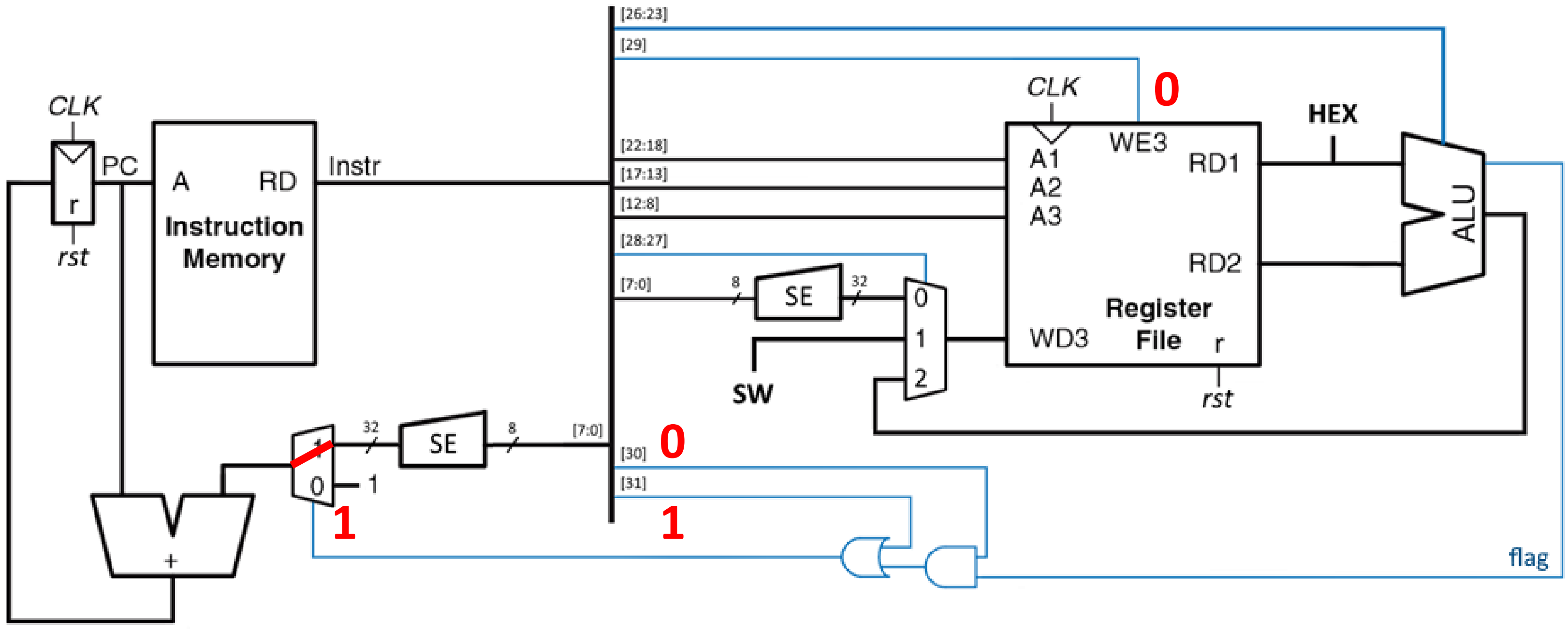




4. Безусловный переход

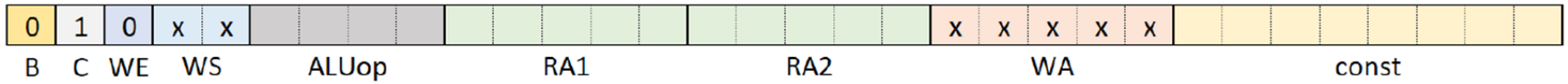
PC ← PC + const

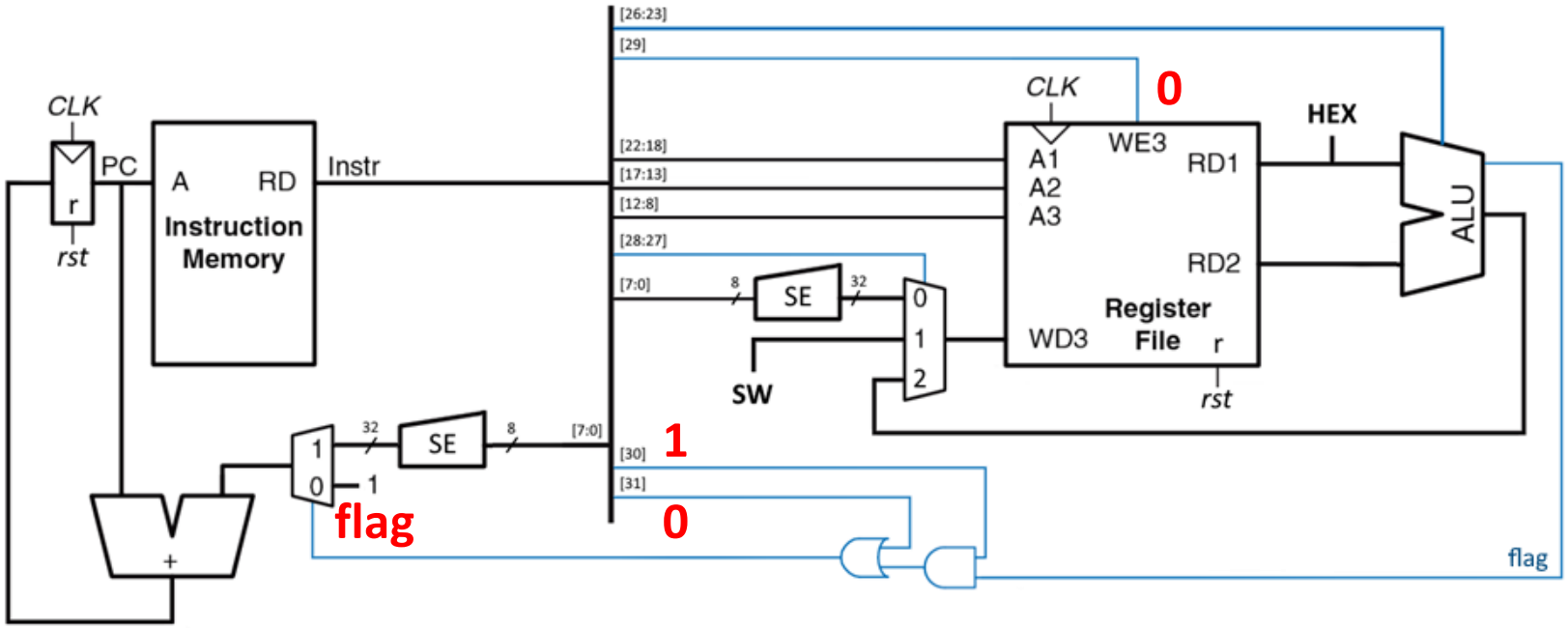
B	C	WE	WS	ALUop	RA1	RA2	WA	const
1	0	0	x x	x x x x	x x x x x	x x x x x	x x x x x	



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

Разрешение записи в регистровый файл

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	C	WE	WS		ALUop				RA1					RA2					WA					const							

Откуда пишем в регистровый файл

Условный переход

Безусловный переход