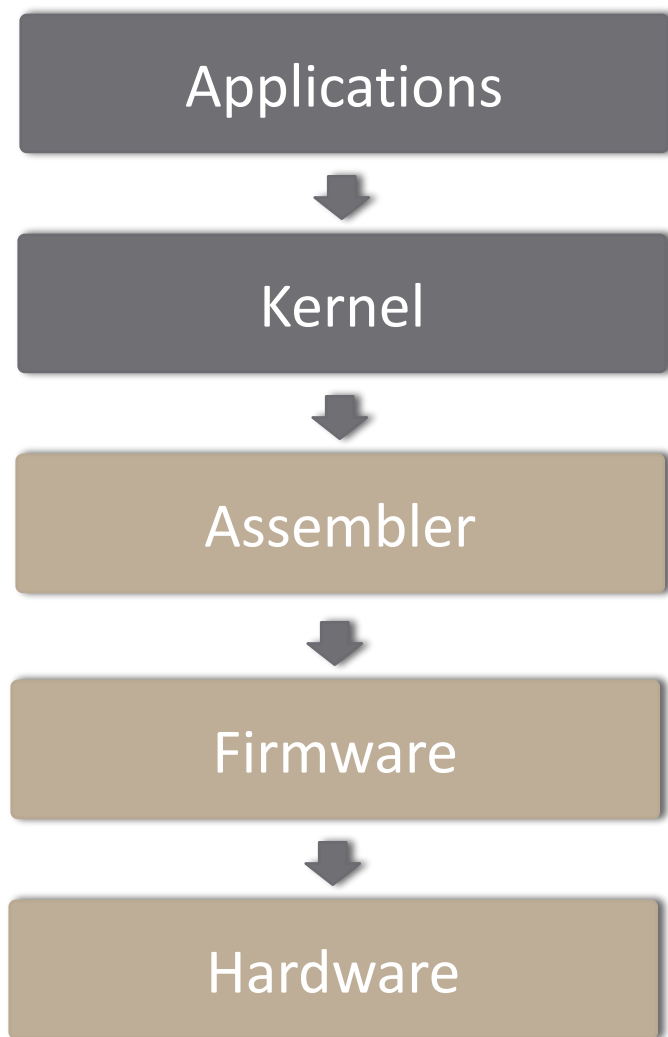


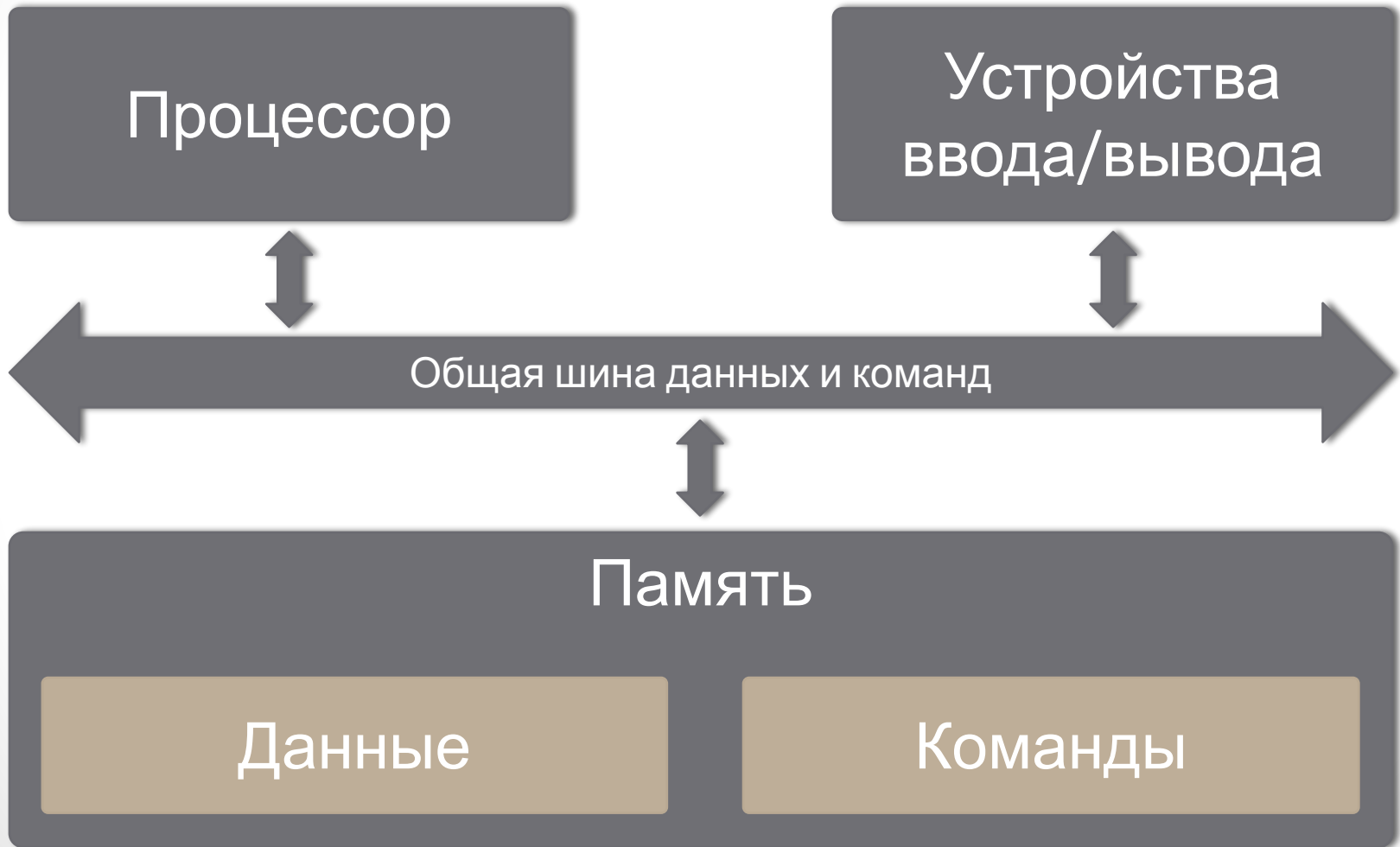
Ассемблер Atmel AVR

Занятие №1: Архитектура AVR,
схемотехника ЭВМ.

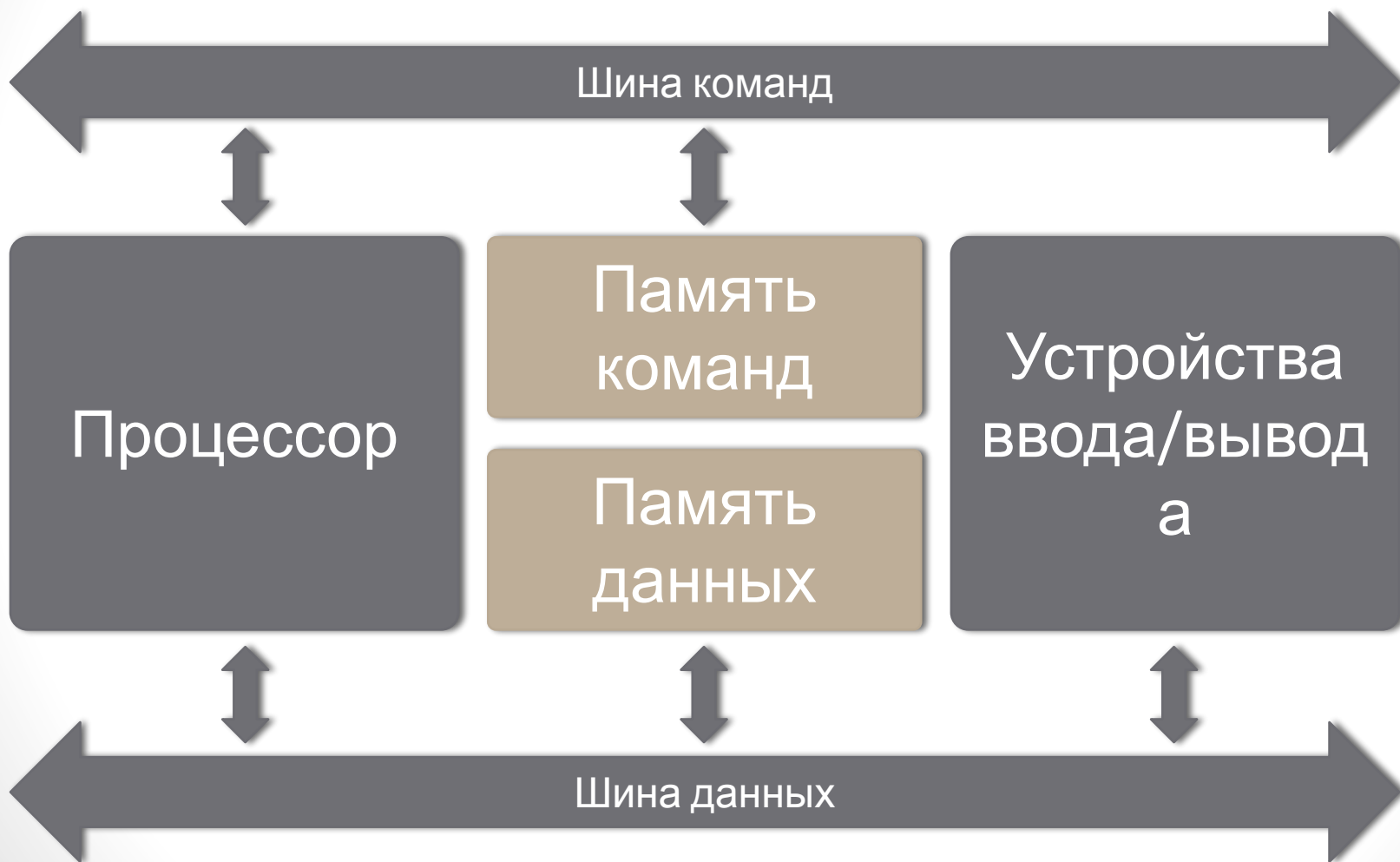
Уровни абстракции



Принстонская архитектура



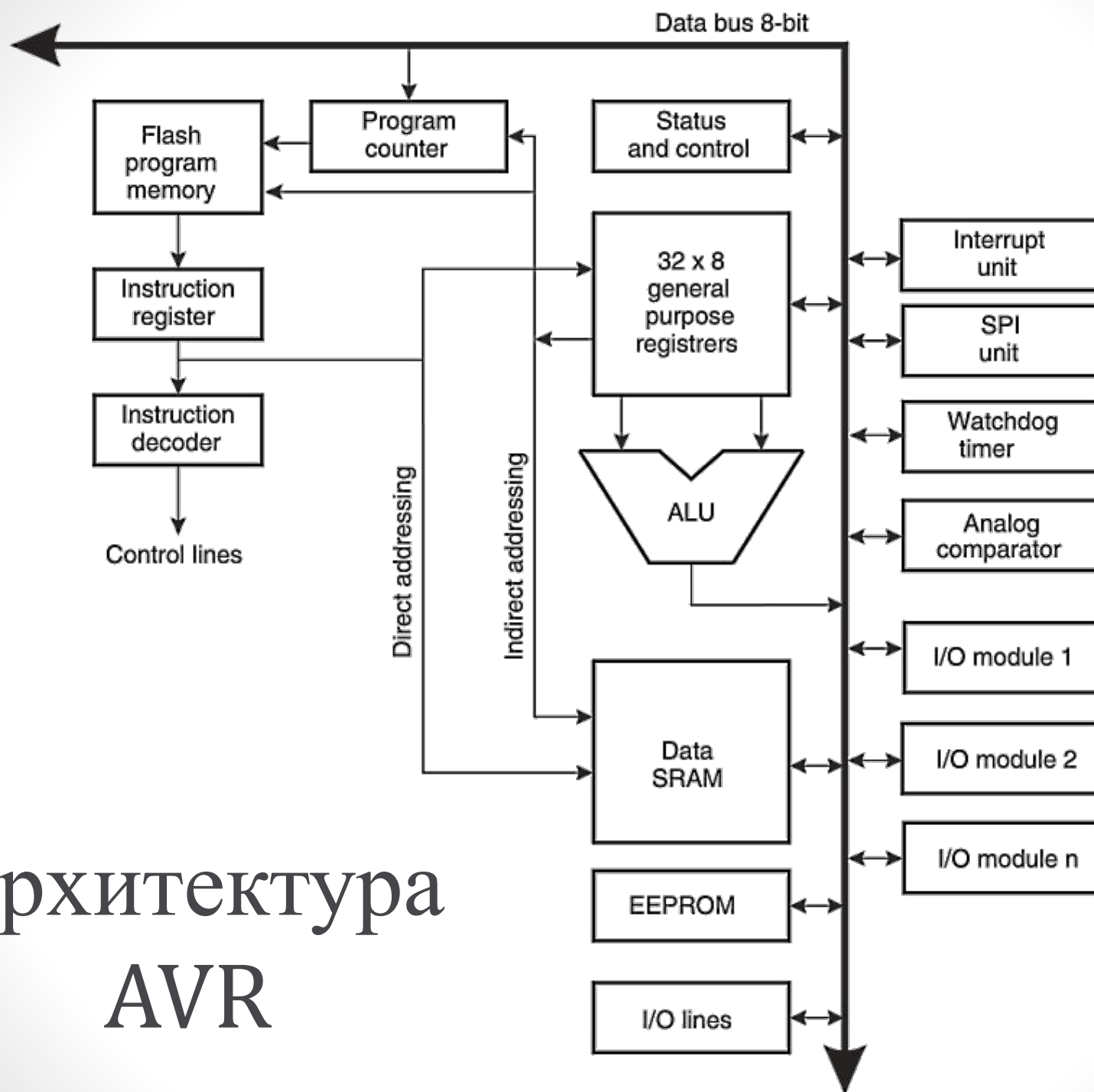
Гарвардская архитектура



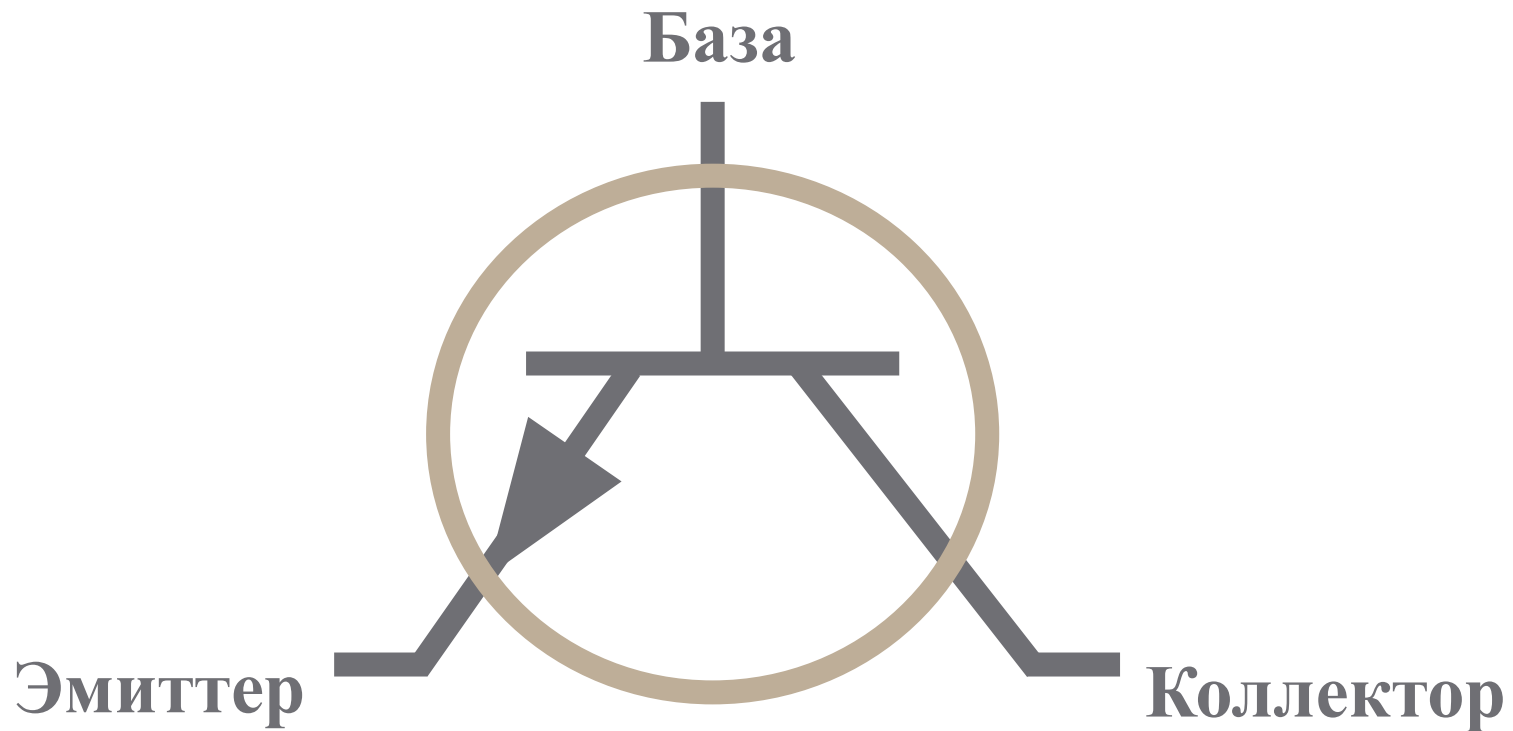
Архитектуры CISC и RISC



Архитектура AVR

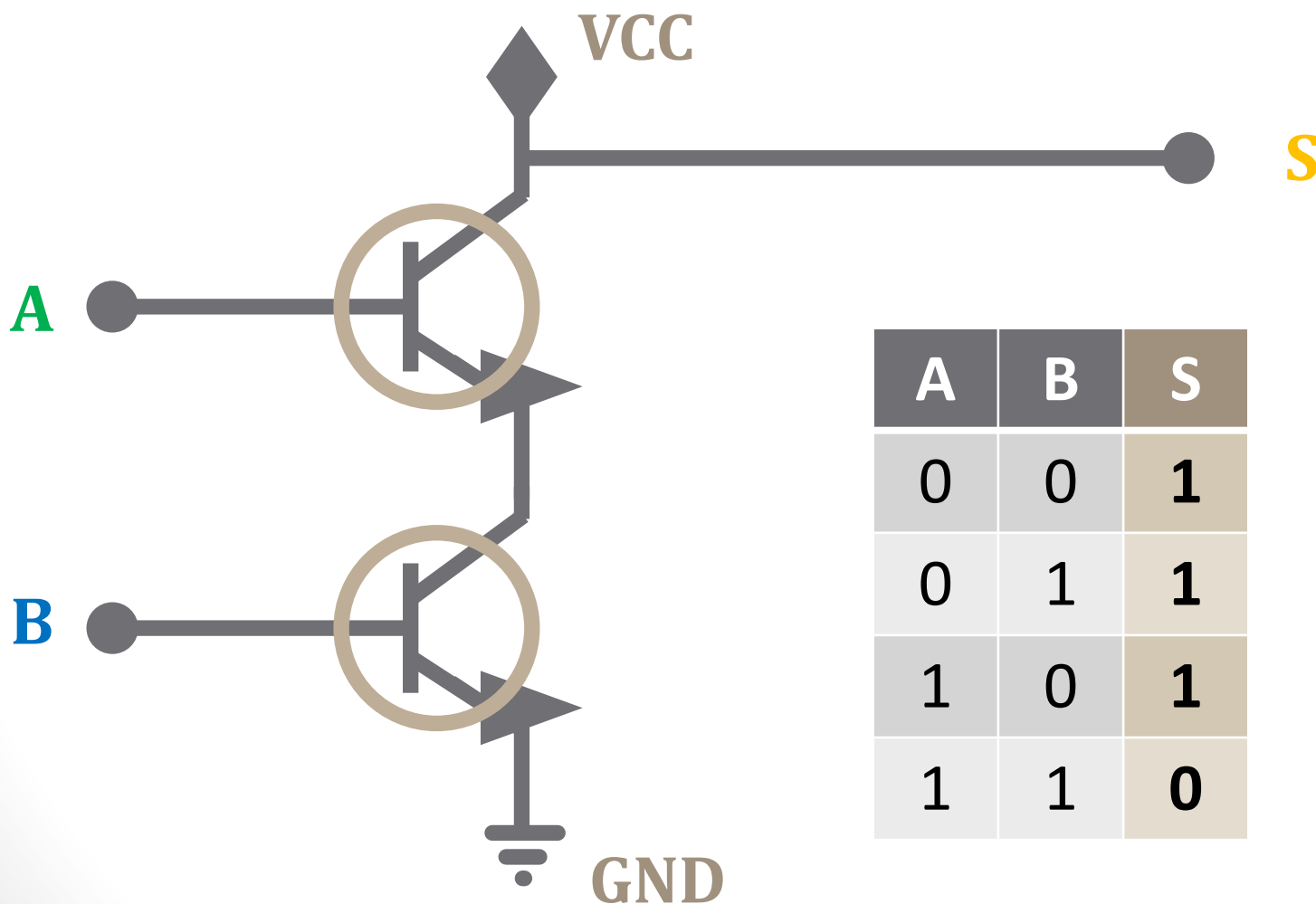


Транзистор – всему голова



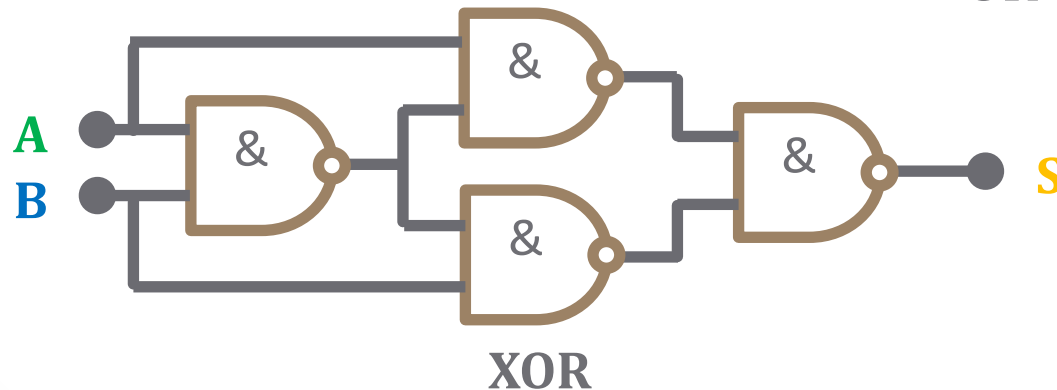
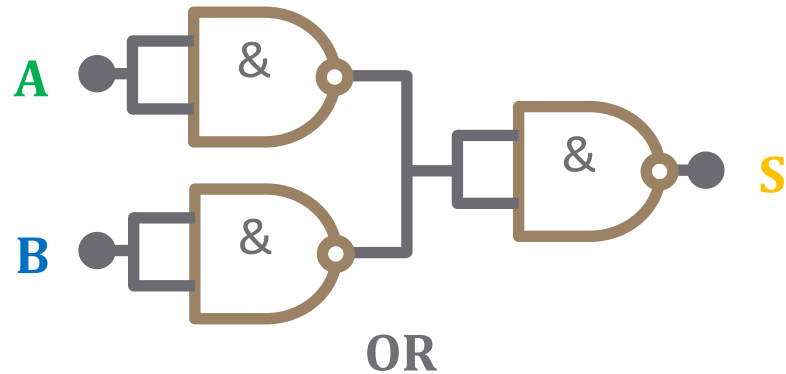
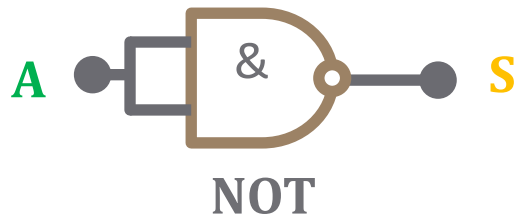
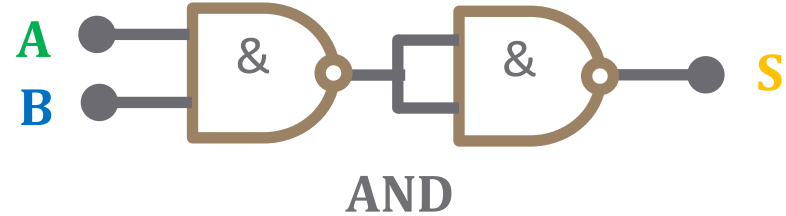
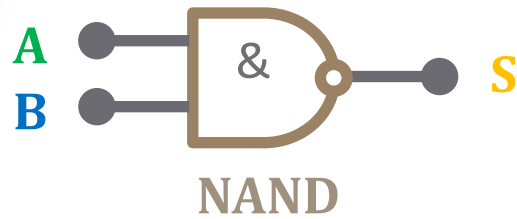
Транзистор – это кнопка, которая нажимается не пальцем, а подачей напряжения на **Базу**, после чего ток начинает протекать между **Коллектором** и **Эмиттером**.

NAND – основной базис

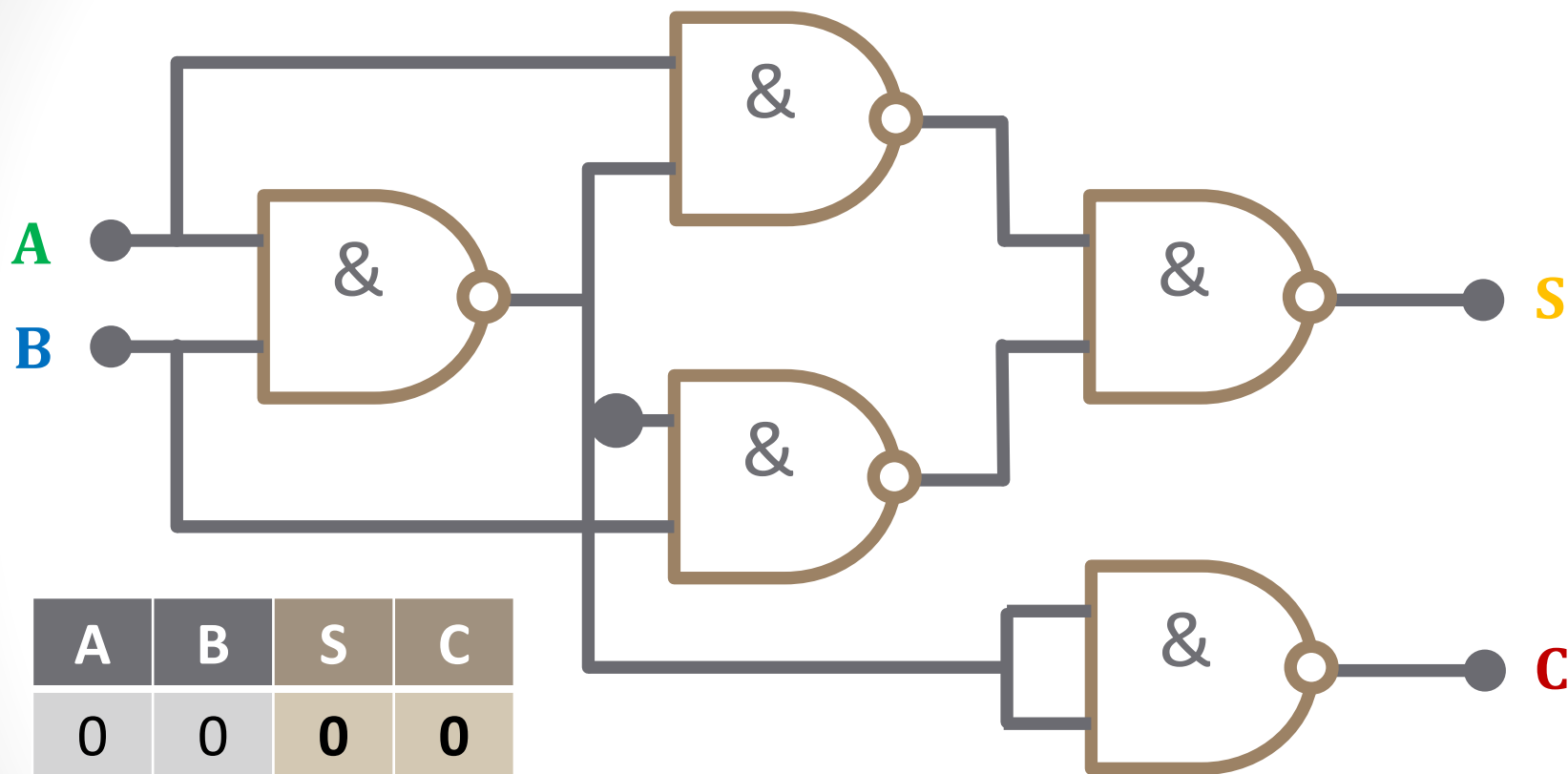


NOT AND OR XOR базис

NAND



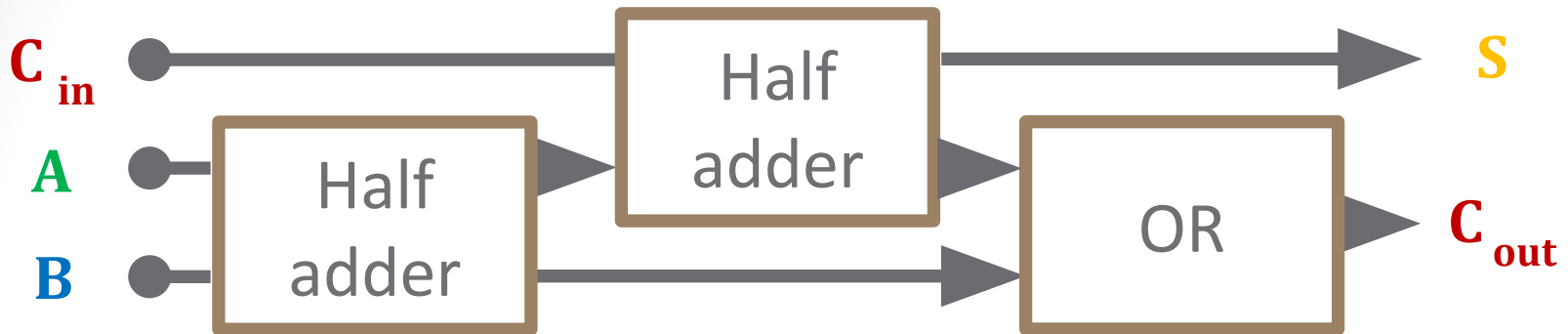
Half adder - полусумматор



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

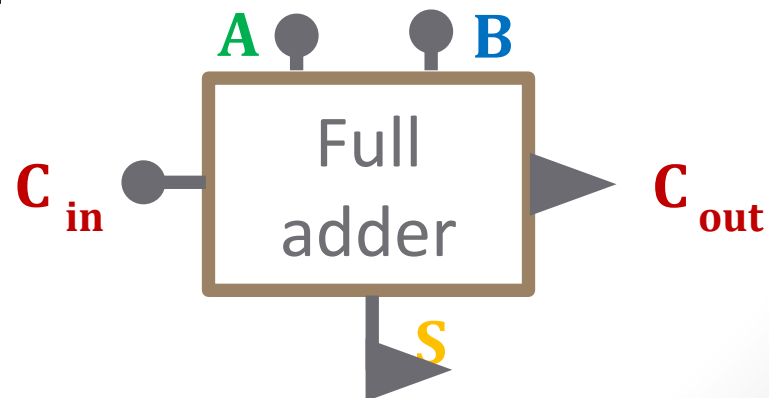
Полусумматор – суммирует два входящих бита, получая бит результата и бит переполнения.

Full adder - сумматор

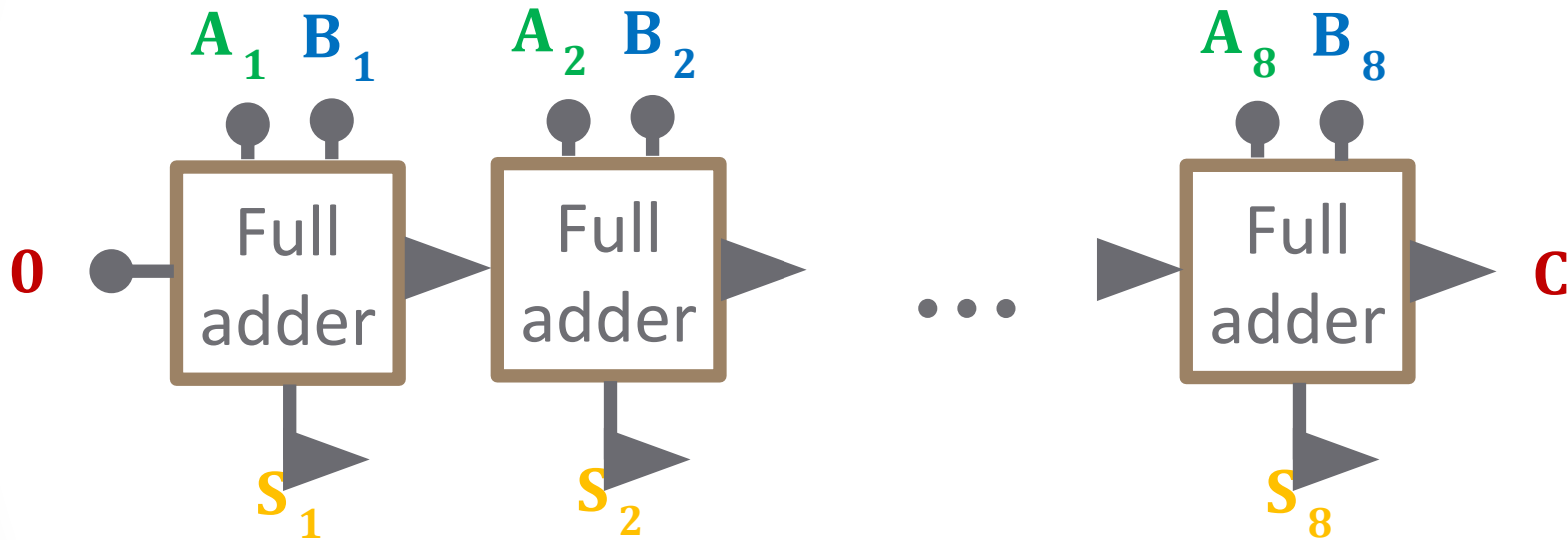


A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Сумматор – суммирует два бита и бит перехода, получая бит результата и бит переполнения.

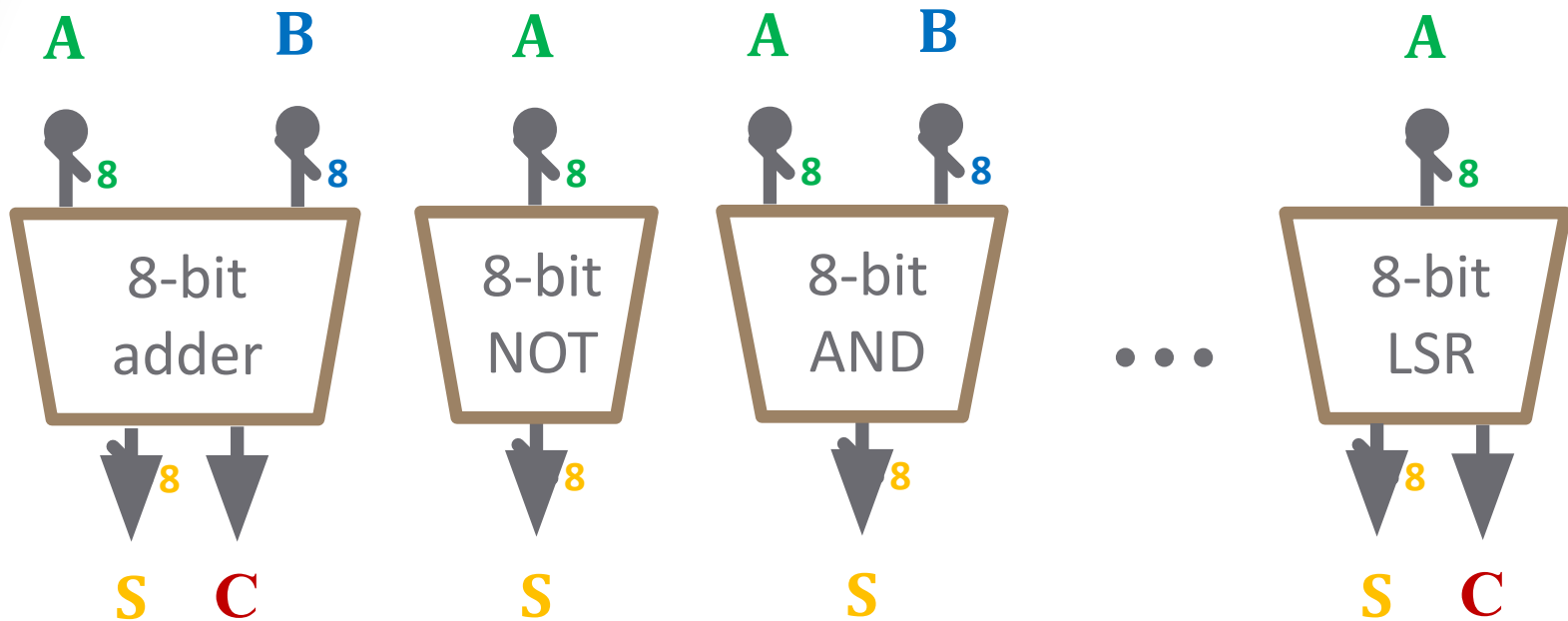


Полный 8 битный сумматор



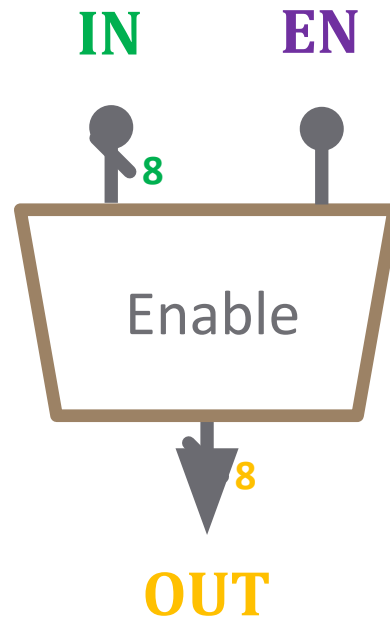
Итого: для создания полного 8 битного сумматора, основанного на базисе логических элементов NAND потребуется $2 * ((2 * 5 + 3) * 8) = 208$ транзисторов.

8 битные операции



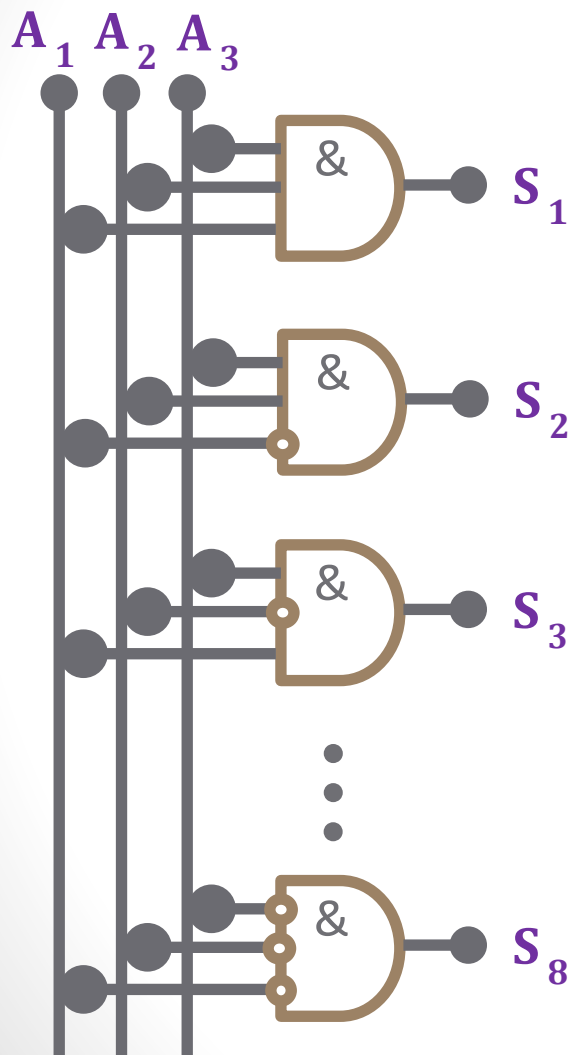
Проводники операндов (8 проводов каждый) можно подсоединить одновременно к блокам всех операций. **Итого:** на 16 входящих проводников, получится по 8 или 9 проводников с каждой операции, которые объединить нельзя (монтажный OR).

Защелка выключатель



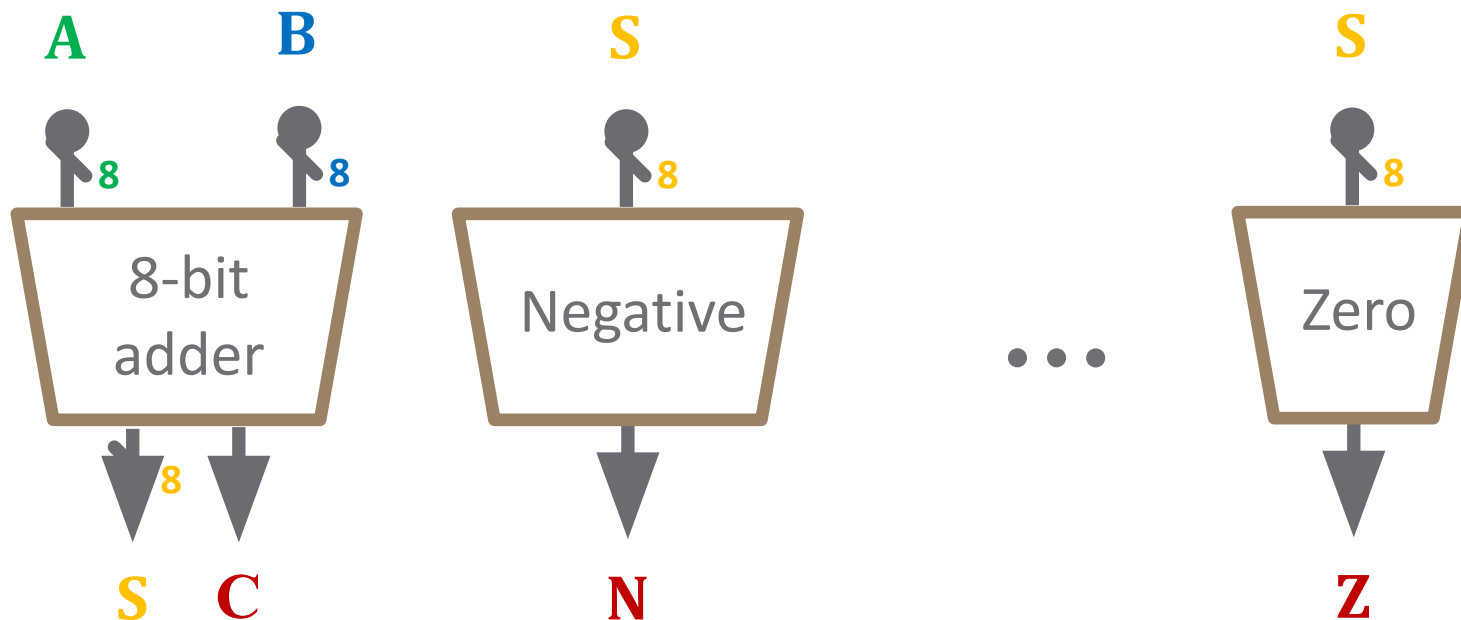
Добавив на выход каждого блока операции по выключателю, мы можем объединить все выходы получив 16 проводников входов и $8 + 1$ выходов, плюс по одному управляющему проводнику на каждую операцию.

Дешифратор



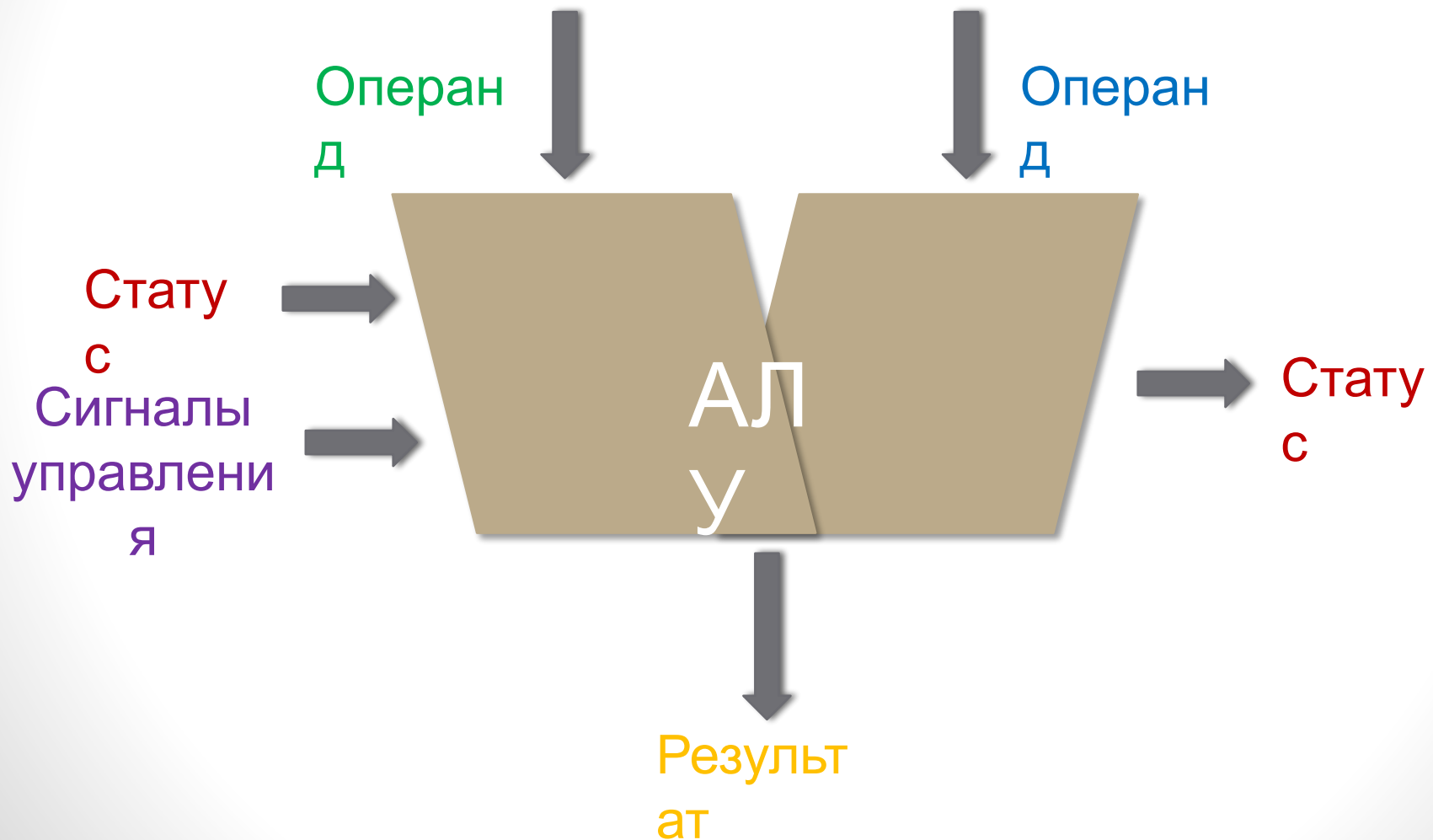
A_3	A_2	A_1	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Статус результата

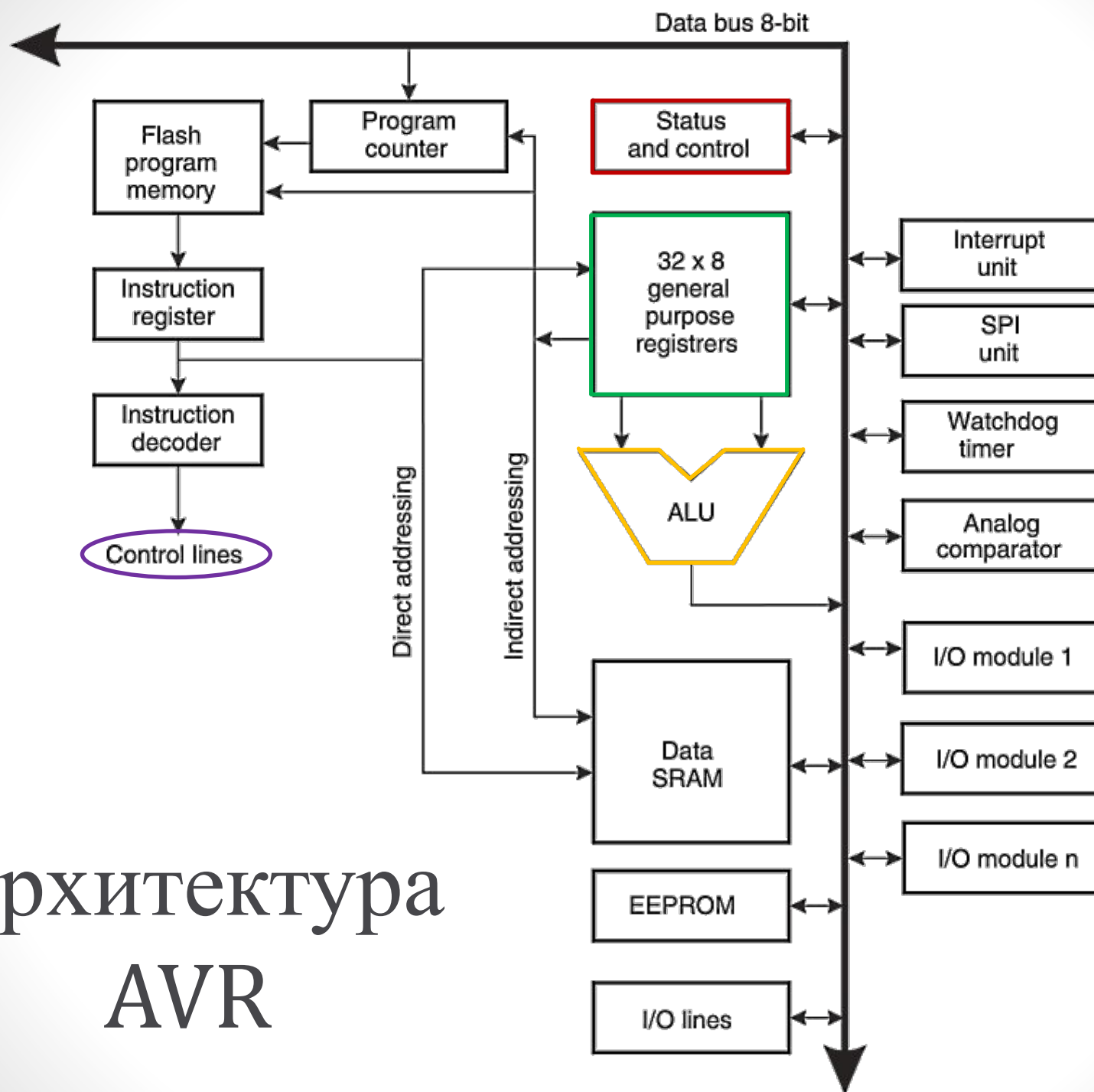


Из результата операции можно сразу же получить полезную информацию например (**C**) переполнение разряда при сложении или особый блок **Zero** который выполняет **XOR** между всеми 8 проводниками результата и если он равен 0 то $Z = 1$.

Арифметико-логическое устройство



Архитектура AVR



Регистры процессора AVR

R0		0x00
R1		0x01
...		
R15		0x0F
R16		0x10
R17		0x11
...		
R24		0x18
R25		0x19
R26	X	0x1A
R27		0x1B
R28	Y	0x1C
R29		0x1D
R30	Z	0x1E
R31		0x1F

Адрес 5 бит

Адрес 4 бита

Адрес 2 бита

Память AVR

Flash 16-bits

0x0000

Память программ

FLASHEND – 0xFFFF

SRAM 8-bits

0x0000 ПОН 0x001F

0x0020 I/O 0x005F

0x0060

Внутренняя SRAM

RAMEND

RAMEND+1

Внешняя SRAM

0xFFFF

EEPROM 8-bits

0x0000

Память EEPROM

EEPROMEND – 0xFFFF

NOP – Ничего не делать

Синтаксис:

NOP

Размер: 2

байта

0000	0000	0000	0000
------	------	------	------

Операнды: –

Счетчик: PC +=

Такты:

1	I	T	H	S	V	N ¹	Z	C
–	–	–	–	–	–	–	–	–

Определение: Операция выполняется вхолостую, ничего не происходит.

LDI – Загрузить значение в регистр

Синтаксис: LDI Rd,

Размер: 2

байта			
K			
1110	KKKK	dddd	KKKK

Операнды: $16 \leq d \leq 31$, $0 \leq K \leq 255$

Счетчик: PC +=

Такты:

1	I	T	H	S	V	N ¹	Z	C
—	—	—	—	—	—	—	—	—

Определение: Загрузить непосредственное значение из кода операции в регистр Rd.

MOV – Копировать регистр

Синтаксис: MOV Rd,

Размер: 2

Rr

байта

0010	11rd	dddd	rrrr
------	------	------	------

Операнды: $0 \leq d \leq 31, 0 \leq r \leq 31$

Счетчик: PC +=

Такты:

1	I	T	H	S	V	N ¹	Z	C
—	—	—	—	—	—	—	—	—

Определение: Копирует содержимое одного регистра в другой регистр. Исходный регистр Rr остается неизменным, в регистр назначения Rd загружается копия содержимого регистра Rr.

ADD – Сложить без переноса

Синтаксис: ADD Rd,

Размер: 2

Rr		байта	
0000	11rd	dddd	rrrr

Операнды: $0 \leq d \leq 31, 0 \leq r \leq 31$

Счетчик: PC +=

Такты:

1	I	T	H	S	V	N ¹	Z	C
–	–	+	+	+	+	+	+	+

Определение: Сложение двух регистров без добавления содержимого флага переноса (C), размещение результата в регистре назначения Rd.

The screenshot shows the AVR Studio 6.0.0.0 interface. The main window displays the assembly code for Lesson01.asm. The code includes comments and instructions for setting up the timer and testing the interrupt. The CPU register window on the left shows the values of the registers. The I/O View window on the right shows the I/O devices and their values.

Assembly Code:

```

OCT    CPU: R16
SBI    DDIF, PR1
CBI    PORTD, PR1

LDI    R17, 0x01000000
LDI    R16, 0x01000000
SDD    R17, R16

;--- CPU1: 10000000 [10000000] 10000000
OCT    00000000

LDI    CPU1, 0x00000000
OCT    TIMER, 0x00000000

CLR    ZRNC

REPEAT_01:
;--- CPU1: 10000000
CALL    CPU1_START
LDI    R16, 0x01000000
CALL    CPU1_STOP
LDI    R16, 0x01000000
CALL    CPU1_STOP
LDI    R16, 0x01000000
CALL    CPU1_STOP
LDI    R16, 0x01000000
CALL    CPU1_STOP

LDI    CPU1, 0x00000000
OCT    CPU1, 0x00000000
LDI    CPU1, 0x00000000
OCT    CPU1, 0x00000000

```

Register Window:

Name	Address	Value	Bits
MCUCR	0x05 (0x55)	0x00	00000000
MCUCSR	0x04 (0x54)	0x00	00000000
OSCCAL	0x01 (0x51)	0x00	00000000
SPICR	0x00 (0x50)	0x00	00000000
SP	0x00 (0x50)	0x00	00000000
SPMCR	0x07 (0x57)	0x00	00000000
SHR0	0x02 (0x52)	0x00	00000000

I/O View:

Name	Value
AD_CONVERTER	0x000000
ANALOG_COMPARATOR	0x000000
CPU	0x000000
EEPROM	0x000000
EXTERNAL_INTERRUPT	0x000000
PORTD	0x000000
PORTC	0x000000
PORTB	0x000000
PORTA	0x000000
SP	0x000000
TIMER_COUNTER_0	0x000000
TIMER_COUNTER_1	0x000000
TIMER_COUNTER_2	0x000000

Processor Window:

Name	Value
Program Counter	0x000000
Stack Pointer	0x000000
X pointer	0x000000
Y pointer	0x000000
Z pointer	0x000000
Cycle Counter	13
Frequency	1.0000 MHz
Stop Watch	13.00 us
Stack	0x000000
Registers	0x000000

Memory Window:

Program	Addr	Value	Address	Value
0x000000	11	00	00	00
0x000000	12	00	00	00
0x000000	13	00	00	00
0x000000	14	00	00	00
0x000000	15	00	00	00
0x000000	16	00	00	00
0x000000	17	00	00	00
0x000000	18	00	00	00
0x000000	19	00	00	00
0x000000	20	00	00	00
0x000000	21	00	00	00
0x000000	22	00	00	00
0x000000	23	00	00	00
0x000000	24	00	00	00
0x000000	25	00	00	00
0x000000	26	00	00	00
0x000000	27	00	00	00
0x000000	28	00	00	00
0x000000	29	00	00	00
0x000000	30	00	00	00
0x000000	31	00	00	00

Build Window:

Segment	Begin	End	Code	Data	Comment	Size	Mem
[.text]	0x000000	0x00000e	224	0	224	0192	2.74
[.data]	0x000000	0x000000	0	0	0	1024	0.00
[.bss]	0x000000	0x000000	0	0	0	512	0.00

INC – Инкрементировать

Синтаксис: INC

Размер: 2

Rd

байта

1001	010d	dddd	0011
------	------	------	------

Операнды: $0 \leq d \leq 31$

Счетчик: PC +=

Такты:

1	I	T	H	S	V	N ¹	Z	C
–	–	–	+	+	+	+	+	–

Определение: Добавление единицы к содержимому регистра Rd и размещение результата в регистре назначения Rd.

DEC – Декрементировать

Синтаксис: DEC

Размер: 2

Rd

байта

1001	010d	dddd	1010
------	------	------	------

Операнды: $0 \leq d \leq 31$

Счетчик: PC +=

Такты:

1	I	T	H	S	V	N ¹	Z	C
–	–	–	+	+	+	+	+	–

Определение: Вычитание единицы из содержимого регистра Rd и размещение результата в регистре назначения Rd.

SUB – Вычитать без переноса

Синтаксис: SUB Rd,

Размер: 2

Rr

байта

0001	10rd	dddd	rrrr
------	------	------	------

Операнды: $0 \leq d \leq 31$, $0 \leq r \leq 31$

Счетчик: PC +=

Такты:

1	I	T	H	S	V	N ¹	Z	C
–	–	+	+	+	+	+	+	+

Определение: Вычитание содержимого регистра-источника Rr из содержимого регистра Rd, размещение результата в регистре назначения Rd.

SUBI – Вычесть значение из регистра

Синтаксис: SUBI Rd,

Размер: 2

байта			
^K 0101	KKKK	dddd	KKKK

Операнды: $0 \leq d \leq 31$, $0 \leq K \leq 255$

Счетчик: PC +=

Такты:

¹	I	T	H	S	V	N ¹	Z	C
–	–	+	+	+	+	+	+	+

Определение: Вычитание константы из содержимого регистра, размещение результата в регистре назначения Rd.