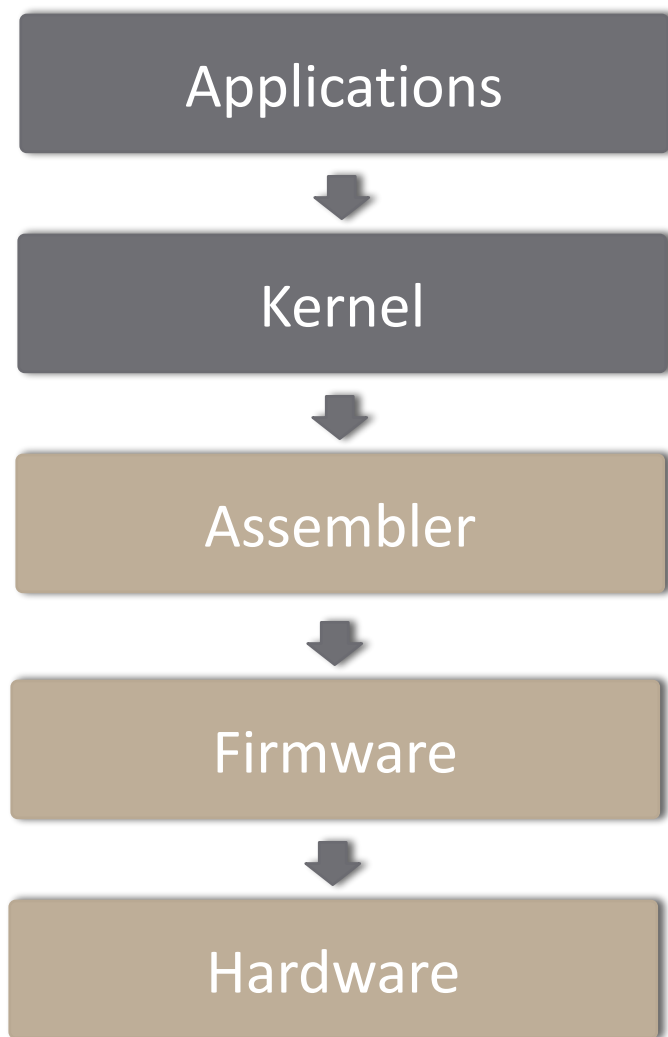


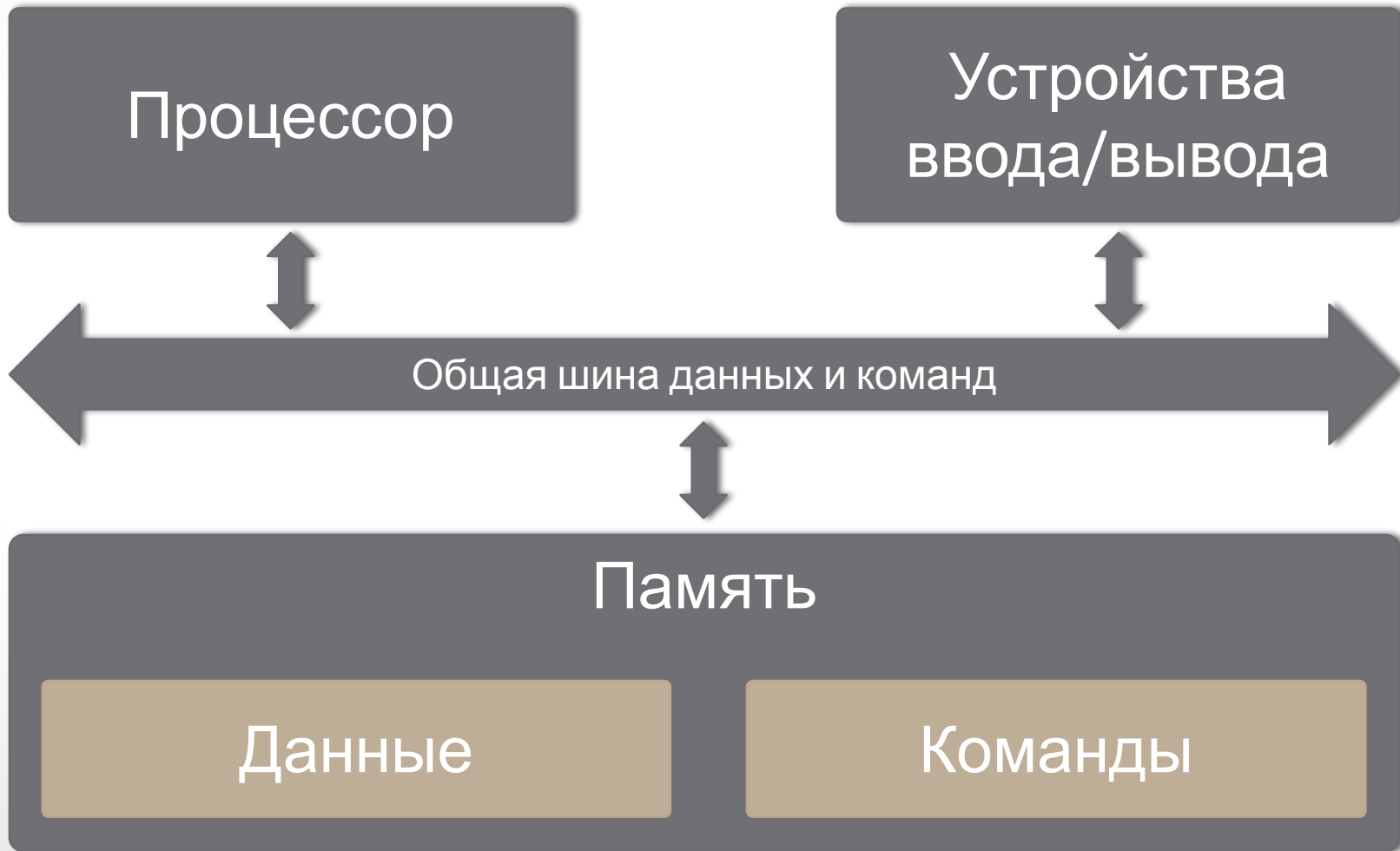
Ассемблер Atmel AVR

Занятие №1: Архитектура AVR,
схемотехника ЭВМ.

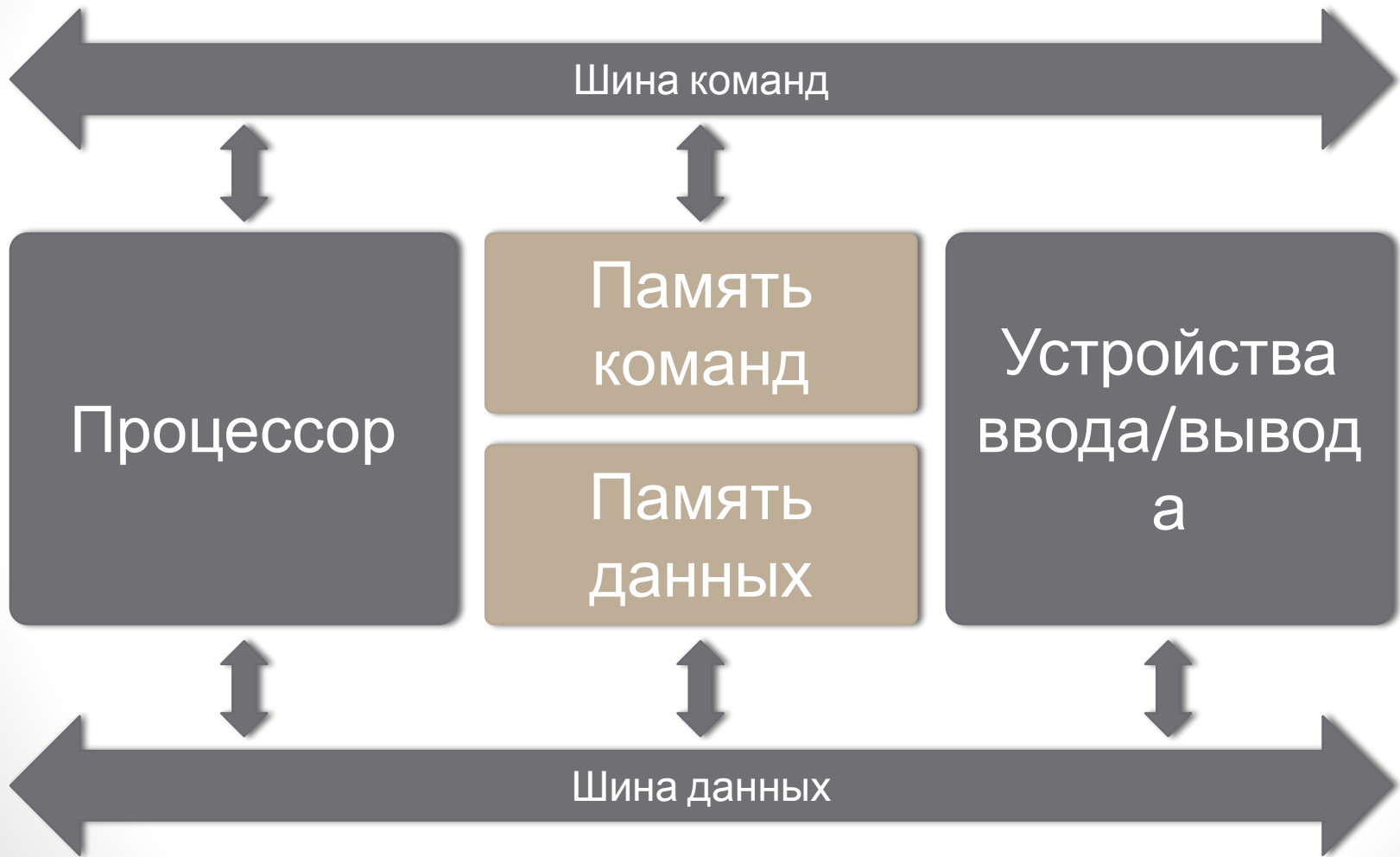
Уровни абстракции



Принстонская архитектура



Гарвардская архитектура



Архитектуры CISC и RISC

CISC

Машинные инструкции



Преобразование микрокода



Микроинструкции



Обработка микроинструкций

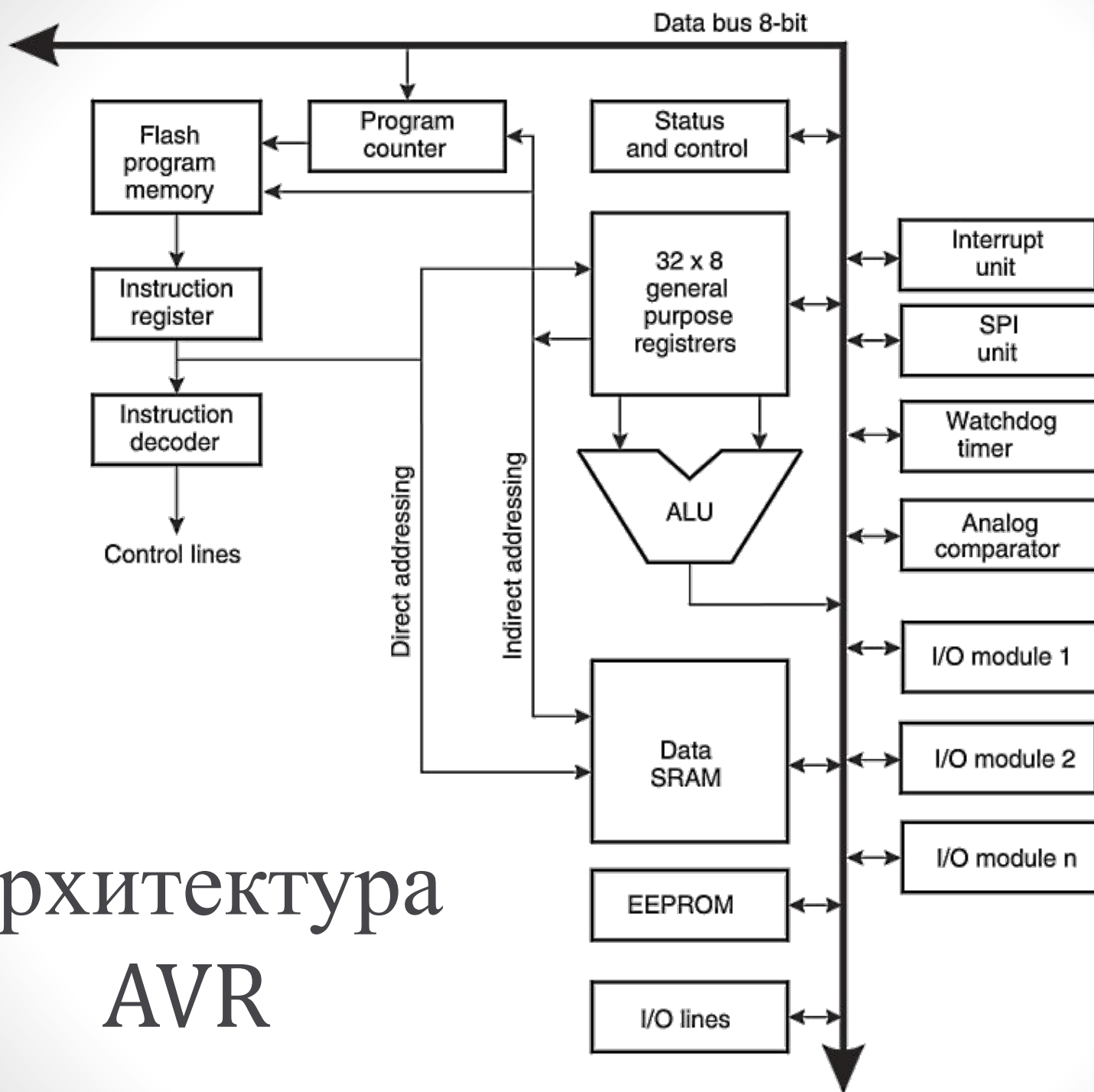
RISC

Машинные инструкции

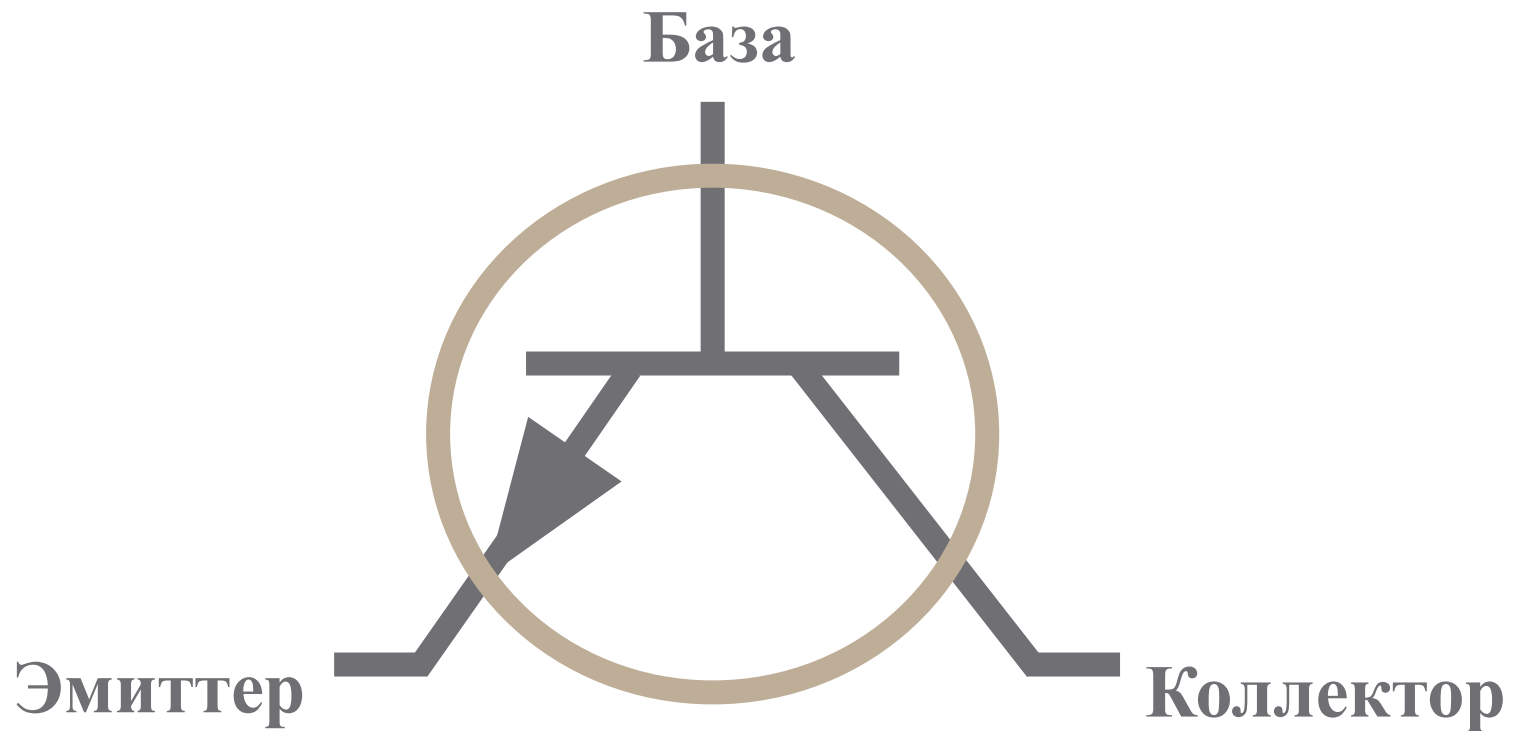


Обработка инструкций

Архитектура AVR

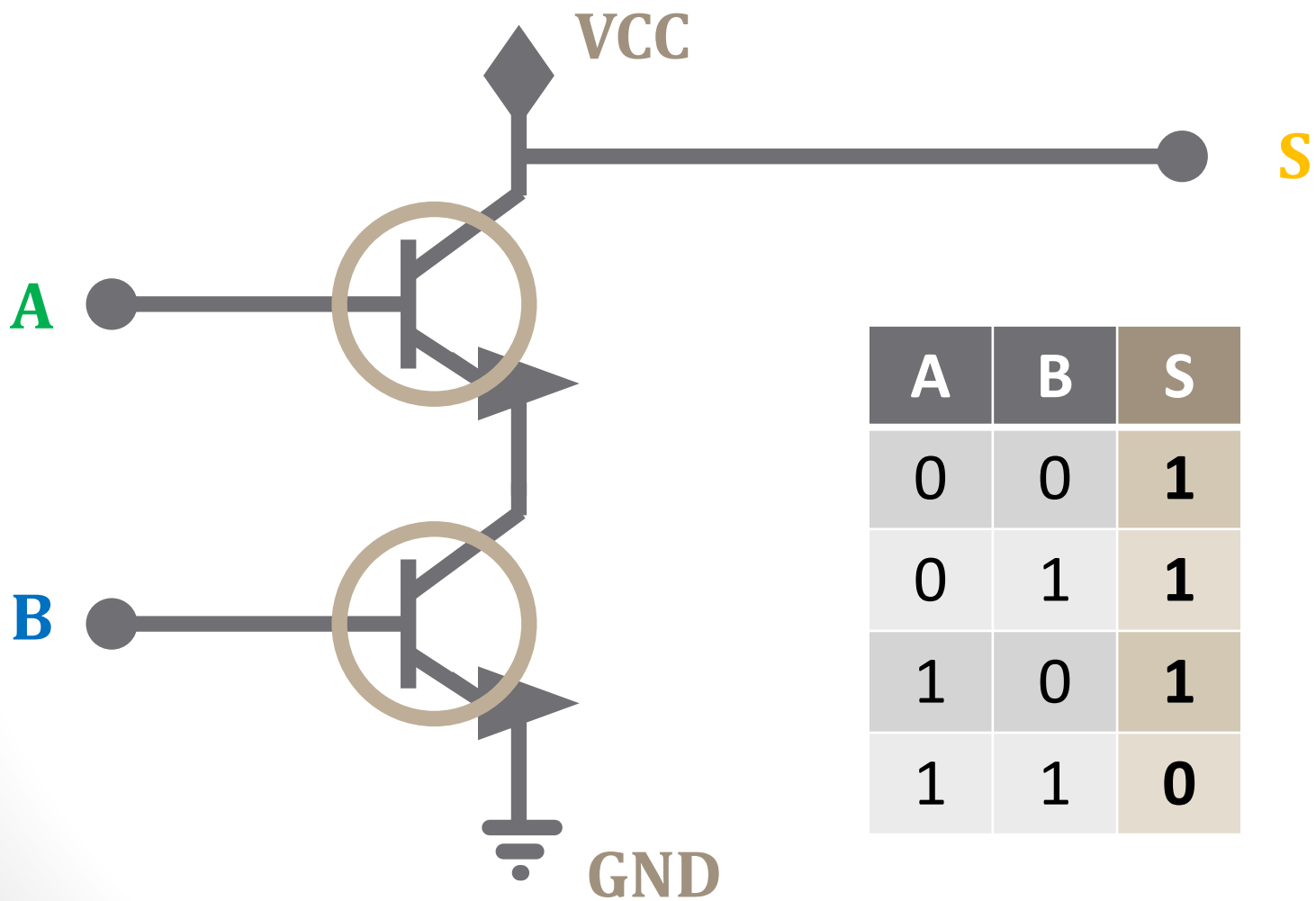


Транзистор – всему голова



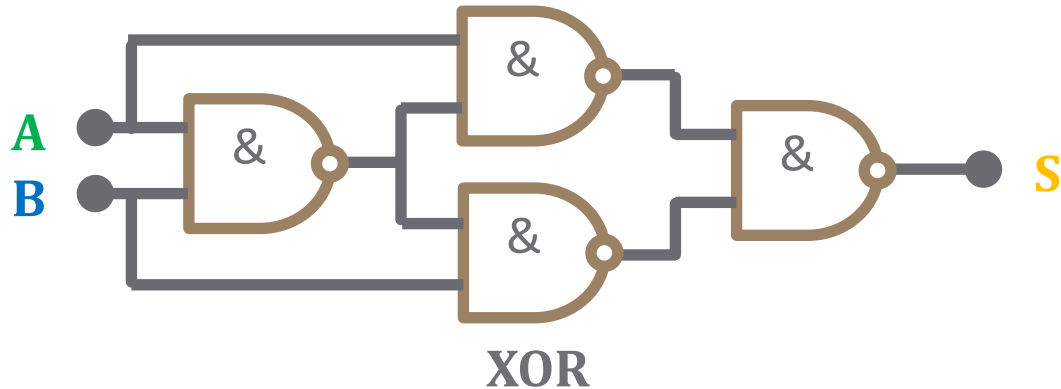
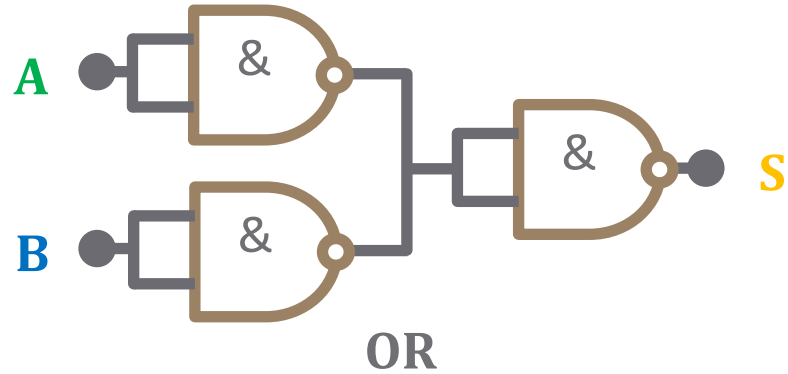
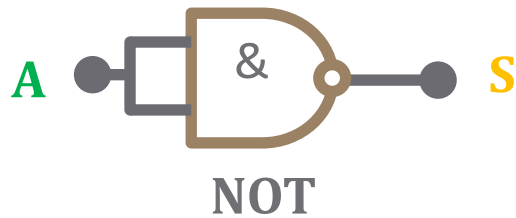
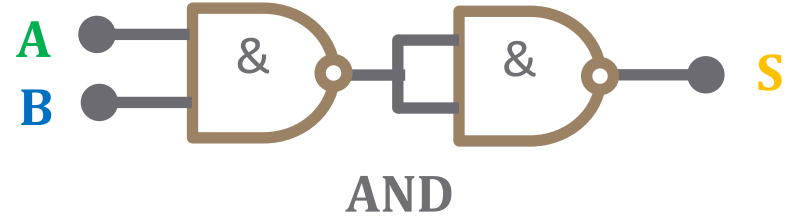
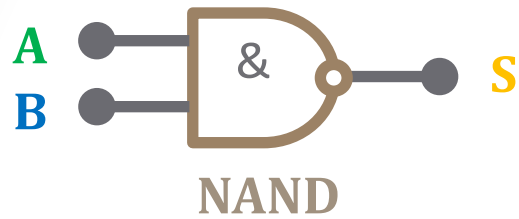
Транзистор – это кнопка, которая нажимается не пальцем, а подачей напряжения на **Базу**, после чего ток начинает протекать между **Коллектором** и **Эмиттером**.

NAND – основной базис

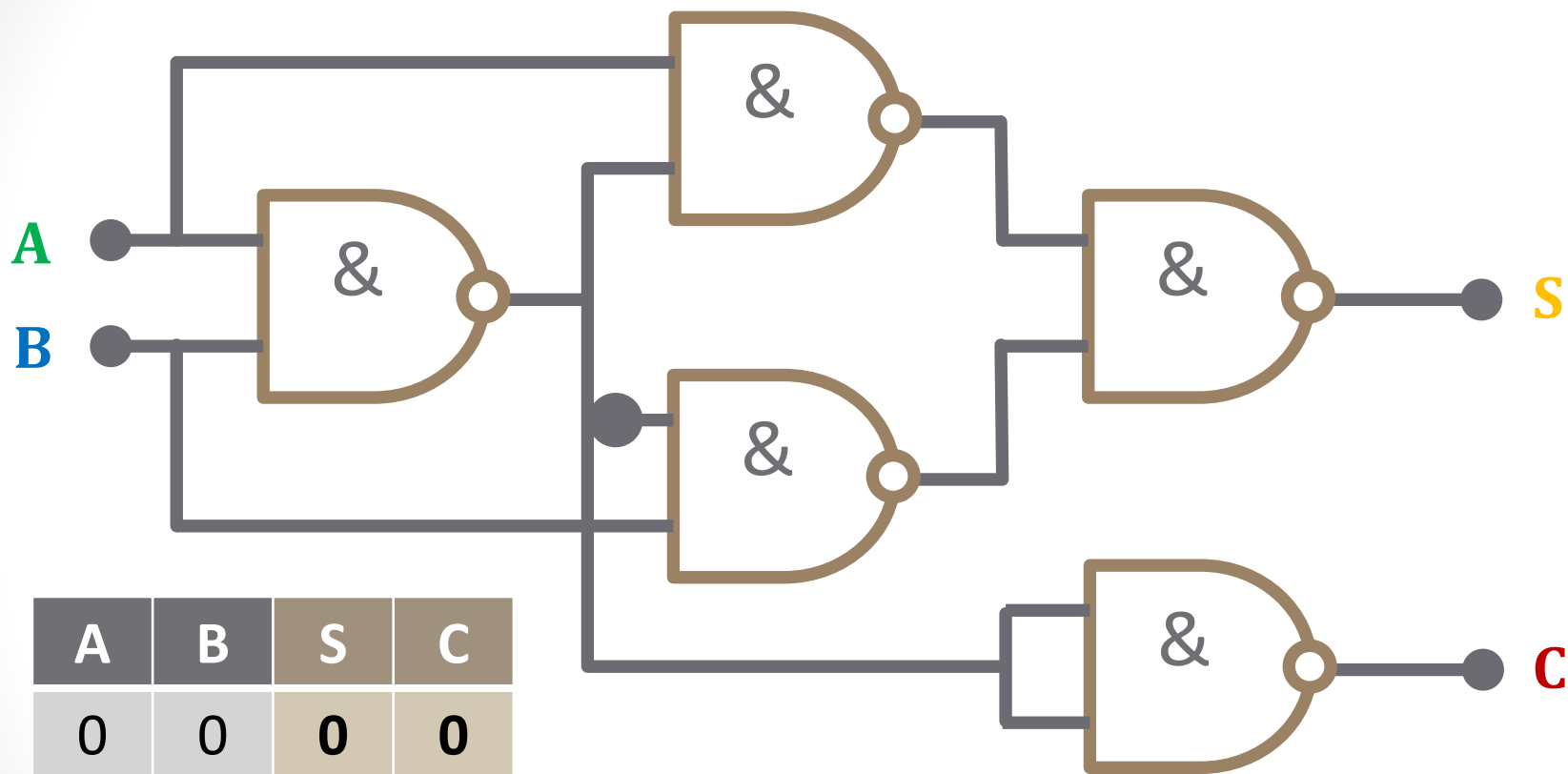


NOT AND OR XOR базис

NAND



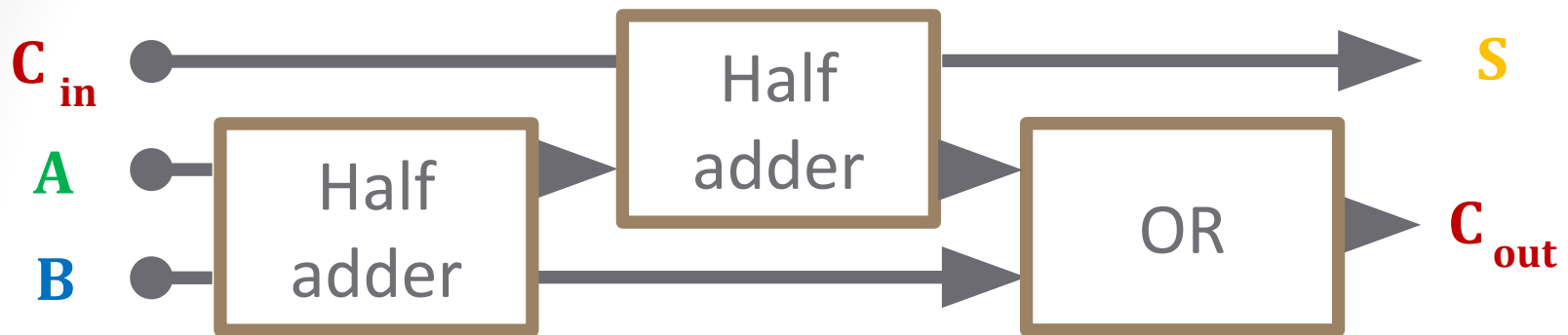
Half adder - полусумматор



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

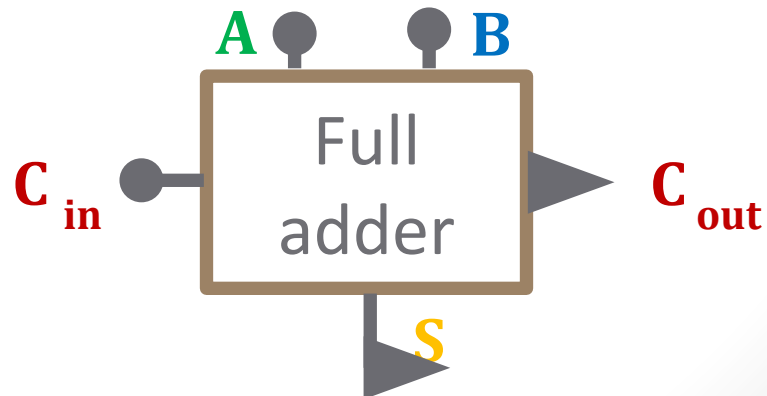
Полусумматор – суммирует два входящих бита, получая бит результата и бит переполнения.

Full adder - сумматор

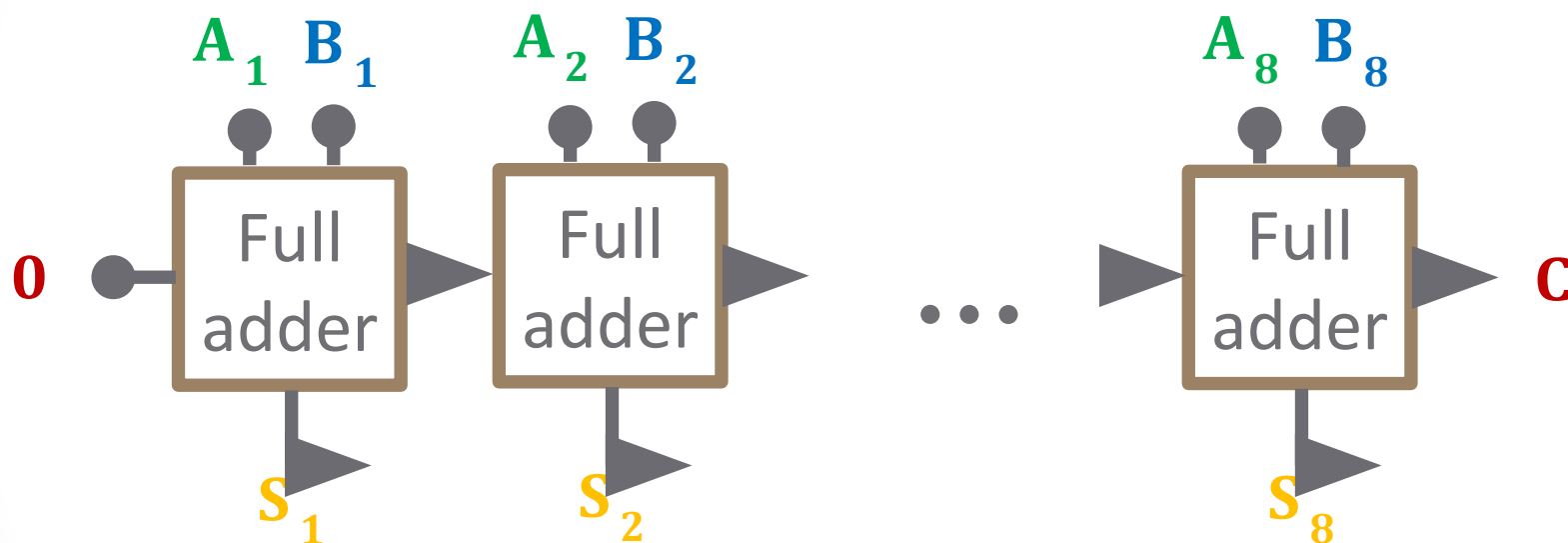


A	B	C _{in}	S	C _{out}
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

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

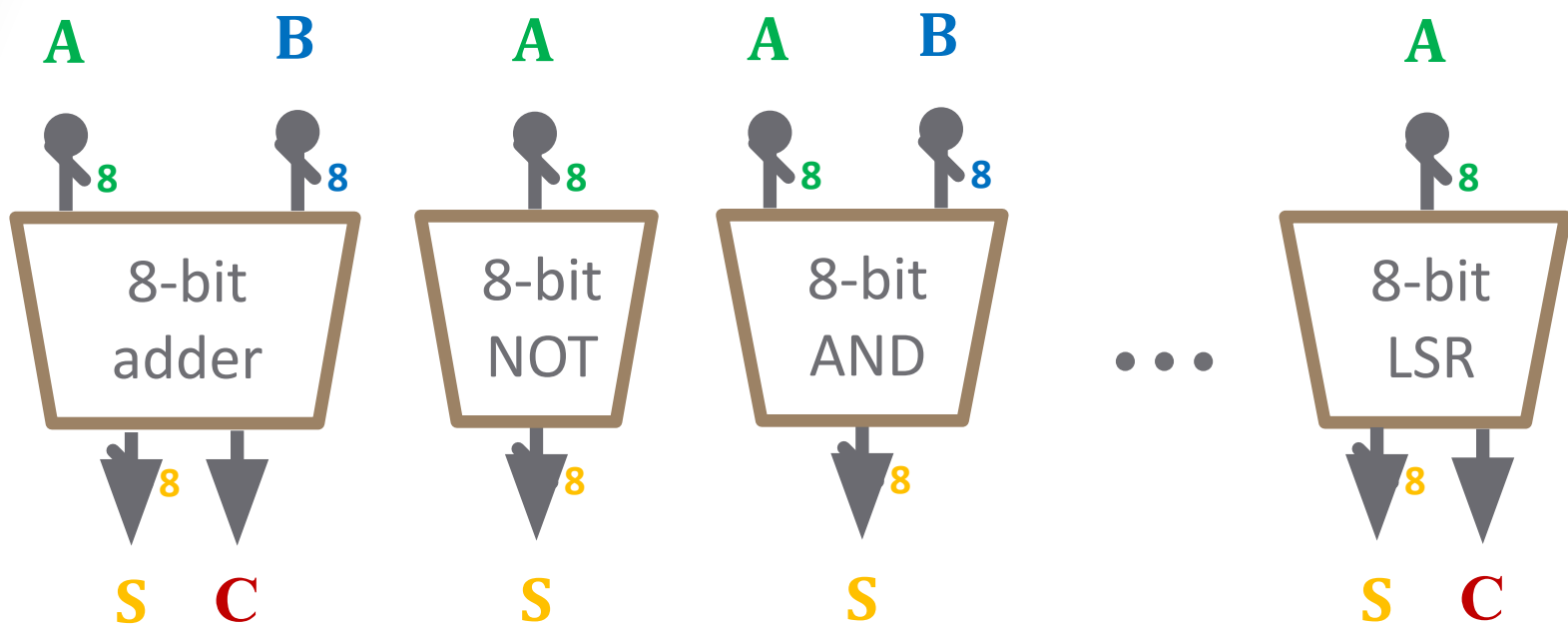


Полный 8 битный сумматор



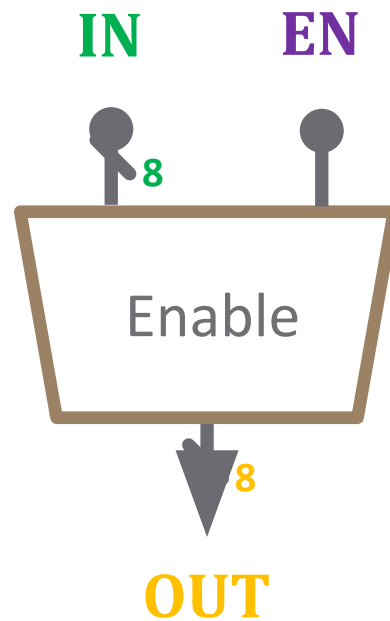
Итого: для создания полного 8 битного сумматора, основанного на базе логических элементов NAND потребуется $2 * ((2 * 5 + 3) * 8) = 208$ транзисторов.

8 битные операции



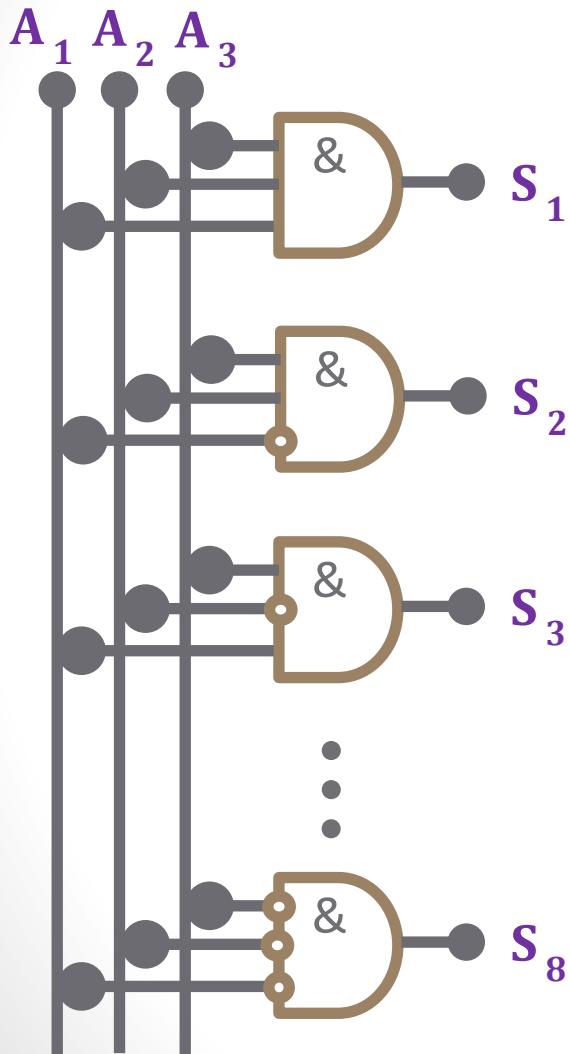
Проводники операндов (8 проводов каждый) можно подсоединить одновременно к блокам всех операций. **Итого:** на 16 входящих проводников, получится по 8 или 9 проводников с каждой операции, которые объединить нельзя (монтажный OR).

Защелка выключатель



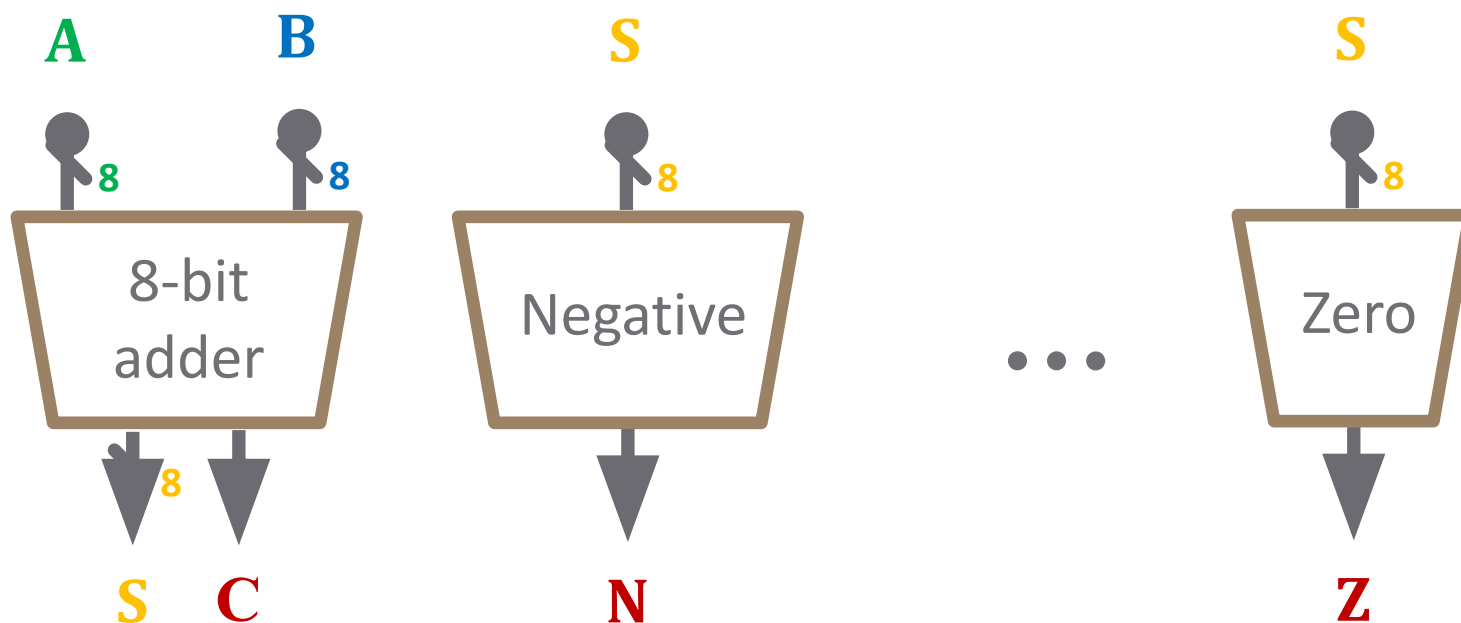
Добавив на выход каждого блока операции по выключателю, мы можем объединить все выходы получив 16 проводников входов и $8 + 1$ выходов, плюс по одному управляющему проводнику на каждую операцию.

Дешифратор



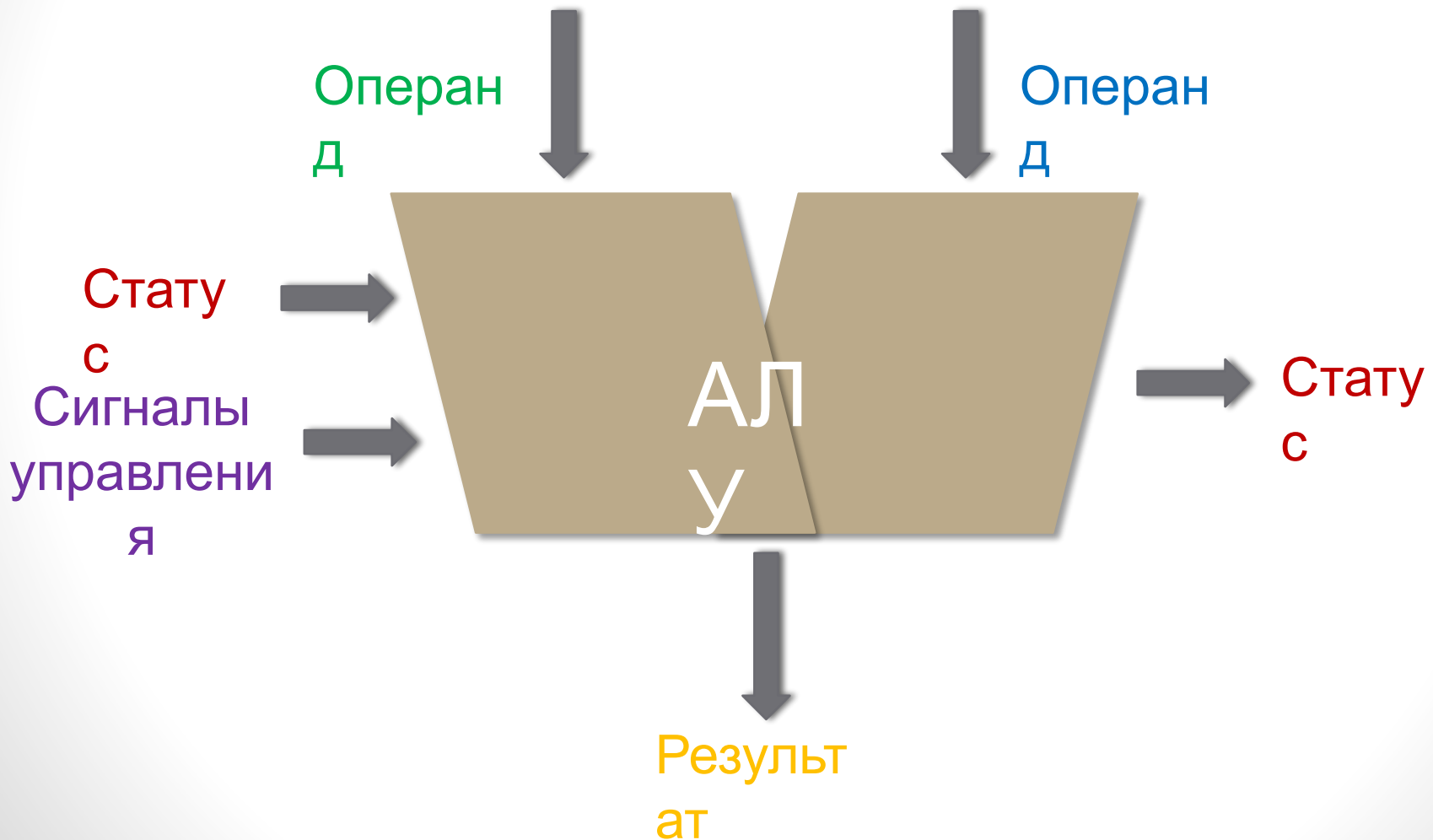
A_3	A_2	A_1	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Статус результата

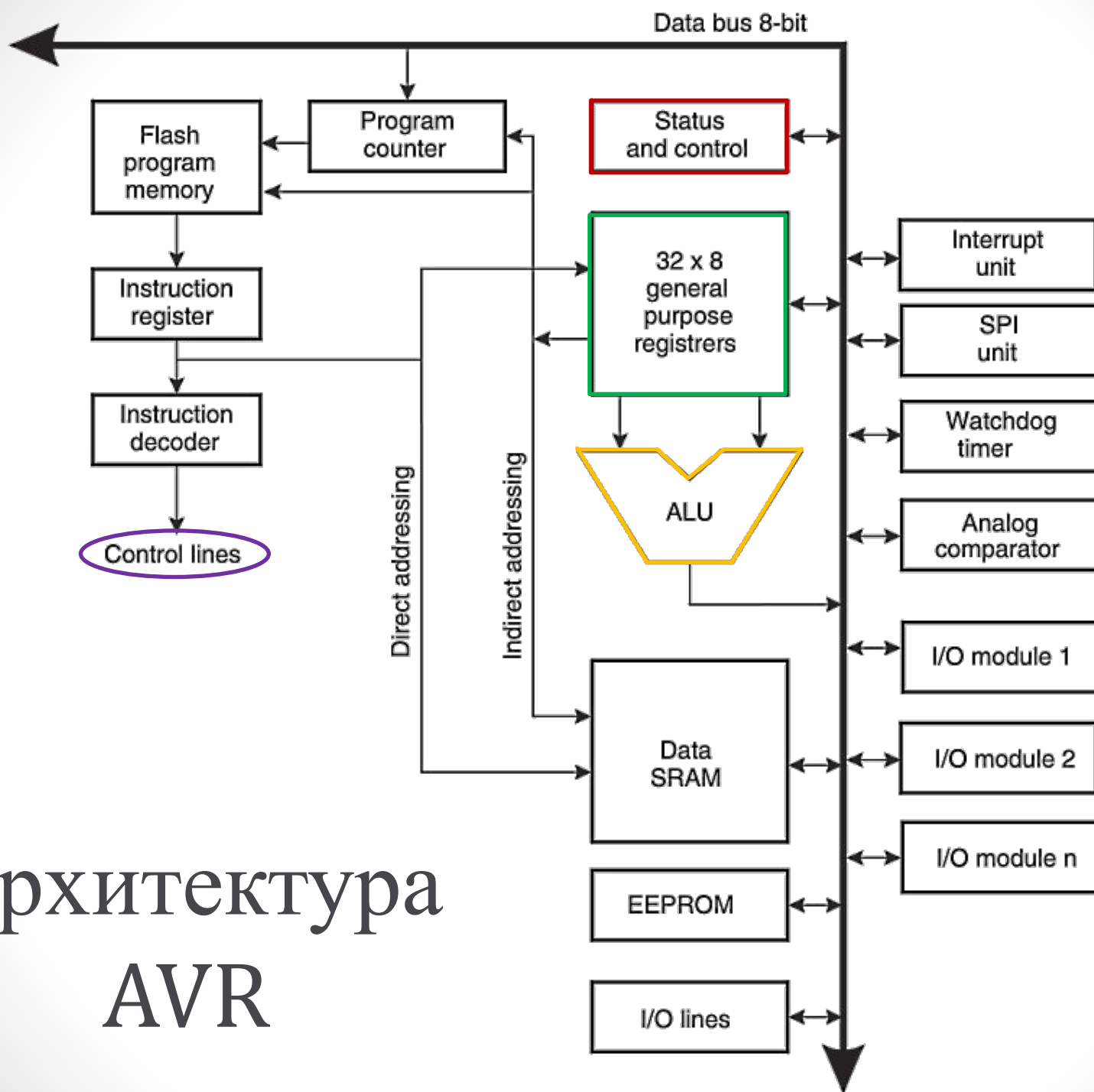


Из результата операции можно сразу же получить полезную информацию например (**C**) переполнение разряда при сложении или особый блок **Zero** который выполняет **XOR** между всеми 8 проводниками результата и если он равен 0 то $Z = 1$.

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 POH 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:

Размер: 2

NOP

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

К

1110	KKKK	dddd	KKKK
------	------	------	------

байта

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.

AVR Studio 4

The screenshot displays the AVR Studio 4 environment. The main window shows assembly code for the CPU, including instructions like `LDI R16, 0x01000000`, `LDI R17, 0x01000000`, and `CALL CALL_SPCALL`. The `CALL_SPCALL` macro is expanded to show `CALL SPCALL` and `CALL SPCALL` instructions. The `CALL_SPCALL` macro is defined as `CALL SPCALL` and `CALL SPCALL`.

The **Processor** window shows the following values:

Name	Value
Program Counter	0x00010E
Stack Pointer	0x045F
X pointer	0x0000
Y pointer	0x0000
Z pointer	0x0000
Cycle Counter	13
Frequency	1,000 MHz
Sleep Watch	13.00 us
SFRs	00000000

The **Memory** window shows a dump of memory addresses and their contents:

Address	Content
00000000	11 00 FF FF FF FF FF FF
00000004	03 00 FF FF FF FF FF FF
00000008	FF FF FF FF FF FF FF FF
0000000C	FF FF FF FF FF FF FF FF
00000010	FF FF FF FF FF FF FF FF
00000014	04 20 0E 0F 0A 01 0A
00000018	10 00 00 00 00 00 00 00
0000001C	78 94 00 0A 02 00 05 E4
00000020	06 BF 00 00 00 00 00 00
00000024	FC 0F 04 0A 06 0F 06 07
00000028	00 70 05 F3 00 95 03 09
0000002C	01 00 0E 0F 0A 01 0A
00000030	09 13 00 05 04 0A 0E 0E
00000034	06 00 00 71 0A 0A 0A 0A
00000038	58 93 42 20 34 27 38 38
0000003C	18 95 31 31 36 95 36 95
00000040	36 95 30 E0 E7 E4 E3 E2
00000044	70 1D 09 16 16 16 16 16
00000048	10 00 17 00 10 00 10 00
0000004C	14 00 13 00 12 00 11 00
00000050	10 00 0F 0F 0F 0F 10 00
00000054	00 00 00 00 00 00 13 00
00000058	08 00 00 00 06 00 18 00

The **Register** window shows the following values:

Register	Value
R00	0x000000
R01	0x000000
R02	0x000000
R03	0x000000
R04	0x000000
R05	0x000000
R06	0x000000
R07	0x000000
R08	0x000000
R09	0x000000
R10	0x000000
R11	0x000000
R12	0x000000
R13	0x000000
R14	0x000000
R15	0x000000
R16	0x01000000
R17	0x01000000
R18	0x000000
R19	0x000000
R20	0x000000
R21	0x000000
R22	0x000000
R23	0x000000
R24	0x000000
R25	0x000000
R26	0x000000
R27	0x000000
R28	0x000000
R29	0x000000
R30	0x000000
R31	0x000000

The **Build** window shows the following output:

```
[.obj] 0x000000 0x00000e 224 0 224 0192 2.74  
[.obj] 0x000000 0x000000 0 0 0 1024 0.04  
[.obj] 0x000000 0x000000 0 0 0 512 0.09
```

<http://www.atmel.com/ru/ru/tools/STUDIOARCHIVE.aspx>

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 – Вычесть 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

К	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.