

Основы профессиональной деятельности

Часть третья (не последняя).

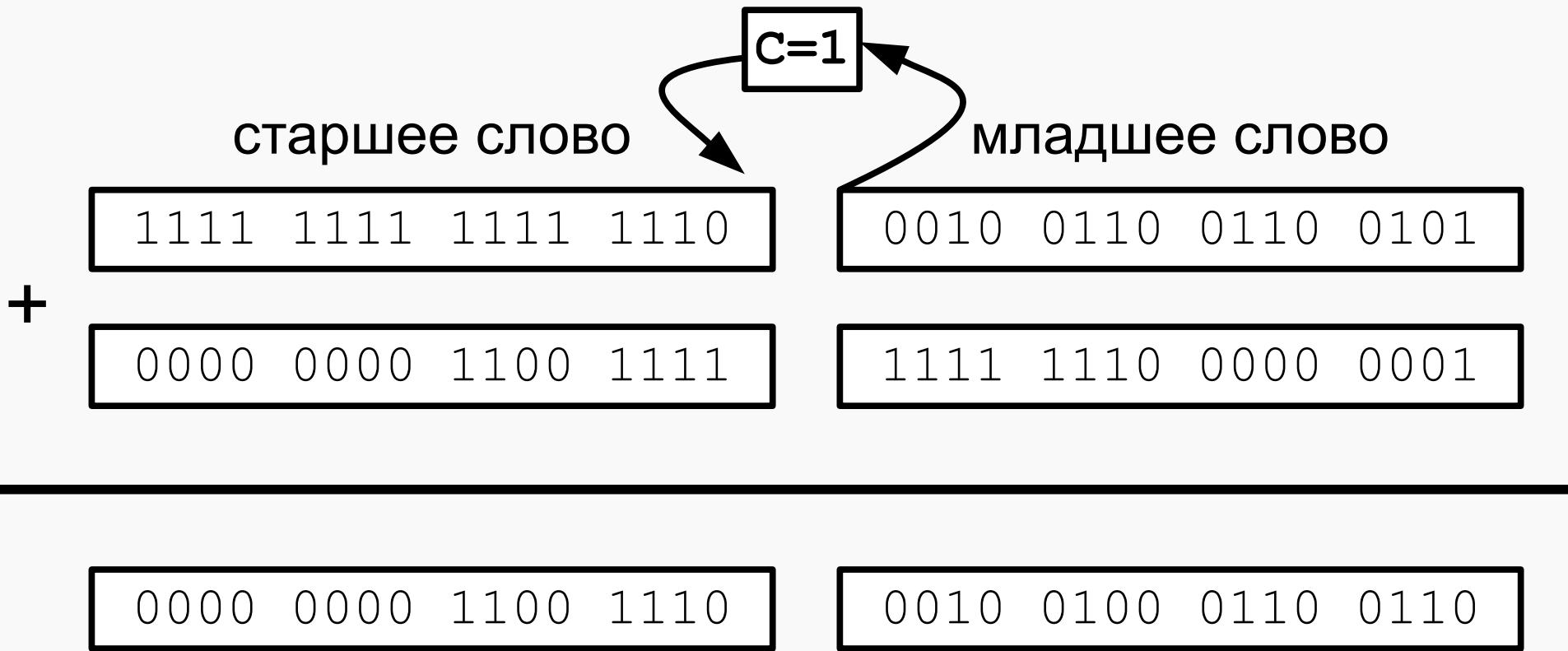
Клименков С.В.
2019-2020 уч. год
v.1.45.07 от 21.03.2022

Выполнение вычислений

1



БЭВМ: 32-х разрядные числа



$$\text{ffffe2665} + \text{cfffe01} = \text{ce2466}$$

$$-121243 + 13630977 = 13509734$$

Суммирование 32-х разрядных чисел

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	A017	LD 0x17	Младшее слово 1-го числа
011	4019	ADD 0x19	Младшее слово 2-го числа
012	E01B	ST 0x1B	Сохранение младшего слова суммы
013	A018	LD 0x18	Старшее слово 1-го числа
014	501A	ADC 0x1A	Старшее слово 2-го числа, перенос между словами
015	E01C	ST 0x1C	Сохранение старшего слова суммы
016	0100	HLT	Останов
017	2665	X	Две ячейки 1-го числа
018	FFFE	X	
019	FE01	Y	Две ячейки 2-го числа
01A	00CF	Y	
01B	2466	R	Две ячейки результата
01C	00CE	R	

$$\text{ffffe2665} + \text{cffe01} = \text{ce2466}$$

Изменение знака 16-ти разрядного числа

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	A015	LD 0x15	X в аккумуляторе
011	0280	NOT	Вычисление дополнения (инверсия битов числа)
012	0700	INC	Инкремент. NEG == NOT+INC
013	E016	ST 0x16	Сохранение результата
014	0100	HLT	
015	0002	X	X
016	FFFE	R	R

$$R = -X$$

Изменение знака 32-х разрядного числа

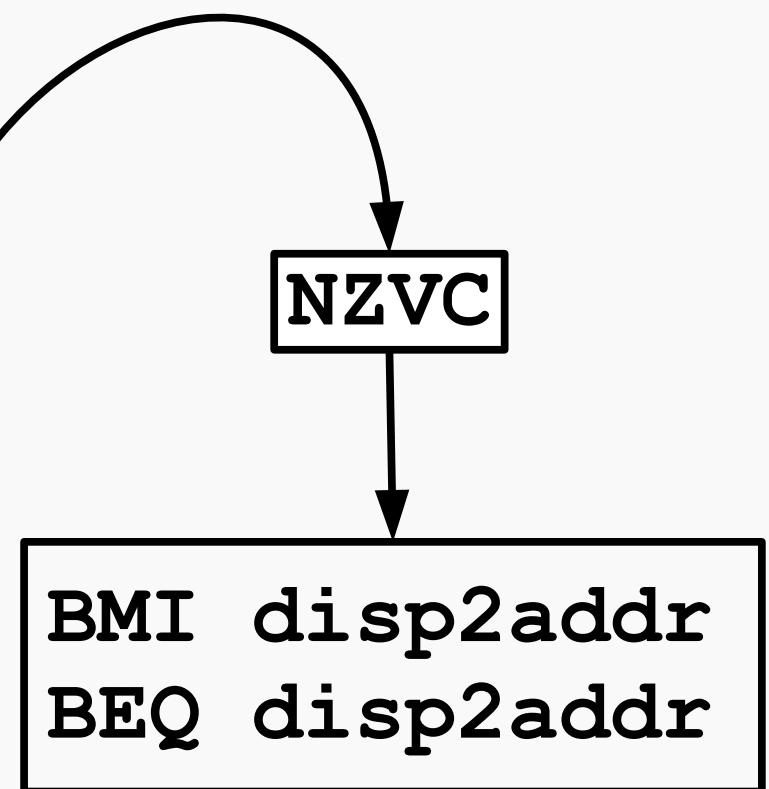
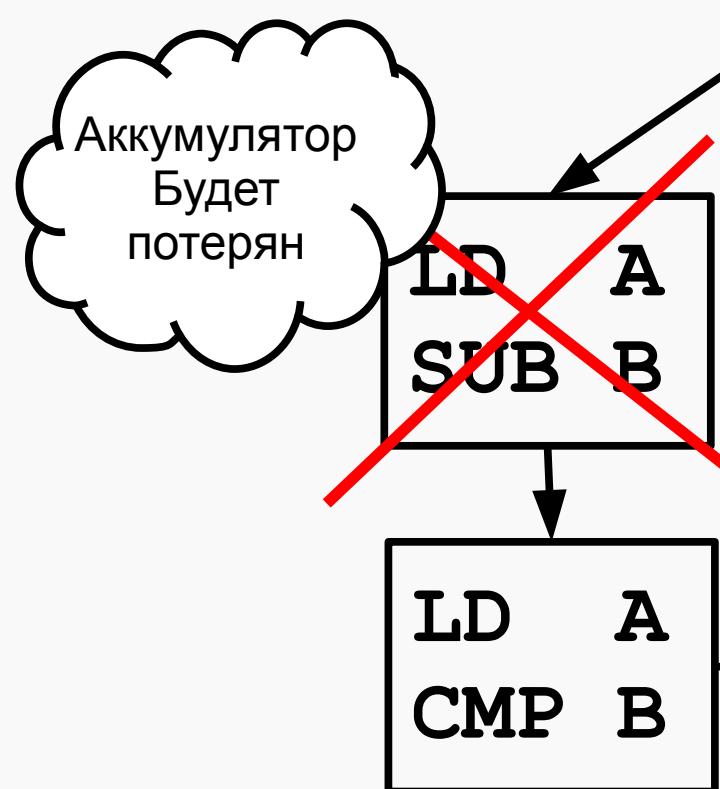
Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	A01B	LD 0x1B	Старшее слово X в аккумуляторе
011	0280	NOT	Вычисление дополнения (инверсия битов числа)
012	E01D	ST 0x1D	Сохранение старшего слова R
013	A01A	LD 0x1A	Младшее слово X в аккумуляторе
014	0780	NEG	Вычисление доп.кода (инверсия битов числа)
015	E01C	ST 0x1C	Сохранение младшего слова
016	0200	CLA	Добавление возможного переноса к старшему слову
017	501D	ADC 0x1D	
018	E01D	ST 0x1D	Сохранение старшего слова
019	0100	HLT	
01A	2665	X	X = -121243
01B	FFFE	X	
01C	D99B	R	R = 121243
01D	0001	R	

$$R = -X$$

Инструкция сравнения CMP: AC + (- operand) → NZVC

- Как сравнить два числа A и B?

$$A \leq B \rightarrow A - B \leq 0$$



Управление вычислительным процессом

Наименование	Мнемон.	Код	Описание
Переход, если равенство	BEQ D	F0XX	IF Z==1 THEN IP+D+1 → IP
Переход, если неравенство	BNE D	F1XX	IF Z==0 THEN IP+D+1 → IP
Переход, если минус	BMI D	F2XX	IF N==1 THEN IP+D+1 → IP
Переход, если плюс	BPL D	F3XX	IF N==0 THEN IP+D+1 → IP
Переход, если выше или равно /перенос	BCS D BHIS D	F4XX	IF C==1 THEN IP+D+1 → IP
Переход, если ниже/нет переноса	BCC D BLO D	F5XX	IF C==0 THEN IP+D+1 → IP
Переход, если переполнение	BVS D	F6XX	IF V==1 THEN IP+D+1 → IP
Переход, если нет переполнения	BVC D	F7XX	IF V==0 THEN IP+D+1 → IP
Переход, если меньше	BLT D	F8XX	IF N⊕V==1 THEN IP+D+1 → IP
Переход, если больше или равно	BGE D	F9XX	IF N⊕V==0 THEN IP+D+1 → IP
Безусловный переход	BR D JUMP D	CEXX	IP+D+1 → IP

Программа вычисления модуля числа

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	A016	LD 0x16	X в аккумуляторе
<u>011</u>	<u>7F00</u>	CMP #0	<u>Если X>=0 то ...</u>
012	F301	BPL IP+1	... переход к адресу 14
013	0780	NEG	Меняем знак
014	E017	ST 0x17	Сохранение результата
015	0100	HLT	
016	FFFE	X	X
017	0002	R	R = X

$$R = |X|$$

Установка флагов

«-» - НЕИЗМ., «*» - по рез., «0»-сб.

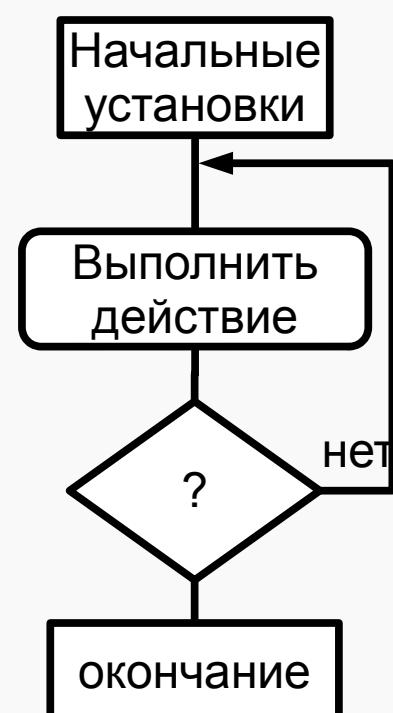
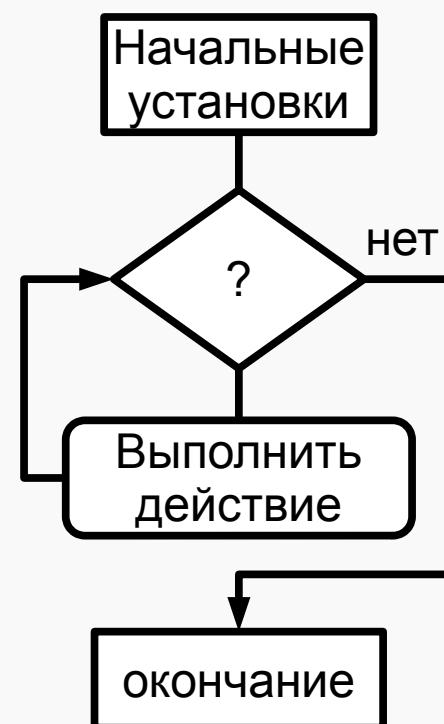
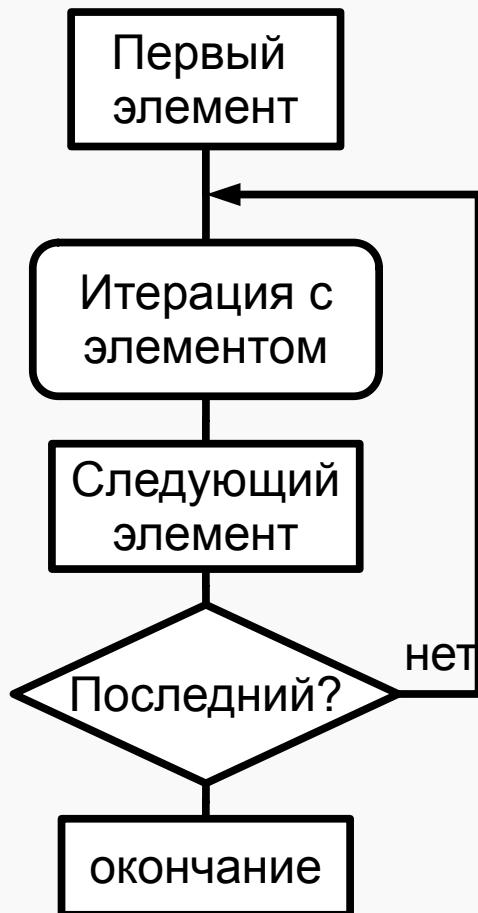
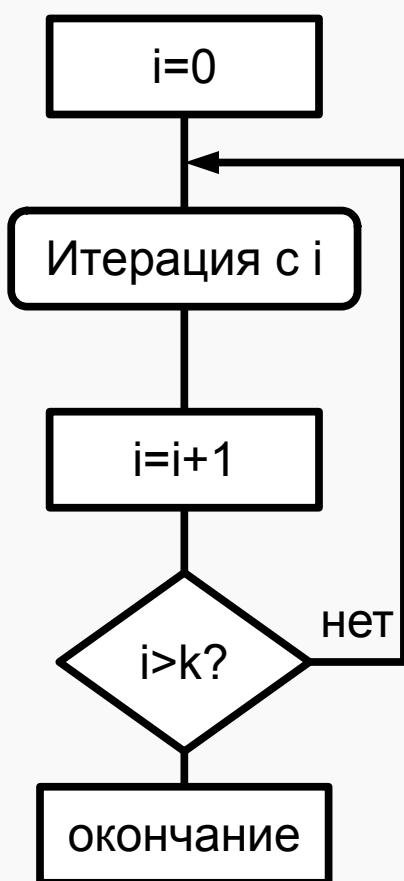
Команда	N	Z	V	C	Операция
NOP	-	-	-	-	Нет операции
CLA	*	*	0	-	$0 \rightarrow AC$
NOT	*	*	0	-	$(^AC) \rightarrow AC$
CLC	-	-	-	0	$0 \rightarrow C$
CMC	-	-	-	*	$(^C) \rightarrow C$
ROL	*	*	*	*	AC и C сдвигается влево. $AC15 \rightarrow C$, $C \rightarrow AC0$
ASR	*	*	*	*	AC сдвигается вправо. $AC0 \rightarrow C$, $AC15 \rightarrow AC14$
SXTB	*	*	0	-	Расширение знака мл. байта $AC7 \rightarrow AC15...AC8$
SWAB	*	*	0	-	Обмен ст. и мл. байта $AC7...AC0 \rightarrow AC15...AC8$
OR M	*	*	0	-	$M AC \rightarrow AC$
ADD M	*	*	*	*	$M + AC \rightarrow AC$
SUB M	*	*	*	*	$AC - M \rightarrow AC$
CMP M	*	*	*	*	Установить флаги по результату $AC - M$
LOOP M	-	-	-	-	$M - 1 \rightarrow M$; Если $M \leq 0$, то $IP + 1 \rightarrow IP$
LD M	*	*	0	-	$M \rightarrow AC$
JUMP M	-	-	-	-	$M \rightarrow IP$

Команды
показаны
не все!

Отступление:

Циклические программы

- for, foreach, while , do-while



R=50Y (вар 1)

ОдЗ!

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	0042	Y	Множимое
011	0000	Z	Ячейка для накопления и хранения результата
012	0032	M	Множитель $50=(32)_{16}$
013	0000	C	Счетчик циклов
014	A011	LD 0x11	К промежуточному результату, находящемуся в ячейке 0x11, добавляется ещё одно значение множимого Y
015	4010	ADD 0x10	
016	E011	ST 0x11	
017	A013	LD 0x13	Содержимое счетчика циклов C увеличивается на 1, а его копия сохраняется в аккумулятор
018	0700	INC	
019	E013	ST 0x13	
01A	7012	CMP 0x12	Если C != M, то продолжаем добавление Y к R
01B	F1F8	BNE IP-8	Переход со смещением на адрес 0x14
01C	0100	HLT	Останов, когда все 50 операций выполнены

R=50Y (LOOP)

Декремент и пропуск	LOOP M	8XXX	M - 1 → M; Если M <= 0, то IP + 1 → IP
---------------------	--------	------	--

Не использует аккумулятор (A) и регистр переноса (C)!

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	0042	Y	Множимое
011	0000	R	Ячейка для накопления и хранения результата
012	0032	M	Множитель 50=(32) ₁₆
013	0200	CLA	К содержимому аккумулятора добавляется Y
014	4010	ADD 0x10	M уменьшается на 1, в случае если M>0 выполняется
015	8012	LOOP 0x12	переход на 14 адрес. Если M становится ==0 (или <0),
016	C014	JMP 0x14	то BR пропускается.
017	E011	ST 0x11	Сохранение результата
018	0100	HLT	

Цикл исполнения LOOP

DR после м.ц. OF содержит значение операнда

AR адрес операнда

- $\sim 0 + DR \rightarrow DR$; $\overline{0x0} = 0xFFFF = -1$; Вычитание единицы из DR
- $\sim 0 + DR \rightarrow BR$, $DR \rightarrow \text{MEM}(AR)$; Записываем значение операнда в память. Вычитание еще единицы (ЗАЧЕМ??!)
- if $BR(15) = 0$ then GOTO INT ; Проверка на положительный (DR-1) и если да, то завершение цикла
- IP + 1 → IP ; Перескок через команду, если BR=DR-1 отрицательное
- GOTO INT ; Завершение цикла

R=50Y

(«китайский», правильный)

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	0042	Y	Множимое
011	0000	R	Ячейка для накопления и хранения результата
012	0000	R'	Промежуточный результат Y*16
013	A010	LD 0x10	Y
014	0500	ASL	Y*2
015	E011	ST 0x11	Сохраним в ячейке R
016	0500	ASL	Y*4
017	0500	ASL	Y*8
018	0500	ASL	Y*16
019	E012	ST 0x12	Сохраним в ячейке R'
01A	0500	ASL	Y*32
01B	4012	ADD 0x12	Добавим к аккумулятору R и R', таким образом, что
01C	4011	ADD 0x11	R=32Y+16Y+2Y=(32+16+2)Y
01D	E011	ST 0x11	Сохраняем результат
01E	0100	HLT	

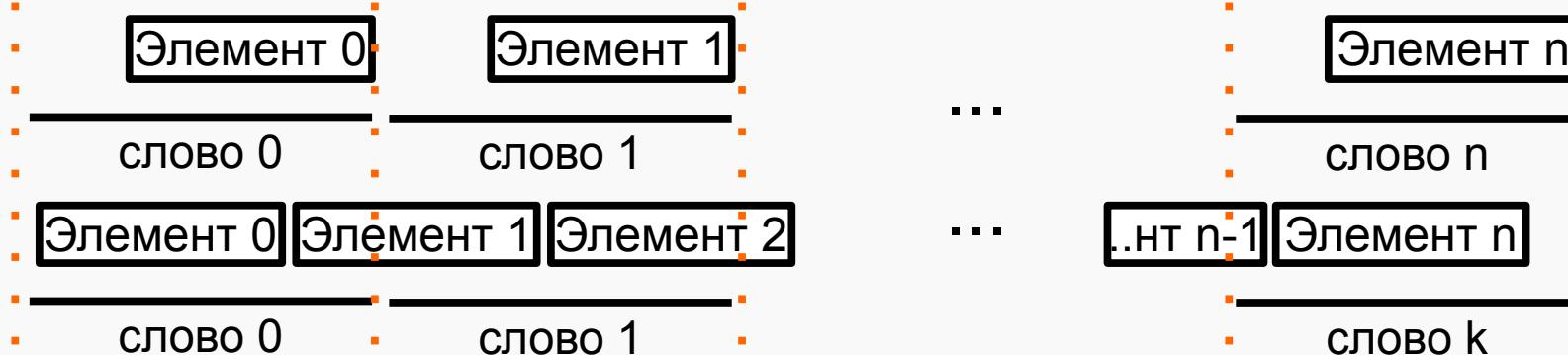
$$50_{10} = 32_{16} = 00110010_2$$

Представление одномерных массивов данных

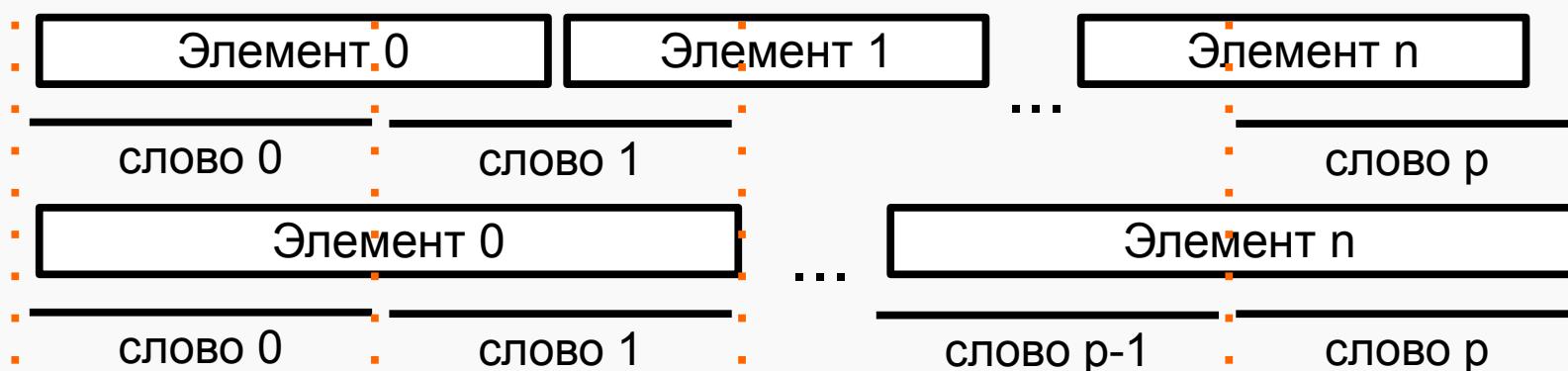
1. Элемент массива занимает ровно слово



2. Элемент меньше слова



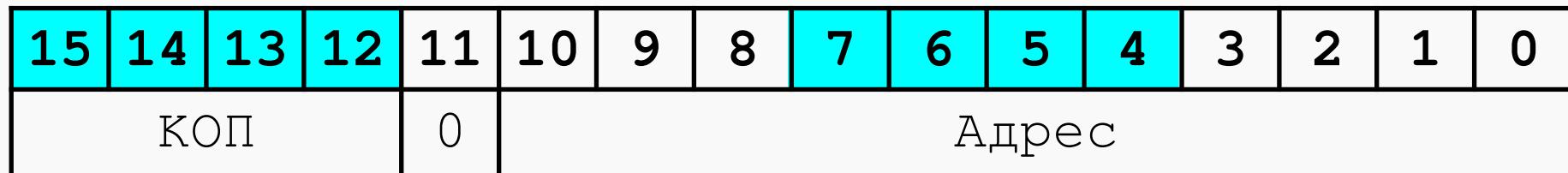
3. Элемент больше слова



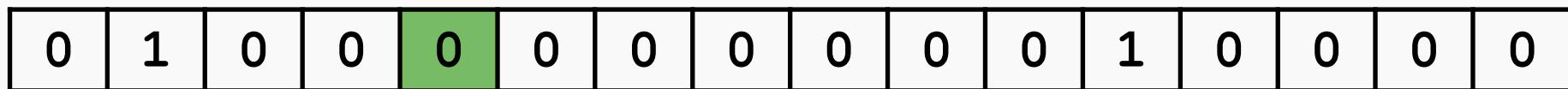
Суммирование элементов массива (переадресация)

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
005	0000	S	Ячейка, отведенная для накопления результата.
006	0020	C	Число элементов массива 32
010 ... 02F			Элементы массива
030 031 032	A005 <u>4010</u> E005	LD 0x5 <u>ADD</u> ? ST 0x5	Предыдущий результат (S) складывается с ячейкой, указанной в коде команды по адресу 31 и записывается в S
033 034 035 036 037 038	A031 0700 E031 8006 CEF8 0100	LD 0x31 <u>INC</u> ST 0x31 <u>LOOP</u> 0x6 BR IP-8 HLT	Взять код команды по адресу 31 Увеличить его на 1 Записать в ячейку 31. Элементы закончились? Нет – переход на 30 адрес Да - останов

Режимы адресации: Прямая абсолютная, режим 0



ADD 0x10



- Адрес полностью кодируется в младших 11 битах
- Непосредственно загружается в AR из кода команды

Режимы адресации: прямая относительная, 1-110



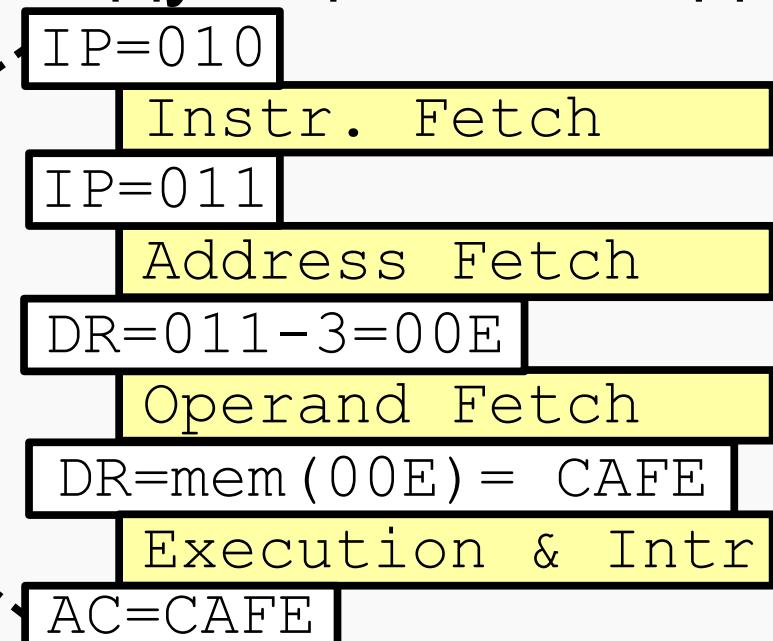
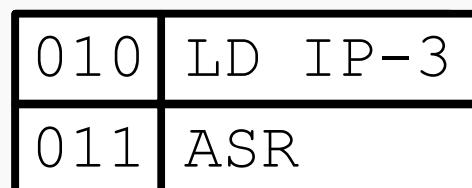
LD IP-3 (код 0xAEFD)



- В битах 0-7 закодировано смещение относительно адреса следующей команды



...

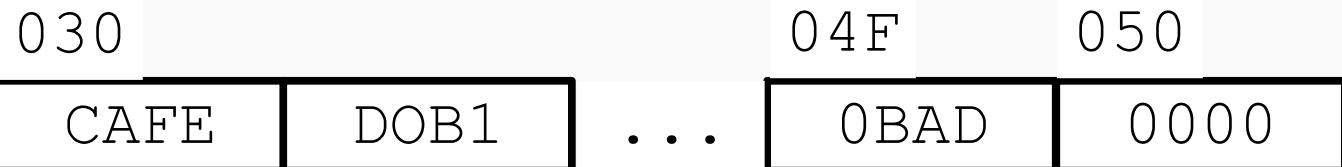


Косвенная адресация: Режимы: 1-000, 1-010, 1-011

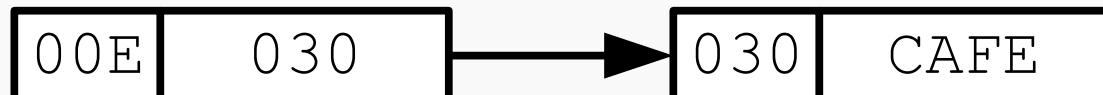
Массив:

Адрес начала = 0x30

Длина = 0x20

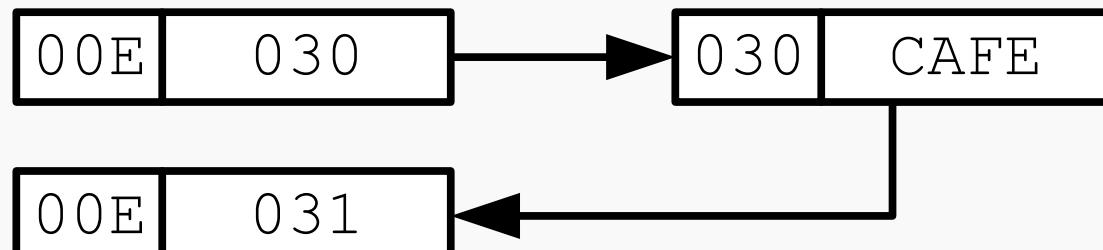


1. Косвенная относительная



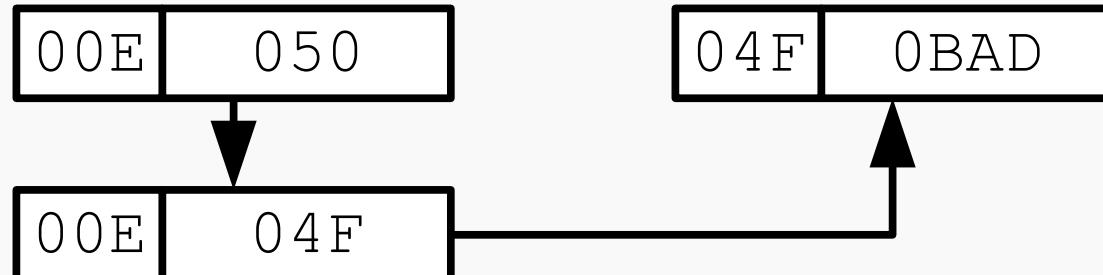
010 | LD (IP-3)

2. Косвенная (относительная) автоинкрементная



010 | LD (IP-3) +

3. Косвенная (...) автодекрементная



010 | LD - (IP-3)

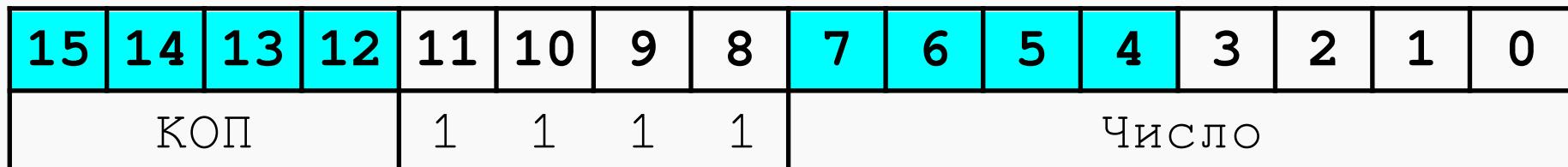
Ассемблер:
 i: WORD 0x30
 ...
 LD (i)+

Режимы адресации:

Со смещением относительно SP

- Будет рассмотрена в дальнейшем
- Но там никакой магии — все тоже самое, что и в относительной, только базовый регистр не IP а SP
- Нужна для выборки параметров со стека

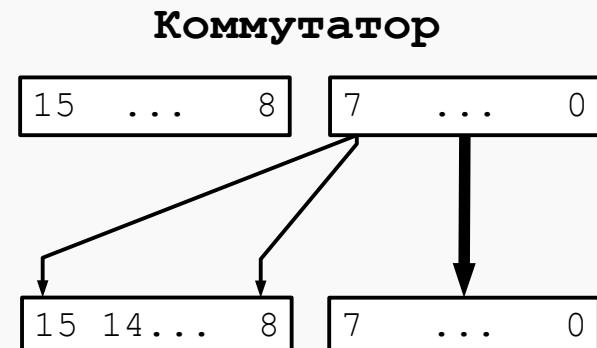
Режимы адресации: Прямая загрузка, режим 111



LD #0x80 (LD #-128, код 0xAF80)



- Значение в битах 0-7 непосредственно загружается в аккумулятор
- Происходит **расширение знака**: бит 7 АС копируется в биты 8-15 АС
(нужно для, например, CMP #-5)



Режимы адресации: Сводная таблица

Код				Мнемоника	Описание	Реализация машинных циклов AF, OF
11	10	9	8			
0	М	М	М	ADD 0ADDR ADD \$L	Прямая абсолютная	AF- нет! $DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	0	0	0	ADD (L)	Косвенная относительная	SXT CR(0..7) → BR, BR + IP → AR, MEM(AR) → DR, $DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	0	0	1		Резерв	
1	0	1	0	ADD (L) +	Косвенная автоинкрементная (постинкремент)	SXT CR(0..7) → BR, BR + IP → AR, MEM(AR) → DR, DR + 1 → DR, DR → MEM(AR), DR - 1 → DR, $DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	0	1	1	ADD - (L)	Косвенная автодекрементная (предекремент)	SXT CR(0..7) → BR, BR + IP → AR, MEM(AR) → DR, DR - 1 → DR, DR → MEM(AR), $DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	1	0	0	ADD &N ADD (SP+N)	Косвенная относительная, со смещением (SP)	SXT CR(0..7) → BR, BR + SP → DR, $DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	1	0	1		Резерв	
1	1	1	0	ADD L ADD (IP+N)	Прямая относительная	SXT CR(0..7) → BR, BR + IP → DR, $DR \rightarrow AR; MEM(AR) \rightarrow DR$
1	1	1	1	ADD #N	Прямая загрузка	SXT CR(0..7) → BR, BR → DR

Где здесь
AF и OF?

Суммирование, адресация косвенная автоинкрементная

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
005	0000	S	Ячейка, отведенная для накопления результата.
006	0010	C	Длина массива (0x10)
007	0030	I	Текущий элемент массива
010	0200	CLA	Очистить результат
011	4AF5	ADD (IP-B) +	$0x12-0xB = 7$; AC + следующий элемент массива
012	8006	LOOP 0x6	Мы сложили все элементы?
013	CEFD	BR IP-3	$0x14-3 = 0x11$; Нет – продолжим складывать
014	E005	ST 0x5	Результат в ячейке 5
015	0100	HLT	
030 ... 04F			Элементы массива

Суммирование (косвенная автодекрементная адресация)

Что нужно изменить по
сравнению с предыдущей
программой?



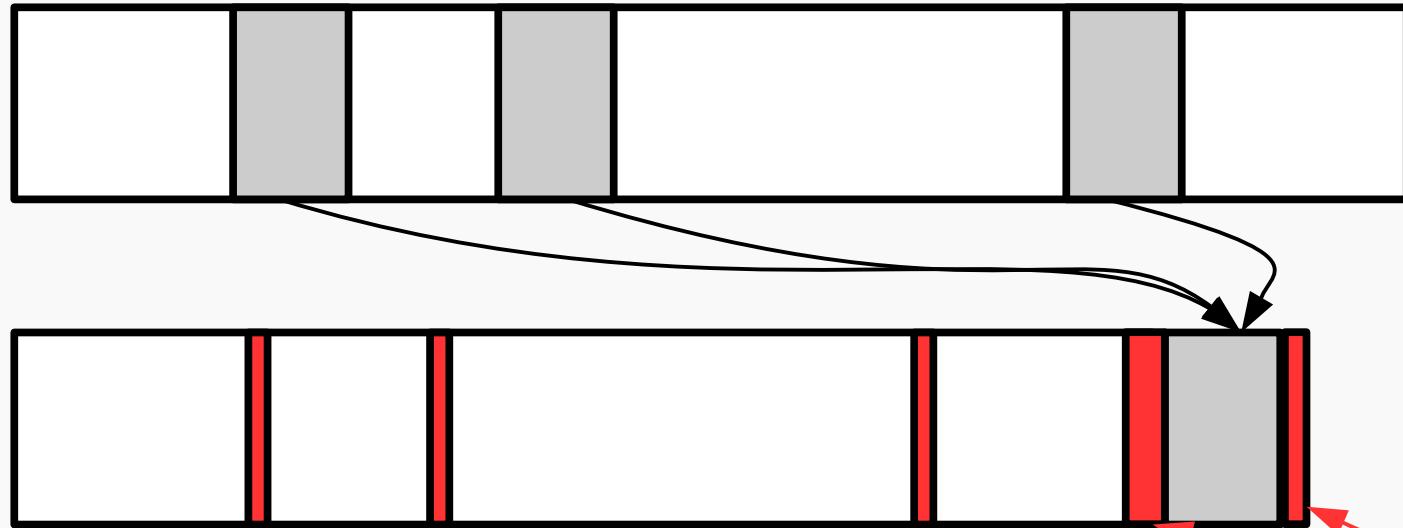
Подпрограммы

2



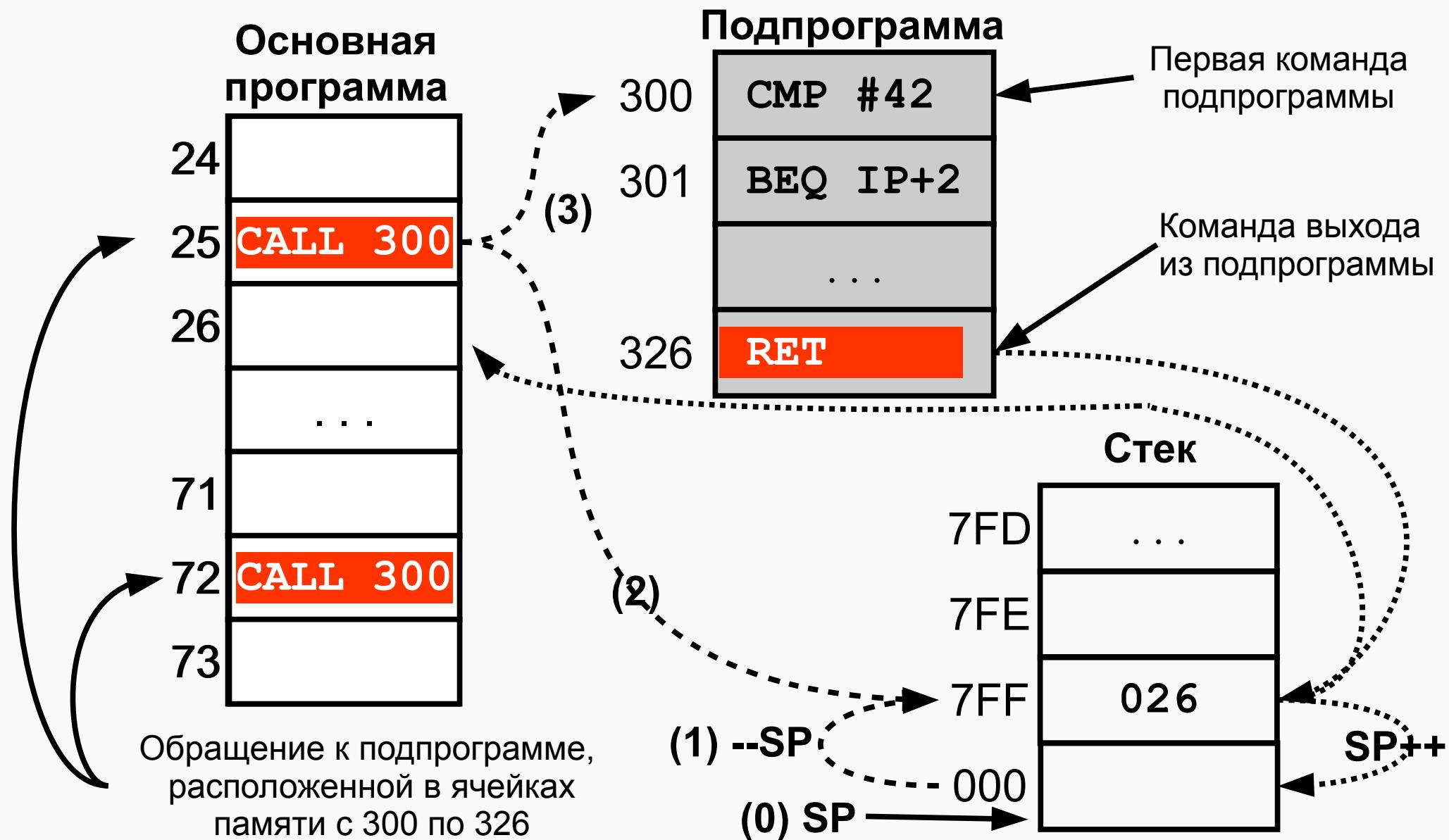
Мотивация

- Зачем?



- Когда выгодно создавать подпрограмму?
 - Размер кода
 - Передача и обработка параметров; возвращение результатов
- Инлайнинг — процесс, обратный выделению кода в подпрограмму

БЭВМ: Вызов программы и возврат из нее



Передача параметров и получение результатов

- Аккумулятор (Регистры Общего Назначения)
 - Сколько параметров можно передать в БЭВМ?
- Адресуемые ячейки памяти
 - Необходимо организовать
- Стек
- Регистровые окна

Комплекс программ: передача параметров через аккумулятор

Комплекс программ для вычисления функции $R=4|X|+1$ для чисел в ячейках 58,63,71 с размещением результатов в ячейках 74,77,82

Основная программа

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	A058	LD 0x58	Первое число в 58
011	D030	CALL 0x30	Вызов п/п
012	E074	ST 0x74	Результат в 74
013	A063	LD 0x63	Второе число в 63
014	D030	CALL 0x30	Вызов п/п
015	E077	ST 0x77	Результат в 77
016	A071	LD 0x71	Третье число в 71
017	D030	CALL 0x30	Вызов п/п
018	E082	ST 0x82	Результат в 82
019	0100	HLT	

Подпрограмма вычисления $R=4|X|+1$

Адрес	Содержимое	
	Код	Мнемоника
030	F301	BPL IP+1
031	0780	NEG
032	0500	ASL
033	0500	ASL
034	0700	INC
035	0A00	RET

Комплекс программ: передача через адреса ячеек памяти

Подпрограмма
вычисления
 $R=4|X|+1$

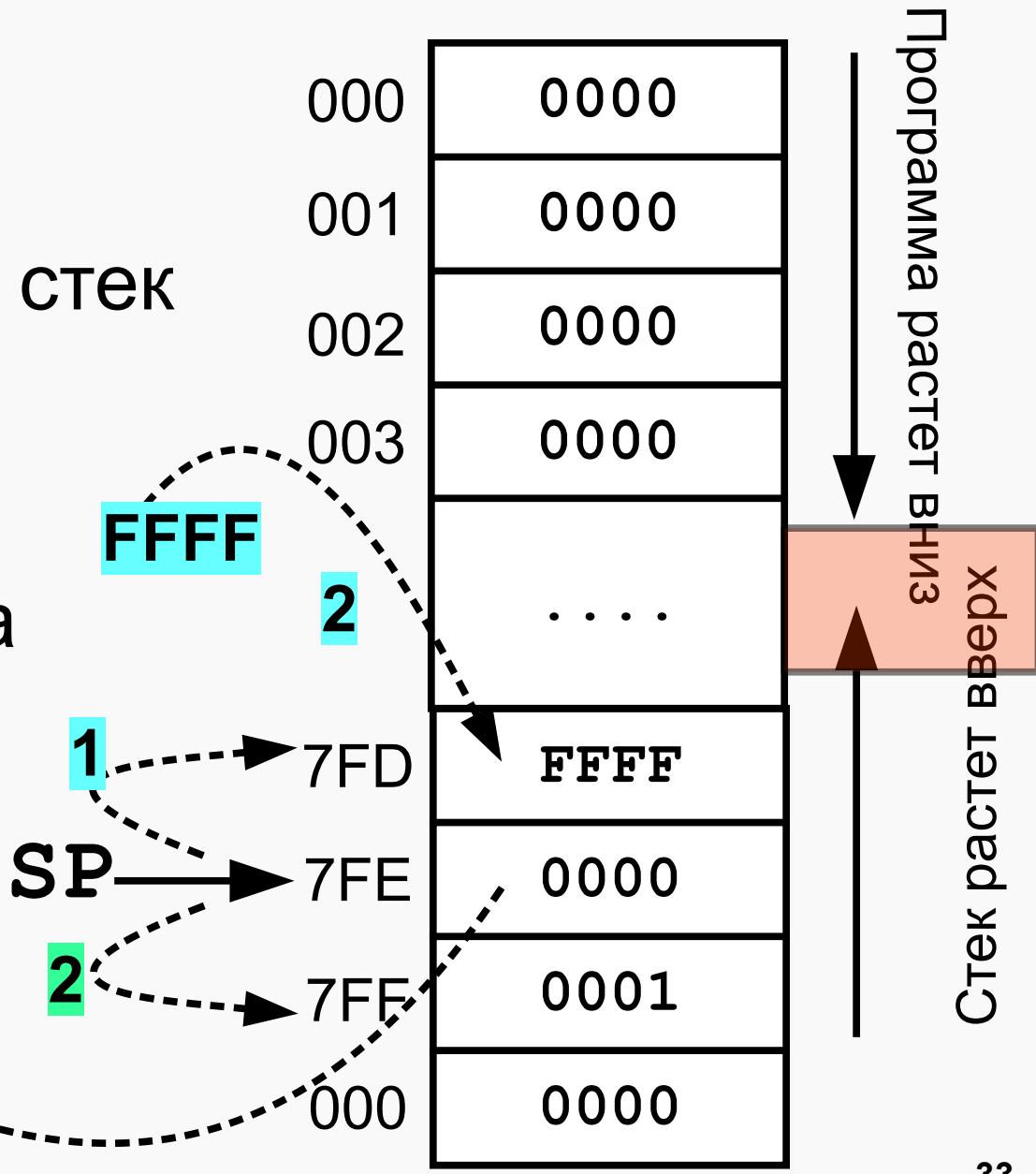
Адрес	Содержимое		Комментарии
	Код	Мнемоника	
030	AC00	LD (SP+0)	Загрузить вершину стека в ячейку 042
031	EE10	ST MSP	
032	AA0F	LD (MSP) +	Адрес X в 043; Сохр. SP=SP+1
033	EE0F	ST ADDR	
034	A80E	LD (ADDR)	
035	F301	BPL IP+1	
036	0780	NEG	
037	0500	ASL	
038	0500	ASL	
039	0700	INC	
03A	0C00	PUSH	Результат временно в стек
03B	AA06	LD (MSP) +	Адрес R в 043
03C	EE06	ST ADDR	Сохр. SP=SP+1
03D	0800	POP	Результат из стека
03E	E804	ST (ADDR)	Сохранить в памяти
03F	AE02	LD MSP	
040	EC00	ST (SP+0)	Взять увелич. SP и сохр. на вершине
041	0A00	RET	Возврат.
042	0000	;MSP	Ячейка для ук. стека
043	0000	;ADDR	Ячейка для адресов

Основная программа

Адрес	Содержимое	
	Код	Мнемоника
010	D030	CALL 0x30
011	0058	Адрес X1
012	0074	Адрес R1
013	D030	CALL 0x30
014	0063	Адрес X2
015	0077	Адрес R2
016	D030	CALL 0x30
017	0071	Адрес X3
018	0082	Адрес R3
019	0100	HLT

Стек

- SP — stack pointer
(указатель стека)
- **PUSH** — положить на стек
 1. $--SP$
 2. Записать по (SP)
- **POP** — снять со стека
 1. Прочитать по (SP)
 2. $SP++$
- Взять i -й элемент
 1. $(SP+/-i)$



Передача параметров с использованием стека $R=2X+Y$

Основная программа

Адрес	Содержимое	
	Код	Мнемоника
010	A019	LD 0x19 ;X
011	0C00	PUSH
012	A01A	LD 0x1A ;Y
013	0C00	PUSH
014	D030	CALL 0x30
015	0800	POP
016	0800	POP
017	E01B	ST 0x1B ;R
018	0100	HLT
019	0030	X
01A	0005	Y
01B	0065	R

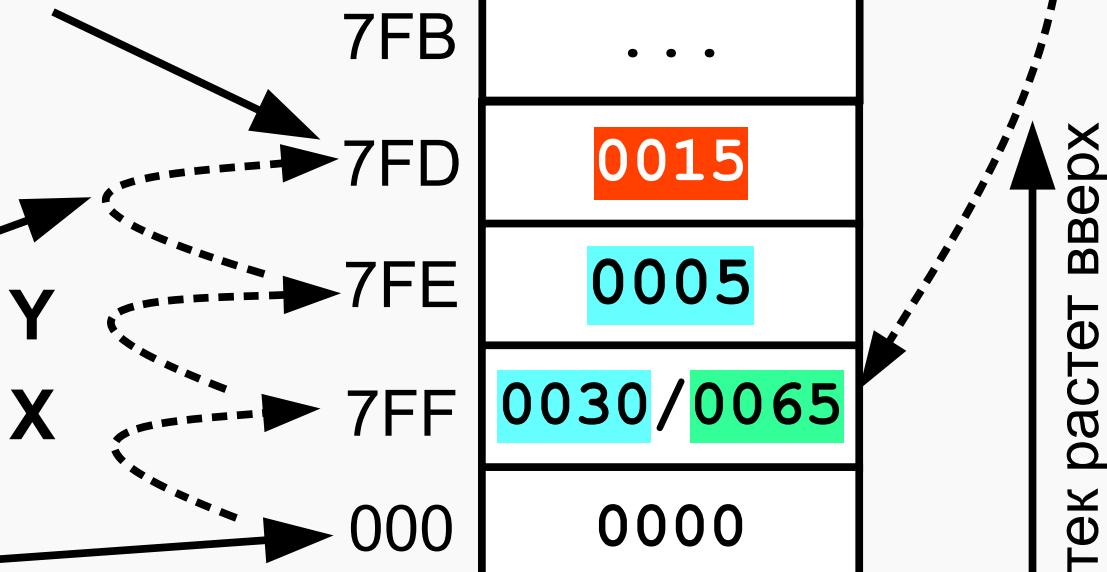
Подпрограмма $R=2X+Y$

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
030	AC02	LD &2	X из 2 эл-та
031	0500	ASL	X*2
032	4C01	ADD &1	X*2+Y из 1 эл-та
033	EC02	ST &2	R в 2-й элемент
034	0A00	RET	возврат

SP внутри п/п

Адрес
возврата

SP до вызова
подпрограммы



Цикл исполнения CALL

CALL: DR после м.ц. OF содержит адрес перехода

- **DR** → **BR** ; Адрес перехода записать в BR
- **IP** → **DR** ; Подготовить адрес возврата для записи в стек
- **BR** → **IP** ; Переход на подпрограмму
- $\sim 0 + SP \rightarrow SP, AR$; Уменьшить стек на 1
- **DR** → **MEM(AR)** ; Записать адрес возврата
- **GOTO INT** ; Завершение цикла

Цикл исполнения RET

- **SP → AR** ; Вершину стека поместить в AR
- **MEM (AR) → DR** ; Прочитать адрес возврата
- **DR → IP** ; Вернуться из подпрограммы
- **SP + 1 → SP** ; Увеличить стек на 1
- **GOTO INT** ; Завершение цикла

Рекурсивность сумма чисел от 1 до N

- Рекурсивность — способность подпрограммы вызывать саму себя.

Стек	
7F9	...
7FA	0036
7FB	0001
7FC	0036
7FD	0002
7FE	0013
7FF	0003
000	0000

Основная
программа

Подпрограмма

Адрес	Содержимое	
	Код	Мнемоника
010	AF03	LD #3
011	0C00	PUSH
012	D030	CALL 0x30
013	0800	POP
014	E016	ST 0x16
015	0100	HLT

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
030	AC01	LD &1	Аргумент в АС
031	7F01	CMP #1	== 1 ?
032	F006	BEQ IP+6	На выход!
033	0740	DEC	Уменьшить
034	0C00	PUSH	Аргумент на 1
035	DEFA	CALL IP-6	Вызвать себя
036	0800	POP	Сумма в АС
037	4C01	ADD &1	Добавить себя
038	EC01	ST &1	Сохр. Сумму
039	0A00	RET	

ОДЗ?

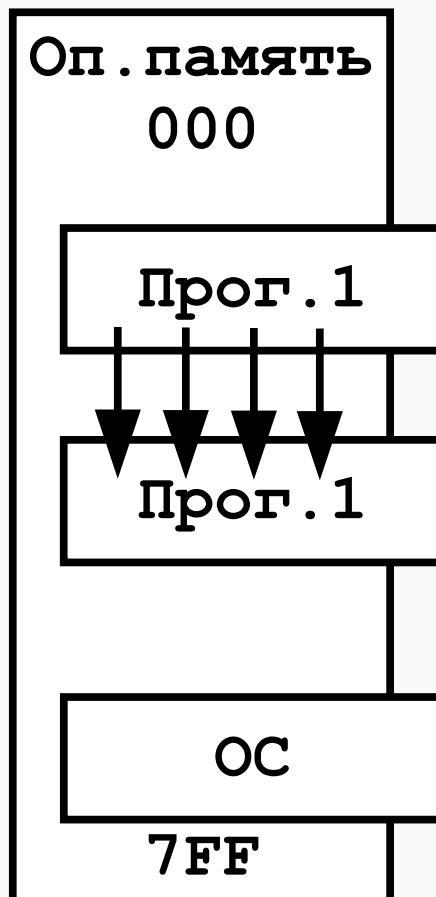
Реентерабельность R=50Y

- Реентерабельность — способность программы быть запущенной несколько раз

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
005	0042	Y	Множимое
006	0000	R	Ячейка для накопления и хранения результата
007	0000	M	Текущее (цикловое) значение множителя
008	0032	Мнач	Начальное количество умножений
00C	0200	CLA	Очистка ячейки для накопления результата
00D	E006	ST 0x6	
00E	A008	LD 0x8	
00F	E007	ST 0x7	
010	0200	CLA	К содержимому аккумулятора добавляется Y
011	4005	ADD 0x5	
012	8007	LOOP 0x7	
013	CEFD	BR IP-3	
014	3006	ST 0x06	
015	F000	HLT	

PIC - Position Independent Code (перемещаемый код)

- Код, который работает относительно того адреса на который загружен
 - Необходим для, например, модулей ядра (даже при наличии виртуальной памяти!)
 - Внутри программы только относительные адреса (смещения)
 - Внешние ссылки только абсолютные
 - В БЭВМ есть оба вида адресации!



Длина перемещаемой программы в БЭВМ?

Загрузчик и динамический линковщик программ

- Любая ОС имеет соответствующую программу или часть ядра
 - Загрузка по выбранному ОС адресу (даже в виртуальной памяти)
 - Изменение константных частей адресов в программе
 - Загрузка базовых значений регистров
 - Динамическая загрузка разделяемых библиотек
 - Связывание адресов основной программы с вызываемыми библиотеками

Библиотеки

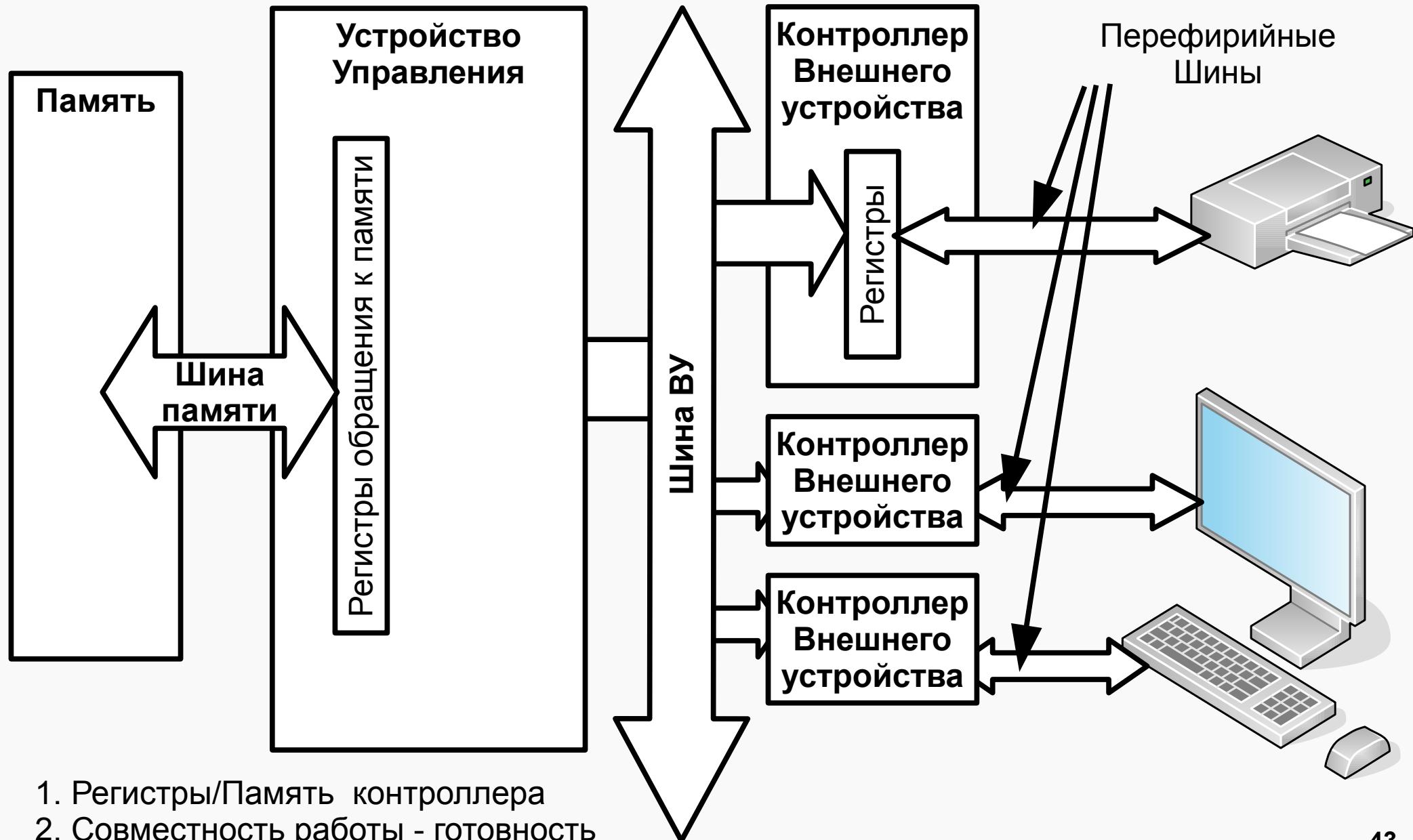
- Набор стандартных библиотечных функций
- Разделяемые (динамически линкуемые) и архивные (статически линкуемые)
 - `# find /lib /usr/lib -name \"*.so\" | wc -l`
3510
 - Статические связывают вызовы функций с телом функции в процессе компиляции
 - Динамические — в момент загрузки
- Если вам нужна функция — см. в библиотеки

Ввод-вывод

3



Подключение устройств



1. Регистры/Память контроллера
2. Совместность работы - готовность

Драйверы

- Организуют совместную работу с устройством
- «Знают» о принципах работы устройства, адресах регистров, поддерживаемых режимах работы
- Управляются единообразным программным интерфейсом

Ввод-вывод

Программно-управляемый

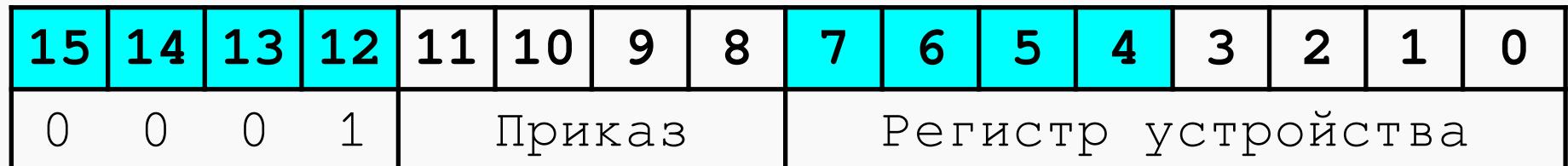
- Инициация обмена
 - Синхронная
 - Асинхронная
 - Управляемая прерываниями
- Передача данных
 - Синхронная/Асинхронная
- Завершение обмена и получение драйвером (программой) результата обмена
 - Синхронное/асинхронное

Управляемый-аппаратурой
ПДП (DMA)



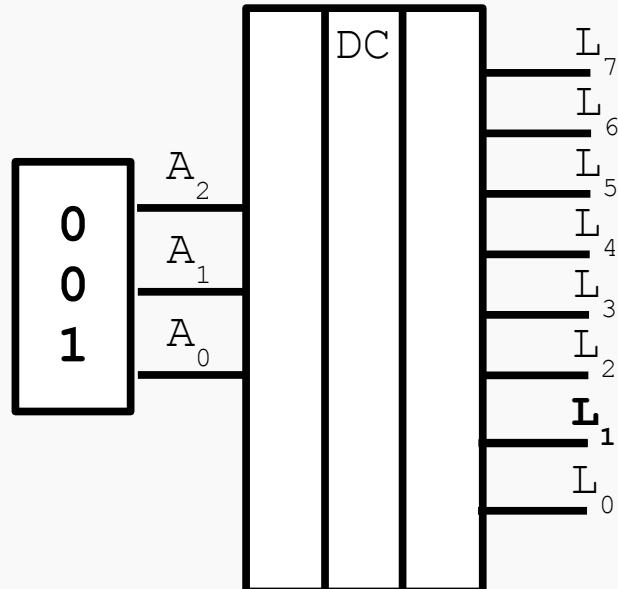
БЭВМ: Команды, связанные с вводом-выводом

Команда ввода-вывода



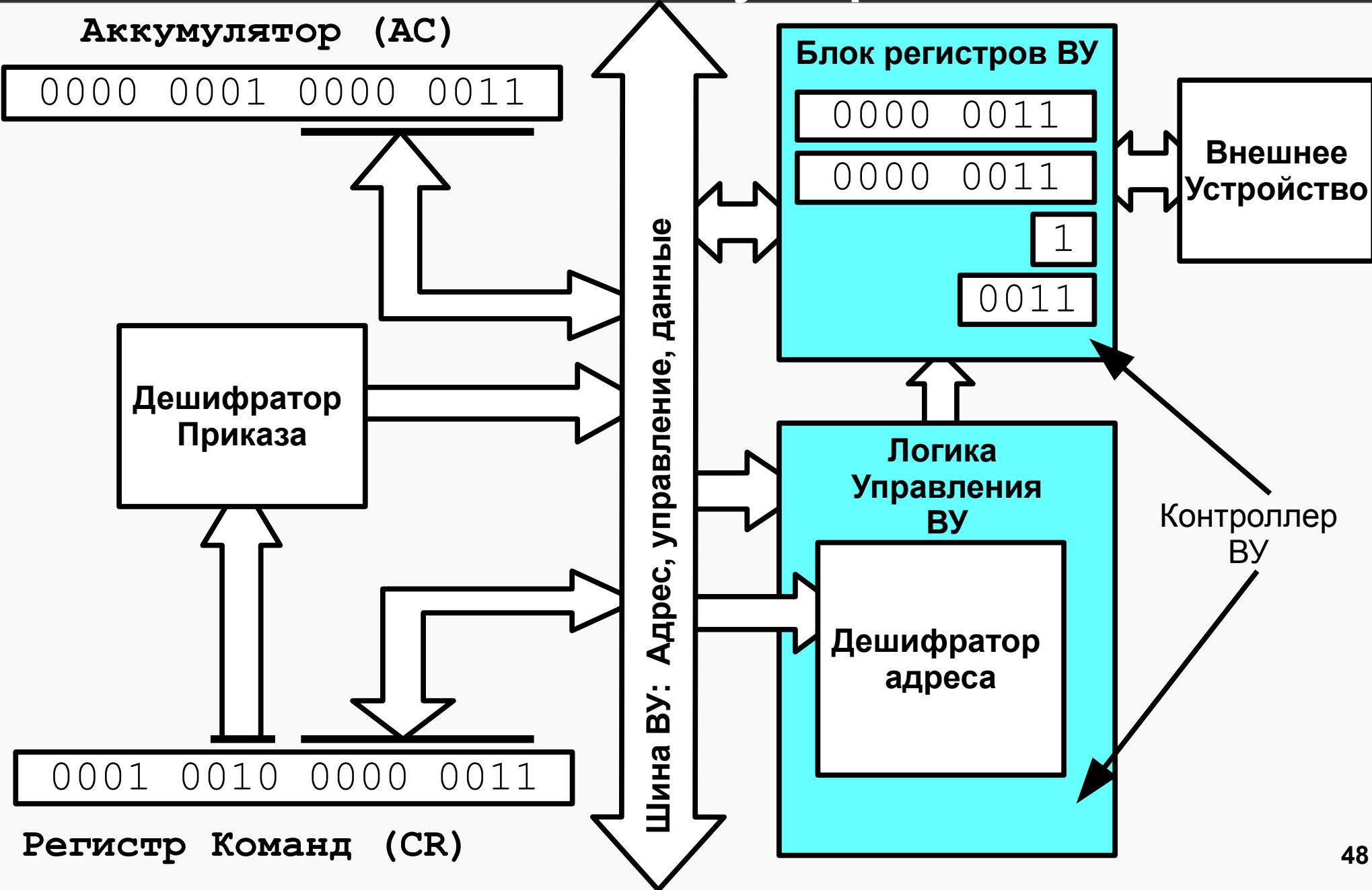
Наименование	Мнемон.	Код	Описание
Запрет прерываний	DI	1000	
Разрешение прерываний	EI	1100	
Ввод	IN REG	12XX	REG → мл. байт АС
Вывод	OUT REG	13XX	мл. байт АС → REG
Прерывание	INT NUM	18XX	Програмное прерывание с вектором NUM
Возврат из прерывания	IRET	0B00	(SP)+ → PS, (SP)+ → IP

Отступление: Дешифратор

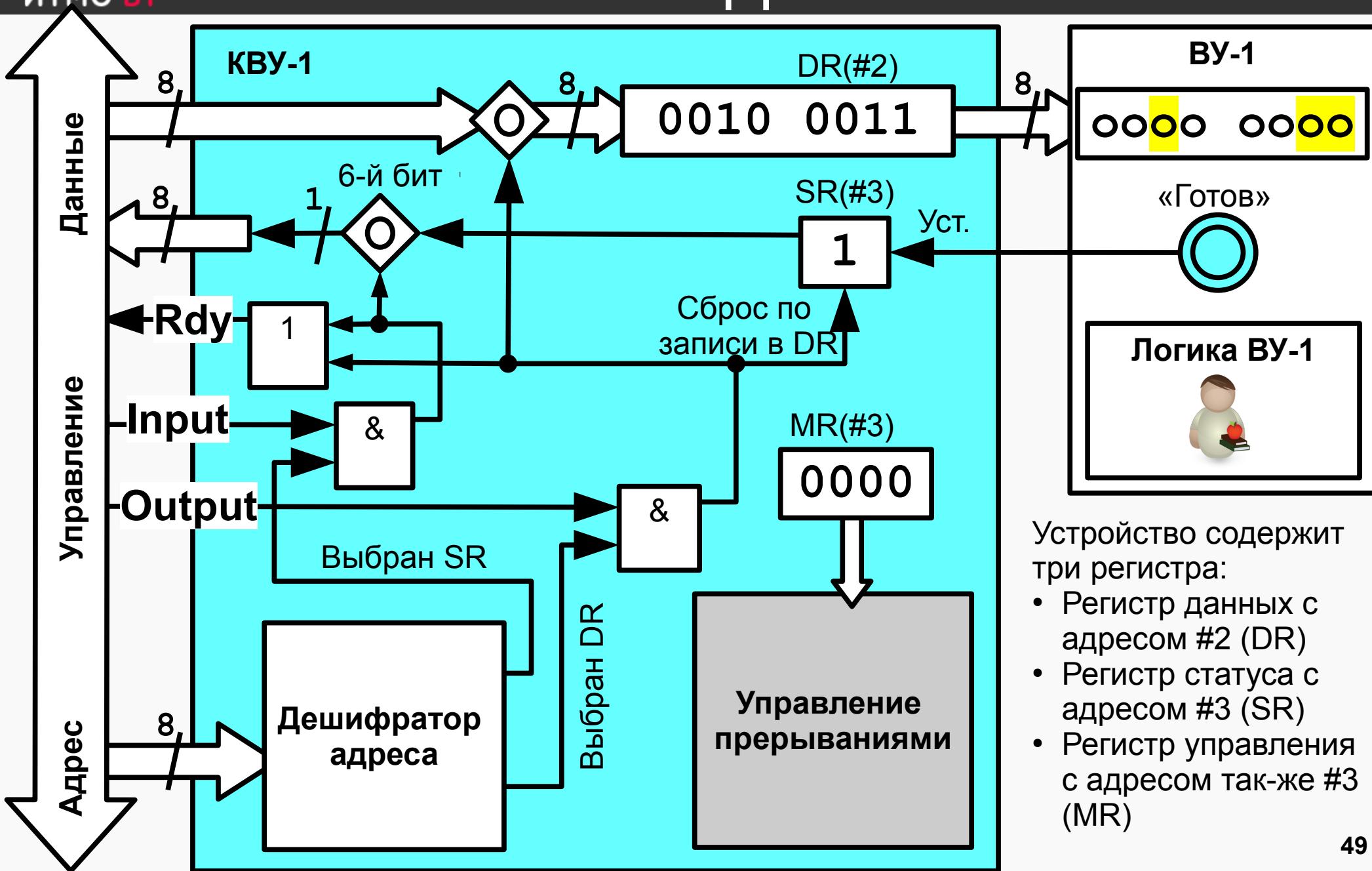


Адрес			Строка							
A_2	A_1	A_0	L_7	L_6	L_5	L_4	L_3	L_2	L_1	L_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Шина БЭВМ-NG и внешние устройства



Контроллер и устройство вывода ВУ-1

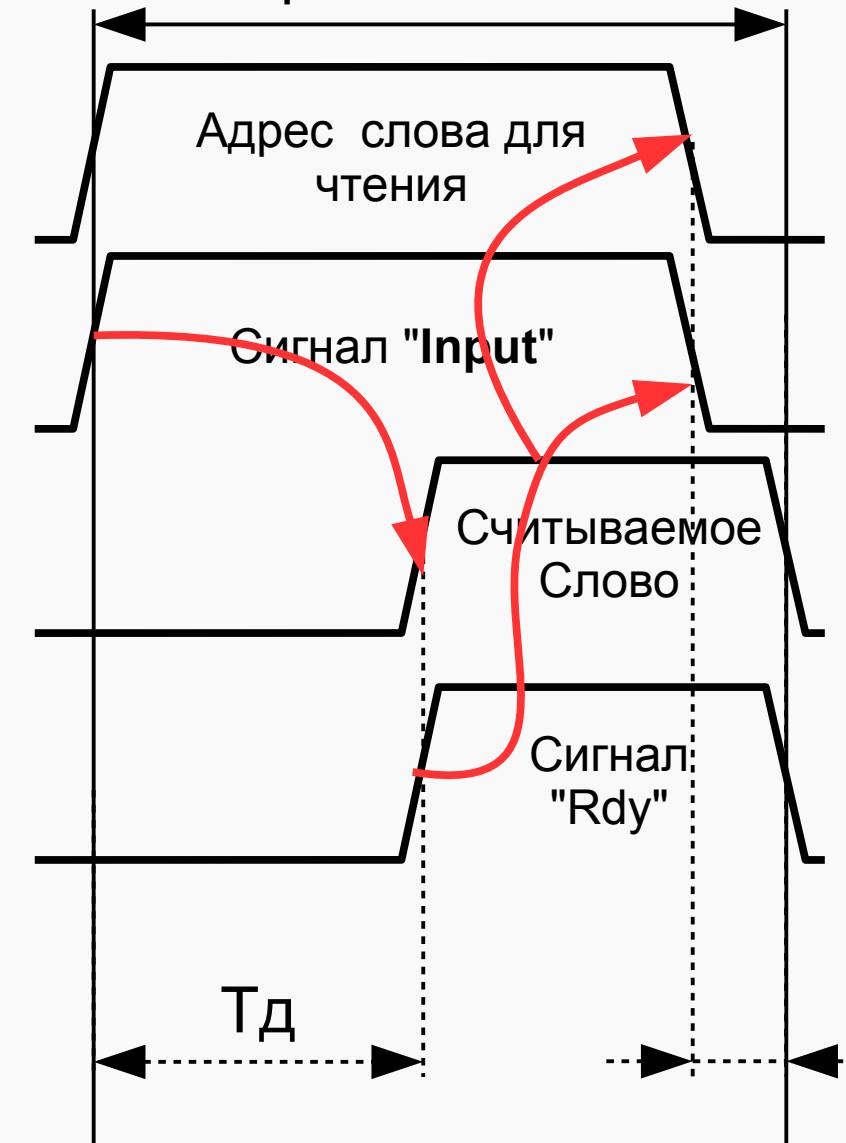


Устройство содержит три регистра:

- Регистр данных с адресом #2 (DR)
- Регистр статуса с адресом #3 (SR)
- Регистр управления с адресом так-же #3 (MR)

Диаграммы ввода-вывода

Цикл обмена



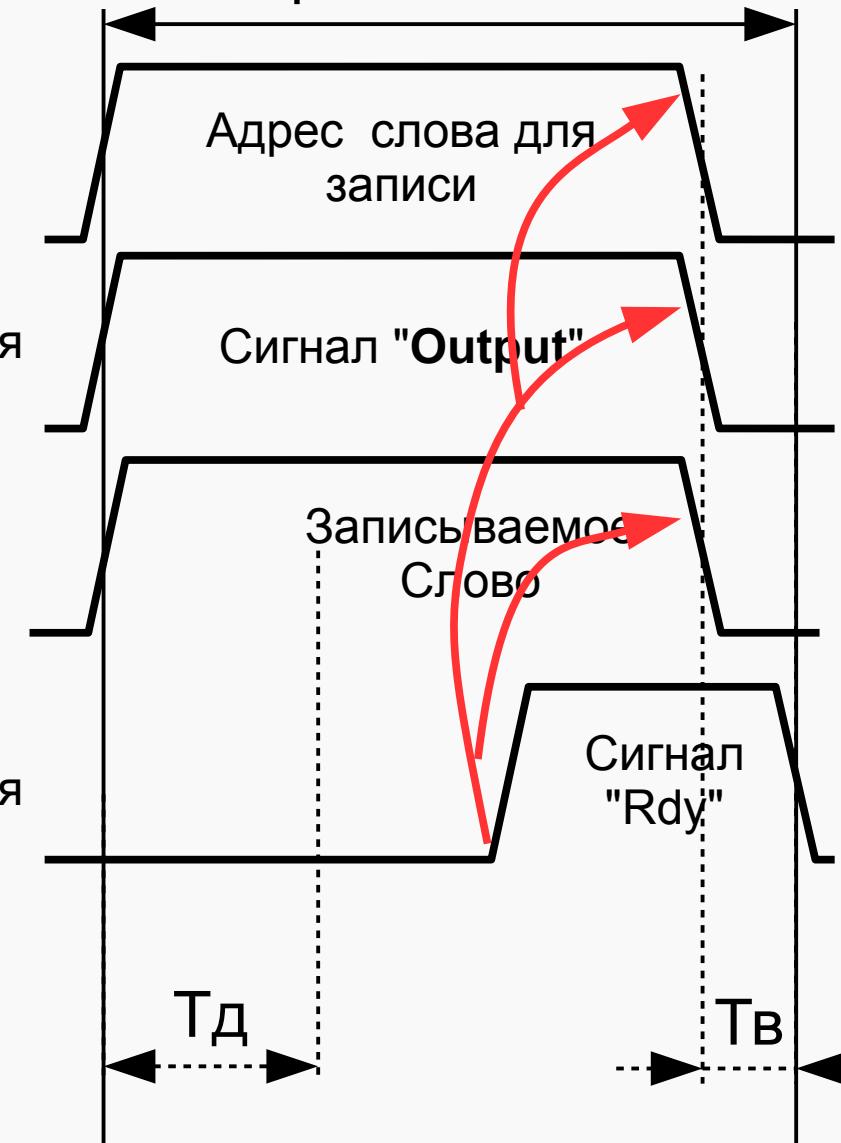
Шина Адреса

Шина Управления

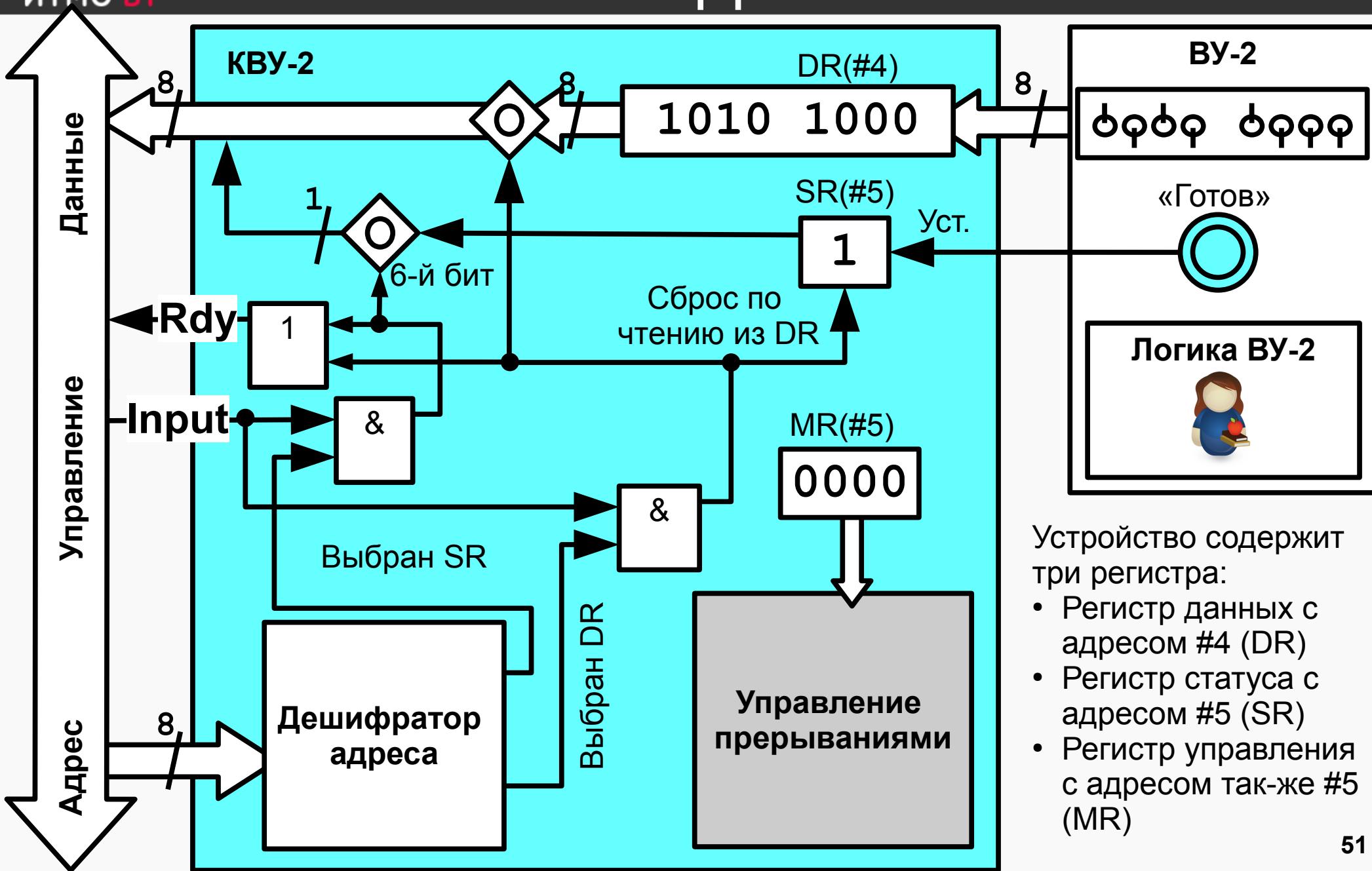
Шина Данных

Шина Управления

Цикл обмена



Контроллер и устройство ввода ВУ-2



Устройство содержит три регистра:

- Регистр данных с адресом #4 (DR)
- Регистр статуса с адресом #5 (SR)
- Регистр управления с адресом так-же #5 (MR)

Ассемблер БЭВМ

Назначение	Синтаксис	Пример использования
Размещение в памяти	ORG адрес	ORG 0x10
Адресная команда	[метка:] МНЕМОНИКА АРГУМЕНТ	LD X ;прямая относительная ST \$Y ;прямая абсолютная LD -(X) JUMP (VALUES) SWAM (ARRAY) +
Безадресная команда	[метка:] МНЕМОНИКА	START: CLA
Команда ввода-вывода	[метка:] МНЕМОНИКА АДРЕСВУ	OUT 0x3
Константы	[метка:] WORD знач. [,знач...] [метка:] WORD кол. DUP (знач.)	X: WORD ? Y: WORD X VALUES: WORD 1,2,3 ARRAY: WORD 10 DUP (?)



Ассемблер БЭВМ

Подсчет отрицательных элементов массива

```
ORG      0x10
ADDR:    WORD     $X ; Адрес первого элемента массива
I:       WORD     0   ; Адрес текущего элемента
N:       WORD     6   ; Количество элементов массива
R:       WORD     0   ; Результат
START:   CLA      ; Первая команда программы
          ST       R
          LD       ADDR
          ST       I
next:    LD       (I) +
          BPL     SKIP
          OR       (R) + ; важна не команда, а адресация
SKIP:    LOOP    N
          BR       NEXT
          HLT
ORG      0x030
X:       WORD     6 DUP (?) ; Элементы массива
```

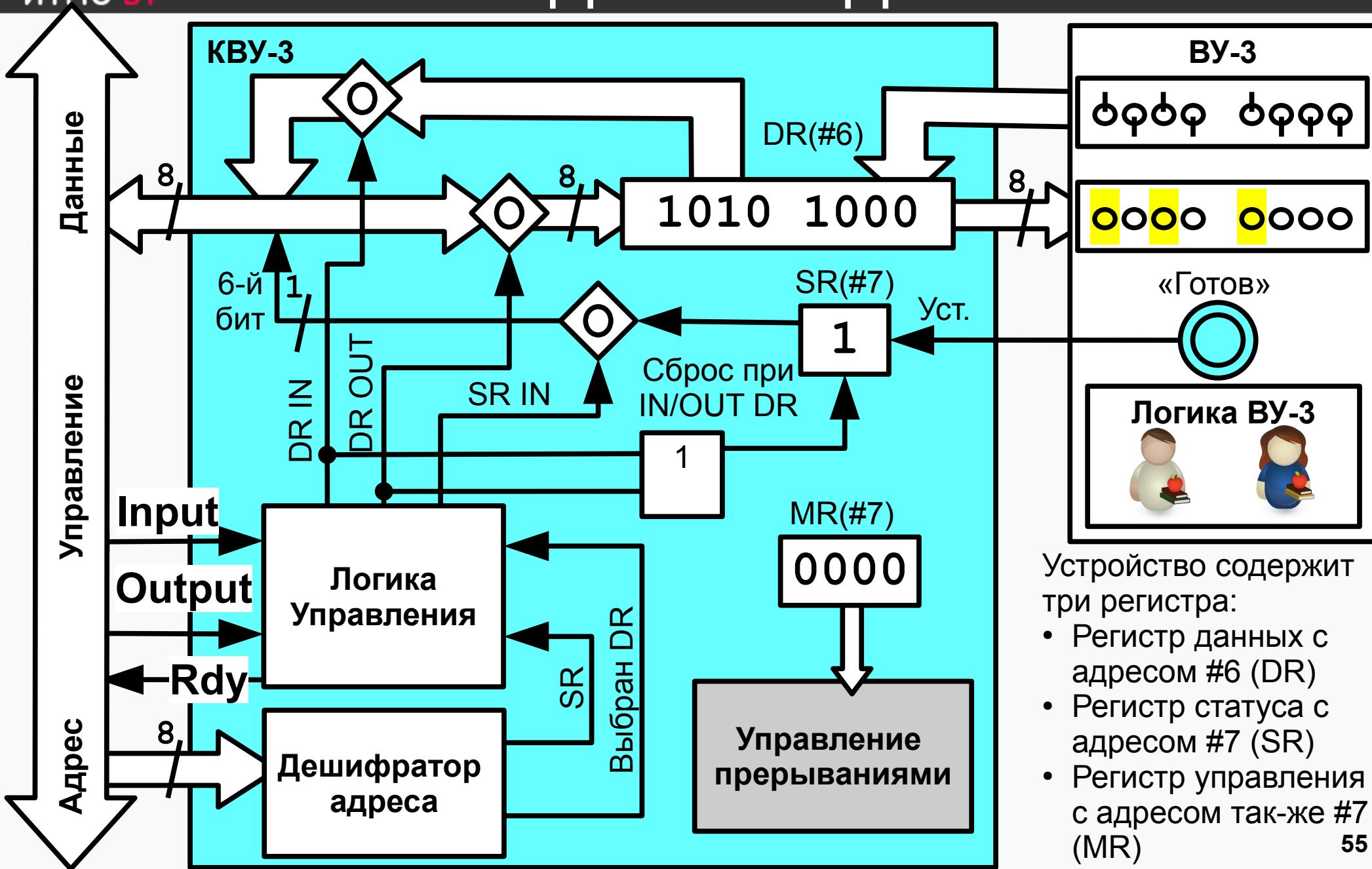
ЦИКЛ ОЖИДАНИЯ ВВОДА

Ввод двух символов с устройства ввода ВУ-2 (DR#4, SR#5)

	ORG	0x10	
START:	CLA		
S1:	IN	5	; Ожидание ввода первого символа
	AND	#0x40	; Бит 6 SR == 0 («Готов» нажата?)
	BEQ	S1	; Нет – “Спин-луп”
	IN	4	; Ввод первого символа
	SWAB		; Перемещение первого символа в ; старшую часть АС
	ST	RES	; Сохранение его в ячейке RES
S2:	IN	5	; Ожидание ввода второго символа
	AND	#0x40	; Бит 6 SR == 0 («Готов» нажата?)
	BEQ	S2	; Нет – “Спин-луп”
	LD	RES	
	IN	4	; Ввод второго в младшие 8 разрядов А
	ST	RES	
	HLT		
RES:	WORD	?	; Ячейка для записи слова “ДА”

Сколько циклов команд БЭВМ будет ждать ввода второго символа?

Контроллер и устройство ввода-вывода ВУ-3



БЭВМ: у-во ввода-вывода ВУ-4 (регистры 0x8 - 0xB)

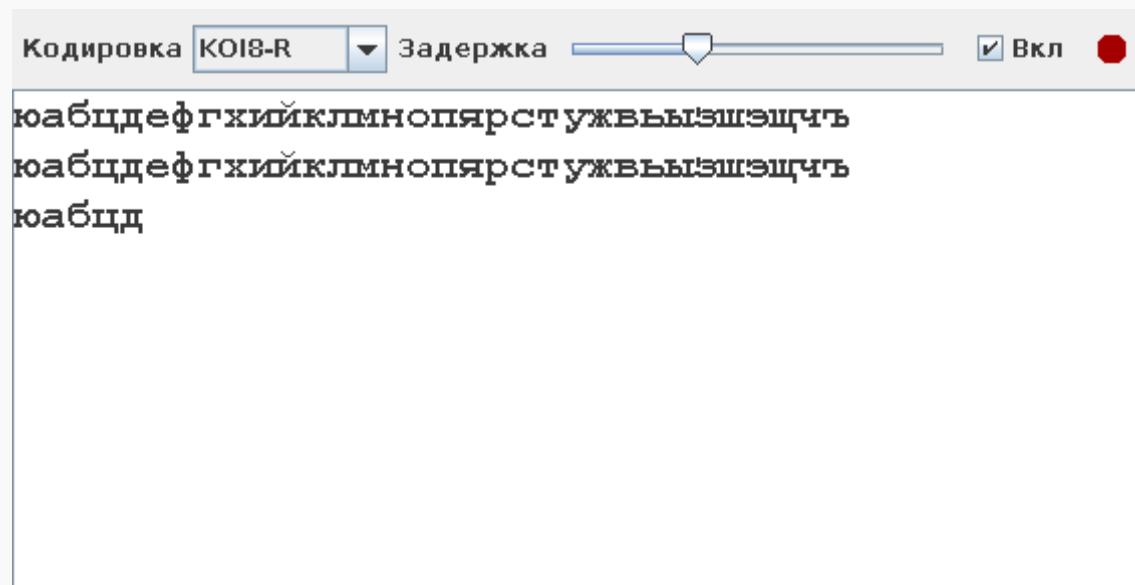
- По функционалу похоже на ВУ-3
- Адресуется 4-мя регистрами
- Отдельные регистры для входных (0x8 и выходных (0x9) данных
- Регистр состояния по адресу (0xA)
 - Бит #6 все также отвечает за готовность ввода-вывода
- Регистр управления по адресу (0xB)
- Все регистры доступны для чтения записи
- Позволяет реализовать сложные конфигурации подключения

БЭВМ: ВУ-0 Таймер (регистры 0x0, 0x1)

- Устанавливает готовность (и вызывает прерывание) раз в 100^*DR миллисекунд
 - Смещенная десятичная фиксированная точка
- Если DR==0 готовность не устанавливается
- Можно использовать для организации синхронного обмена (Как?)

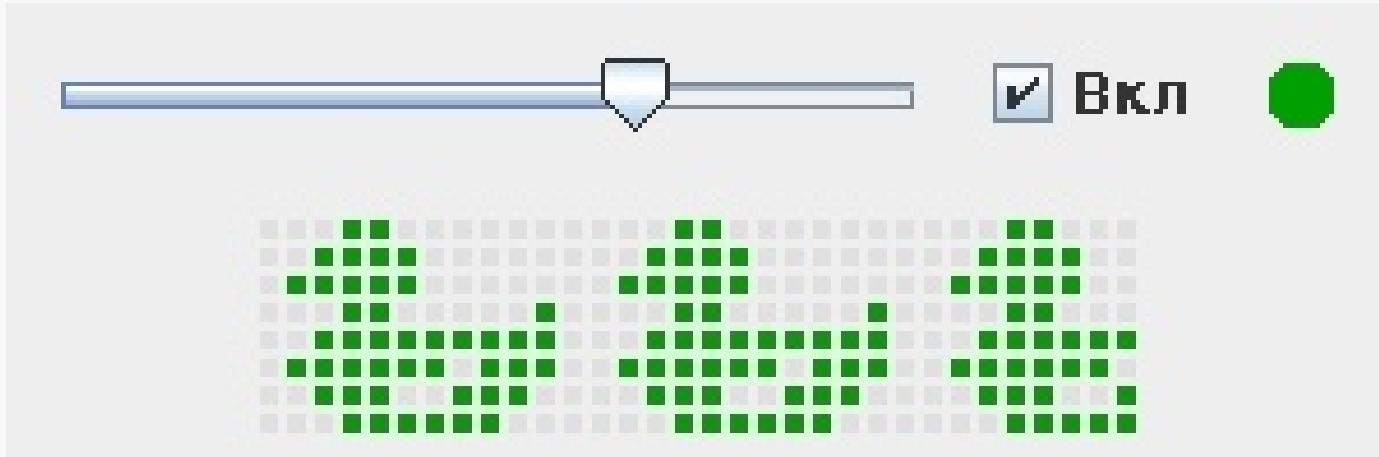
БЭВМ: ВУ-5 Текстовый принтер (регистры 0xC - 0xF)

- Печатает символы из РДВУ в заданной кодировке
- Регулируемая задержка (время печати) от 100 мс до 10 с
- Перевод строки по символу CR ($0A_{16}$)
- NUL (0) — очистка листа
- Остальные - неопределенное поведение



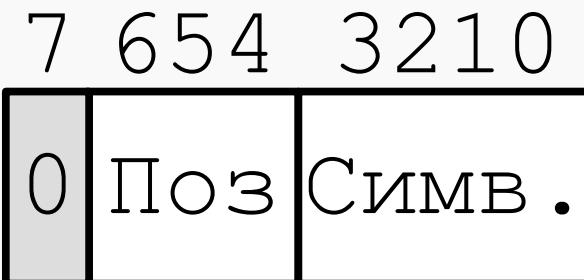
БЭВМ: ВУ-6 Бегущая строка (регистры 0x10 - 0x13)

- Размер матрицы: 32x8
- Сдвиг при записи нового значения в РДВУ
- Новое значение — справа
- Младший бит - нижний

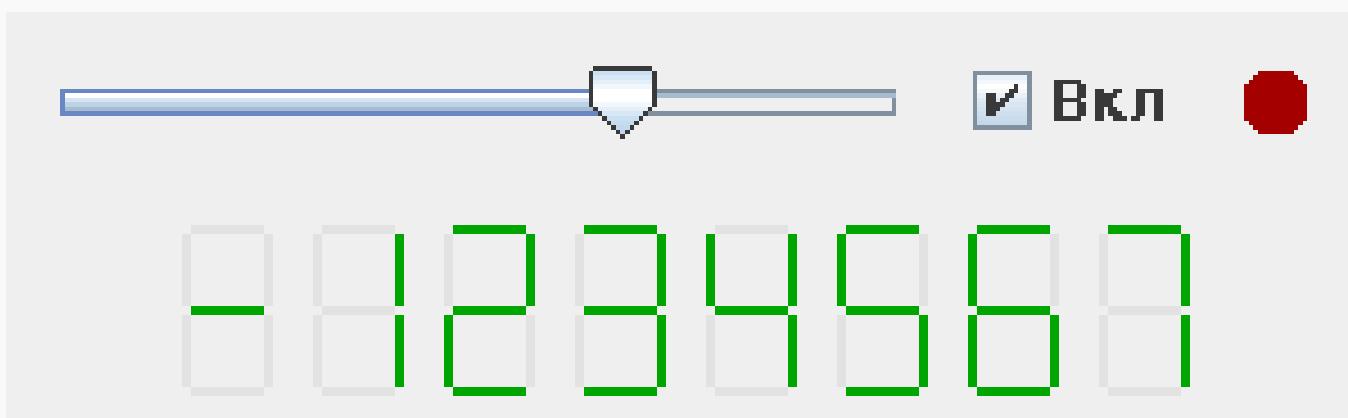


БЭВМ: ВУ-7 8-ми разрядный 7-сегм. индикатор (0x14 - 0x17)

- Формат РДВУ:

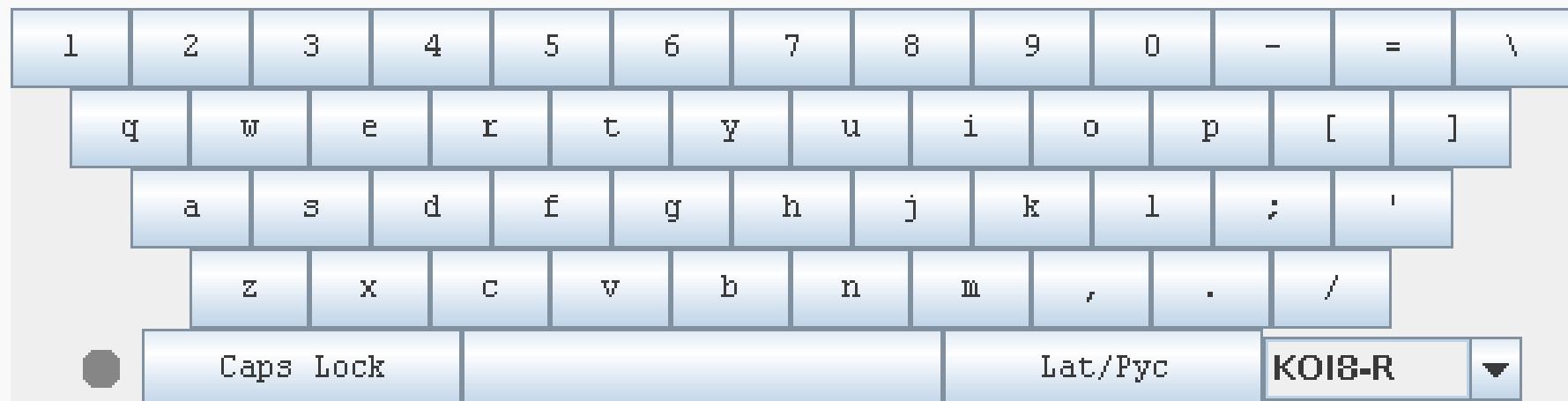


- Симв==(A_{16}) — установка в разряде знака «-»
- Симв==($B_{16}-F_{16}$) — сброс разряда



БЭВМ: ВУ-8 клавиатура (регистры 0x18-0x1C)

- Код нажатой клавиши в выбранной кодировке устанавливается в РДВУ
- Автоматически устанавливается готовность



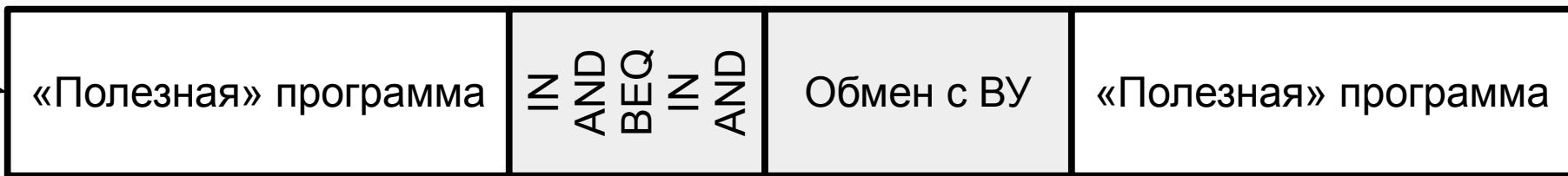
БЭВМ: ВУ-9 Цифровая клавиатура (0x1C-0x1F)

- При нажатии клавиши ее код помещается в РДВУ
- Клавиша 0-9 код 0-9
- Клавиша «-» код А
- «+» код В
- «/» код С
- «*» код D
- «.» код Е
- «=» код F

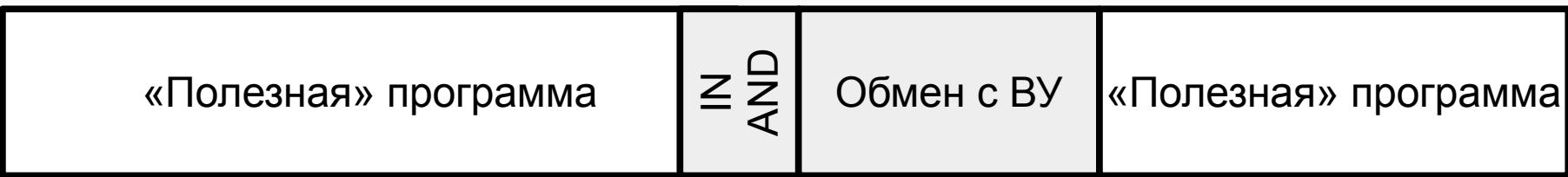
7	8	9	/
4	5	6	*
1	2	3	-
0	.	=	+

Временные издержки

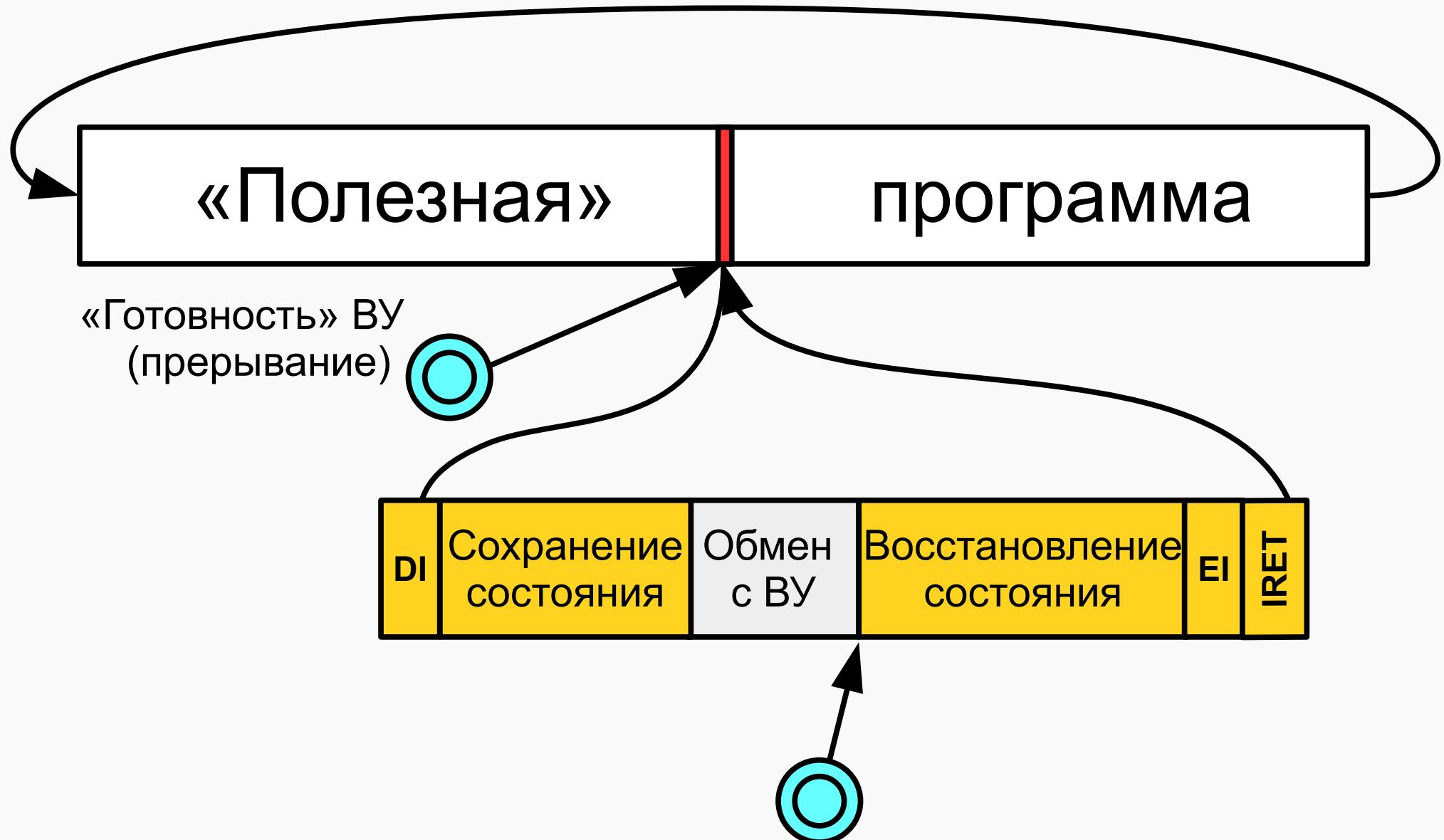
1)



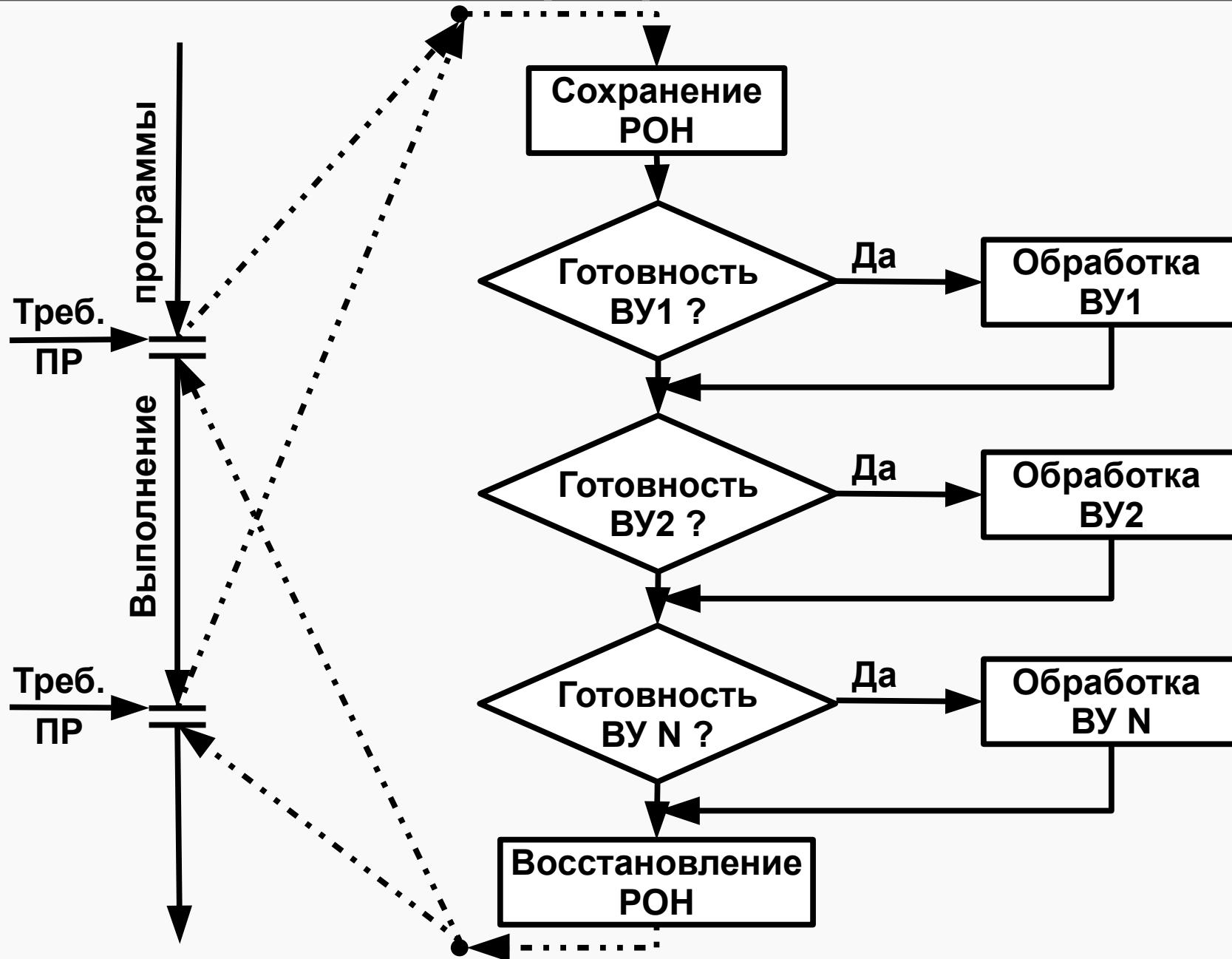
2)



Инициация обмена по прерыванию



Логика обработки и приоритет: одно прерывание, много ВУ.

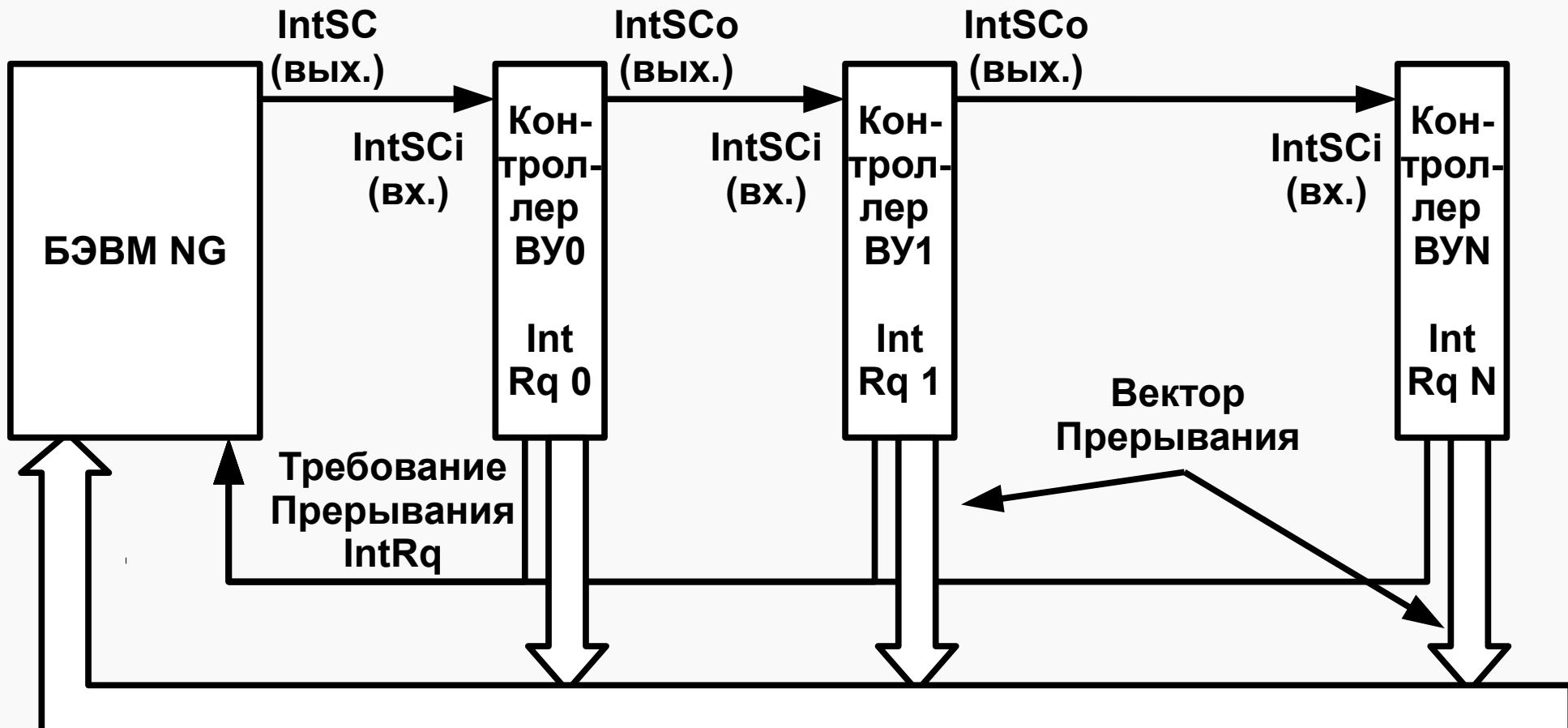


Вектор прерывания

- Совокупность адреса программы обработки прерывания и регистра состояния (PS)
- Необходимо инициализировать перед началом обработки прерывания
 - Хотя бы установить на подпрограмму, которая ничего не делает
 - Ответственность OS и БИОС
- В БЭВМ-NG ячейки с 0x0 по 0xF
 - Всего 8 векторов, по два слова на вектор
 - На одном векторе может быть несколько прерываний

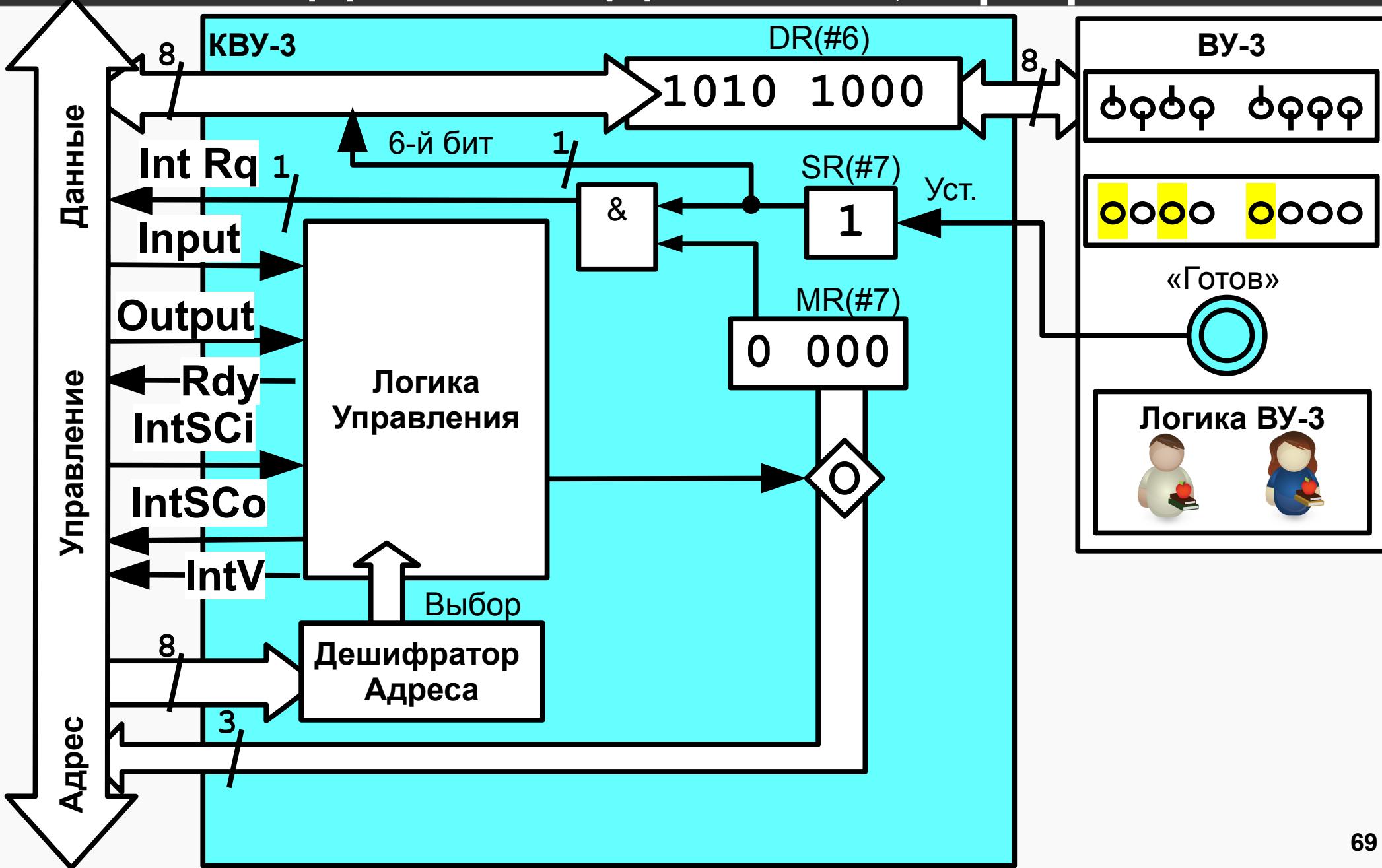
Адр.	Сод.
0x0	Адр 0
0x1	PS 0
0x2	Адр 1
0x3	PS 1
0x4	Адр 2
0x5	PS 2
0x6	Адр 3
0x7	PS 3
0x8	Адр 4
0x9	PS 4
0xA	Адр 5
0xB	PS 5
0xC	Адр 6
0xD	PS 6
0xE	Адр 7
0xF	PS 7

Организация прерываний БЭВМ NG



ПРП (IntSC) — Предоставление Прерывания (Interrupt supply chain).
Может быть входной и выходной.

Контроллер и устройство ввода-вывода ВУ-3, прерывания



Шпаргалка сигналов шины

Сигнал (ы)	Направление CPU. . .CNTRL	Описание
Addr0..7	CR → Деш. адреса	Шина адреса регистров контроллеров внешних устройств
Int# Addr0..2	CR ← MR контр.	Шина номера вектора прерывания (НВП)
IntV	CPU ← л/у контр.	Сигнал подтверждения передачи в CR НВП
Input	Деш. команд → л/у	Сигнал управления "Ввод"
Output	Деш. команд → л/у	Сигнал управления "Выход"
Rdy	CPU ← л/у	Сигнал подтверждения готовности
IntRq	PS (IRQ) ← SR контр.	Запрос прерывания
IntSC IntSCI# IntSCo#	УМК IRQS → IntSCI0 IntSCo0 → IntSCI1 IntSCo1 → IntSCI2	Цепочка сигналов предоставления прерывания. Передается далее в случае отсутствия требования прерывания в контроллере.
SYN	Такт. ген → л/у	Синхросигнал тактового генератора
Data0..7	AC ↔ рег. контр.	Двунаправленная шина данных связи АС с регистрами контроллера

Регистр состояния и команды

Бит	Мнем.	Содержимое
0	C	Перенос
1	V	Переполнение
2	Z	Нуль
3	N	Знак
4	O	0 – используется для организации безусловных переходов в МПУ
5	EI	Разрешение прерываний
6	INT	Прерывание (логическое "И" шины запроса на прерывание и бита 5 PC – "разрешение прерываний")
7	W	Состояние тумблеров РАБОТА/ОСТАНОВ (1 – РАБОТА)
8	P	Программа

Цикл прерывания

- **if PS(W) = 0 then GOTO STOP;** Проверка тумблера работа-останов, стоп если останов
 - **if PS(INT) = 0 then GOTO INFETCH;** Если нет прерывания, то на выборку след. команды
 - **INTS ;** Сформировать сигнал предоставление прерывания
 - **~0 + SP → SP, AR**
 - **IP → DR**
 - **DR → MEM(AR)**
- }; IP → -(SP)

Цикл прерывания (2)

- $\sim 0 + SP \rightarrow SP, AR$
 - $PS \rightarrow DR$
 - $DR \rightarrow \text{MEM}(AR) ;$
 $LTOI(CR) \rightarrow BR ;$ младшие 8 разрядов CR
(номер вектора прерывания) записать в BR
 - $SHL(BR) \rightarrow BR, AR ;$ Вычисляем адрес ячейки с переходом на подпрограмму обработки прерывания, как номер вектора * 2
- } ; $PS \rightarrow -(SP)$
а также...

Цикл прерывания (3)

- **MEM (AR) → DR**; адрес обработчика прерывания записать в DR ...
- **DR → IP**; ... а затем в IP
- **LTOI (BR + 1) → AR**; ... выбрать адрес следующей ячейки вектора прерывания, ограничивая результат 8-ю разрядами
- **MEM (AR) → DR**; содержимое PS обработчика прерывания записать в DR ...
- **DR → PS**; ... а затем установить его в регистр

После каких команд нет цикла прерывания?

?

Обработка прерываний: Основная программа

Готовность ВУ1: 2*A→РДВУ1, Готовность ВУ3: РДВУ3→ яч. ЗF

```

ORG 0x0          ; Инициализация векторов прерывания
V0: WORD $DEFAULT,0x180 ; Вектор прерывания #0
V1: WORD $INT1,0x180   ; Вектор прерывания #1
V2: WORD $DEFAULT,0x180 ; Вектор прерывания #2
V3: WORD $INT3,0x180   ; Вектор прерывания #3
...
V7: WORD $DEFAULT,0x180 ; Вектор прерывания #7
DEFAULT:IRET          ; Просто возврат

```

```

ORG 0x020 ; Заргужка начальных векторов прерывания
START: DI
        CLA
        OUT 1      ; MR КВУ-0 на вектор 0
        OUT 5      ; MR КВУ-2 на вектор 0
        LD #9       ; разрешить прерывания и вектор №1
        OUT 3      ; (1000|0001=1001) в MR КВУ-1
        LD #0xB     ; разрешить прерывания и вектор №3
        OUT 7      ; (1000|0011=1011) в MR КВУ-3
        ...
JUMP $PROG

```

Обработка прерываний: Программа и прерывание 1

Готовность ВУ1: 2*A→РДВУ1, Готовность ВУ3: РДВУ3→ яч. ЗF

PROG:

	EI	; Установка состояния разр. прерывания
	CLA	; Первоначальная очистка аккумулятора
INCLP:	INC	; Цикл для наращивания
	BR INCLP	; содержимого аккумулятора

	ORG 0x03F	
	; Ячейка для хранения кодов, поступающих с ВУ-3	
IO3:	WORD ?	

	ORG 0x040	
INT1:		; Прерывание сохранило содержимое PS
	NOP	; отладочная точка останова (NOP/HLT)
	PUSH	; Сохранили АС
	ASL	; Умножили АС на 2
	OUT 2	; Записали в РДВУ-1 (DR#2)
	POP	; Вернули АС назад
	NOP	; отладочная точка останова (NOP/HLT)
	IRET	; Возврат из обработки прерывания

Обработка прерываний: Прерывание 3

Готовность ВУ1: 2*A→РДВУ1, Готовность ВУ3: РДВУ3→ яч. ЗF

	ORG	0x040	
INT3:			; Прерывание сохранило содержимое PS
	NOP		; отладочная точка останова (NOP/HLT)
	PUSH		; АС кладем в стек
	CLA		
	IN	6	; РДВУ-З
	ST	\$IO3	; сохранение
	NOP		; отладочная точка останова (NOP/HLT)
	POP		; АС вынимаем из стека
	IRET		; Возврат из обработки прерывания

Ввод-вывод



Микропрограммное устройство управления

4



Многоуровневая ЭВМ

Уровень программных систем (специальный язык)

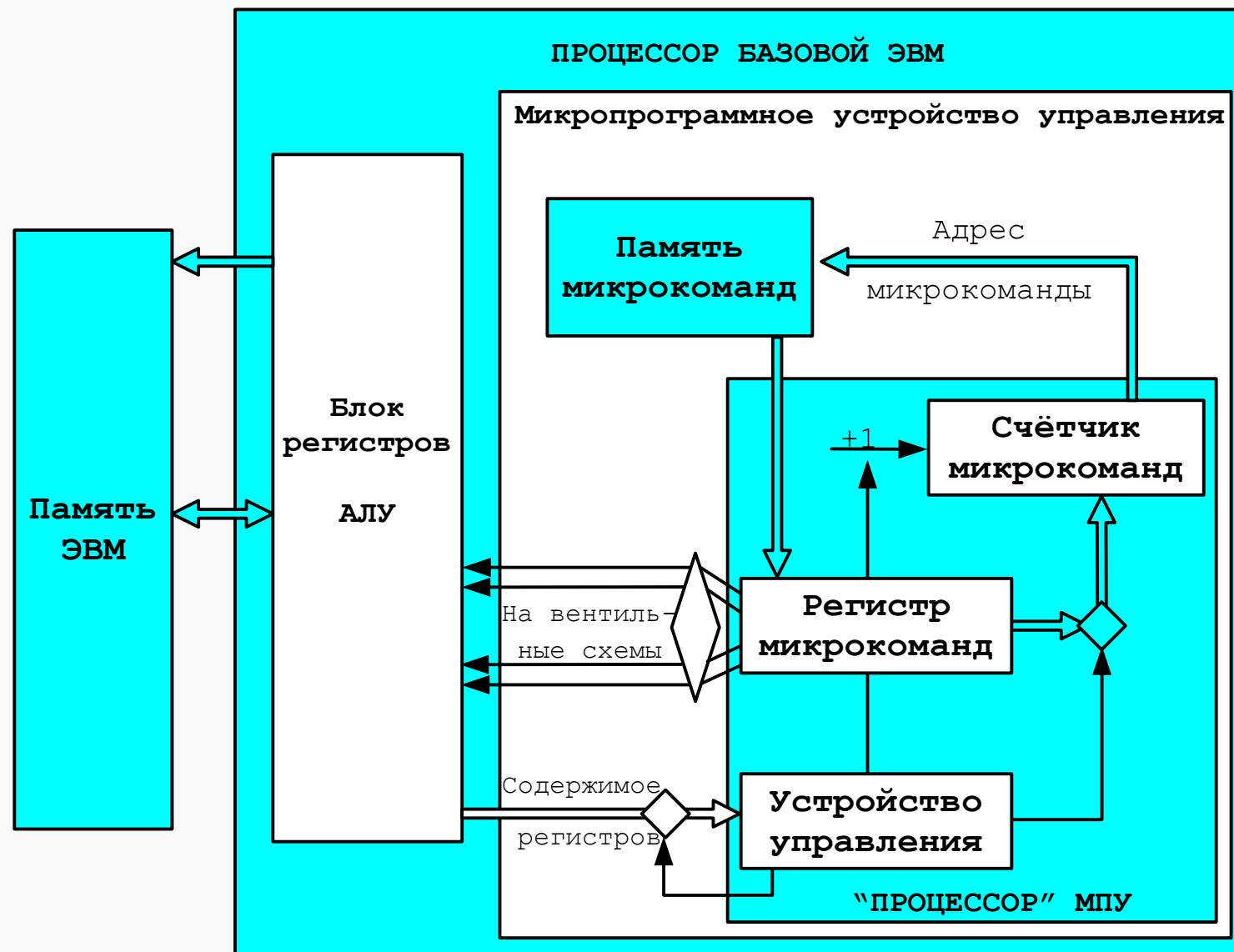
Уровень проблемно-ориентированных задач
(один из алгоритмических языков)

Язык Ассемблера

Машинные команды

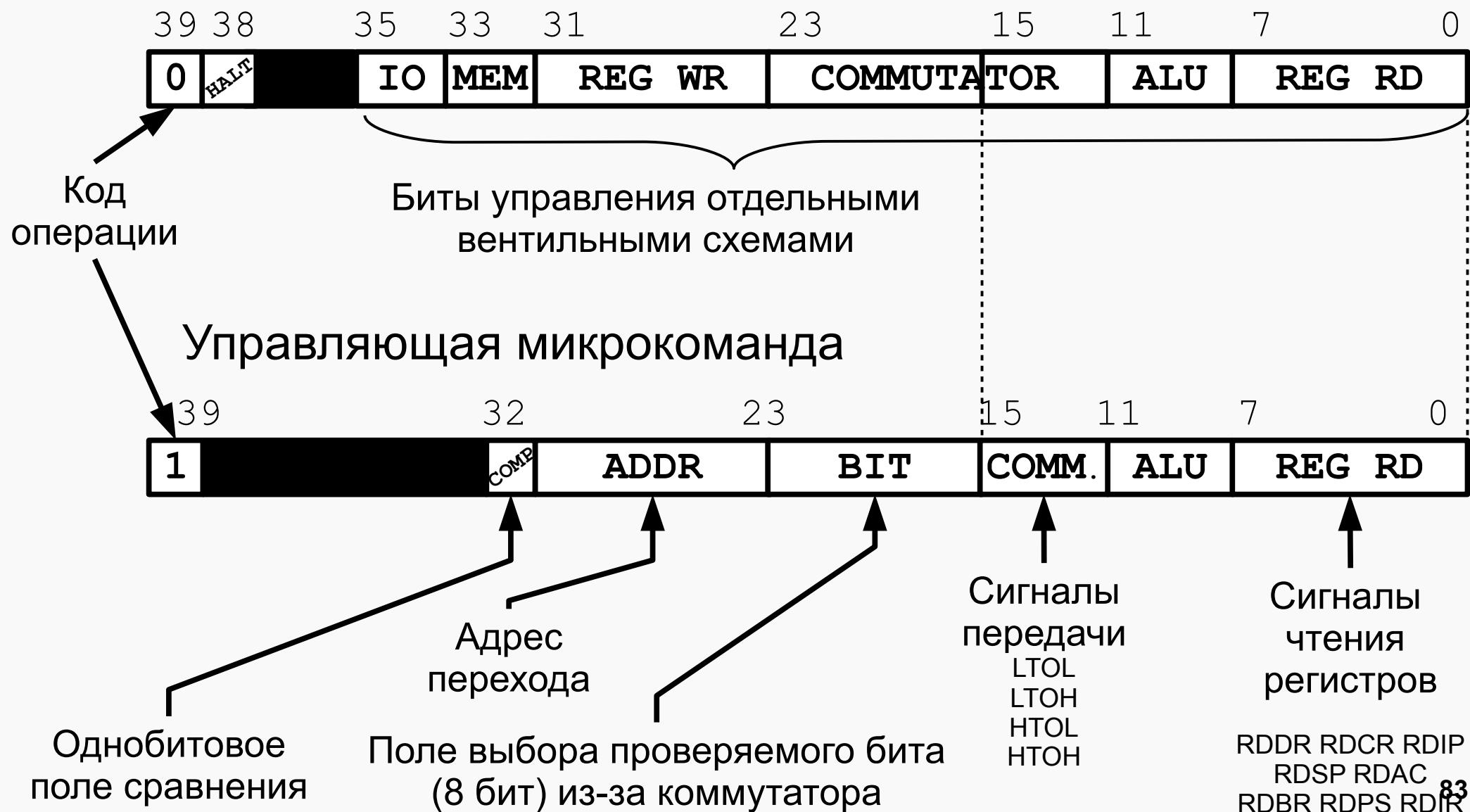
Микропрограммный
Уровень

МПУ

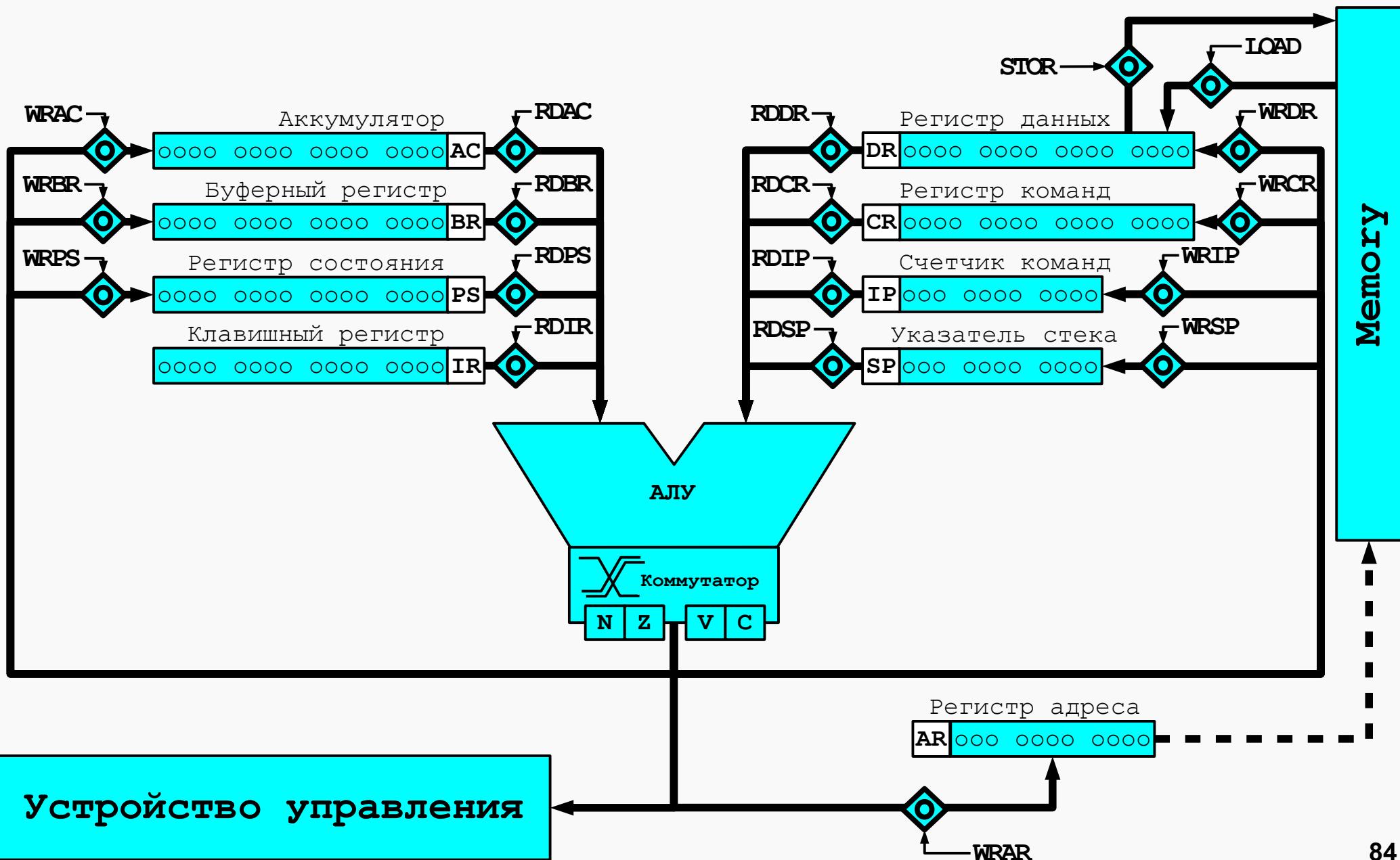


Горизонтальные микрокоманды

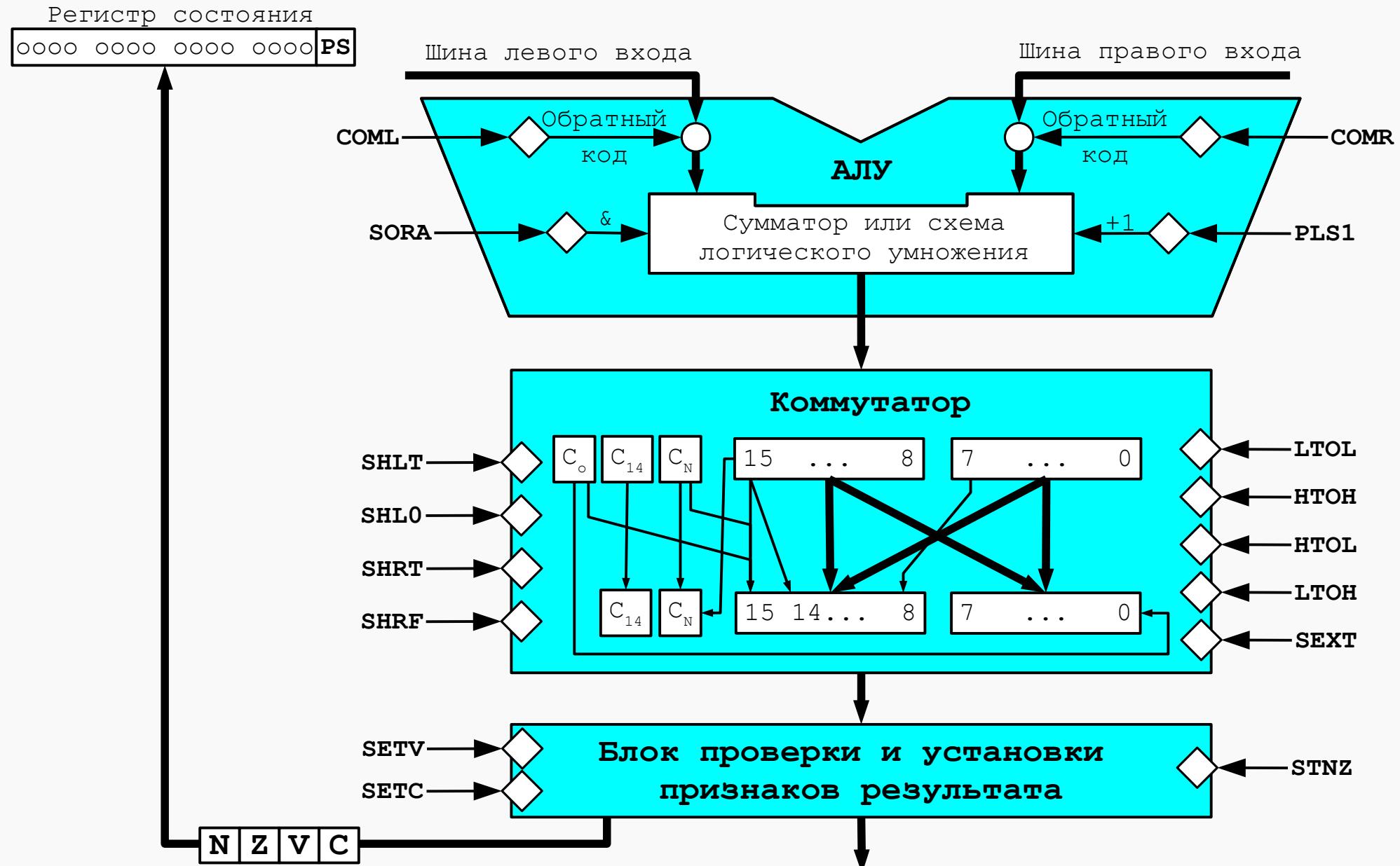
Операционная микрокоманда



Блок регистров

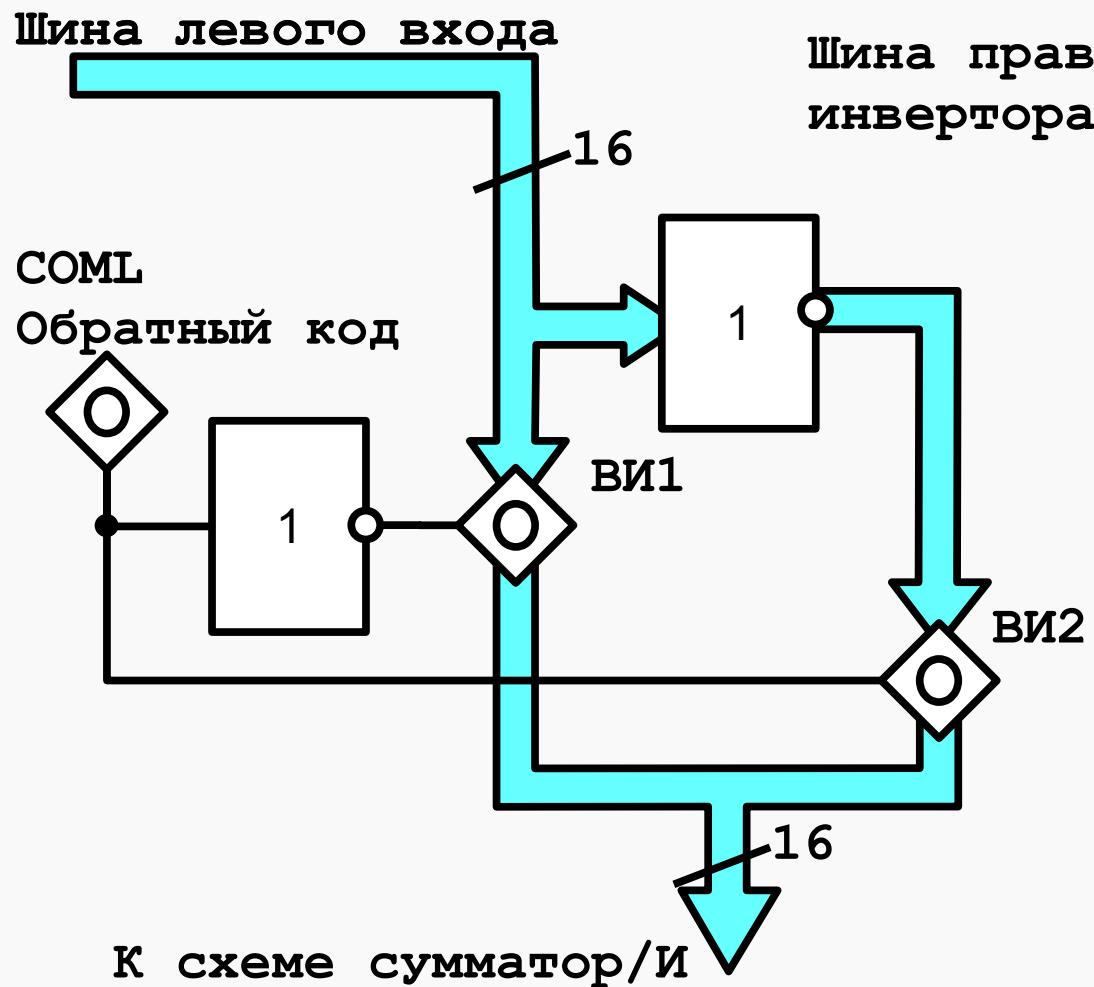


АЛУ, коммутатор, блок признаков результата

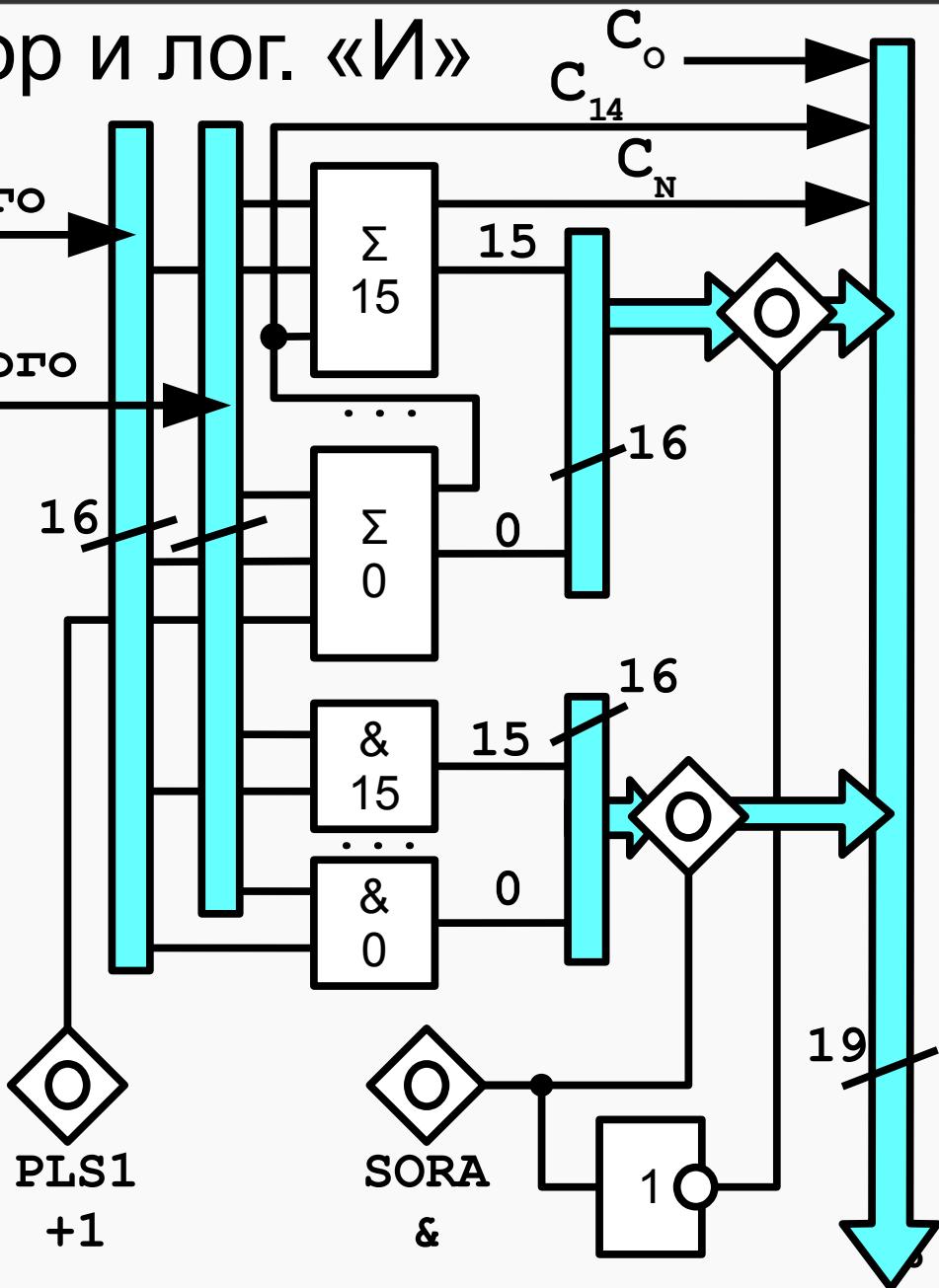


АЛУ: Обратный код, сумматор и логическое «И»

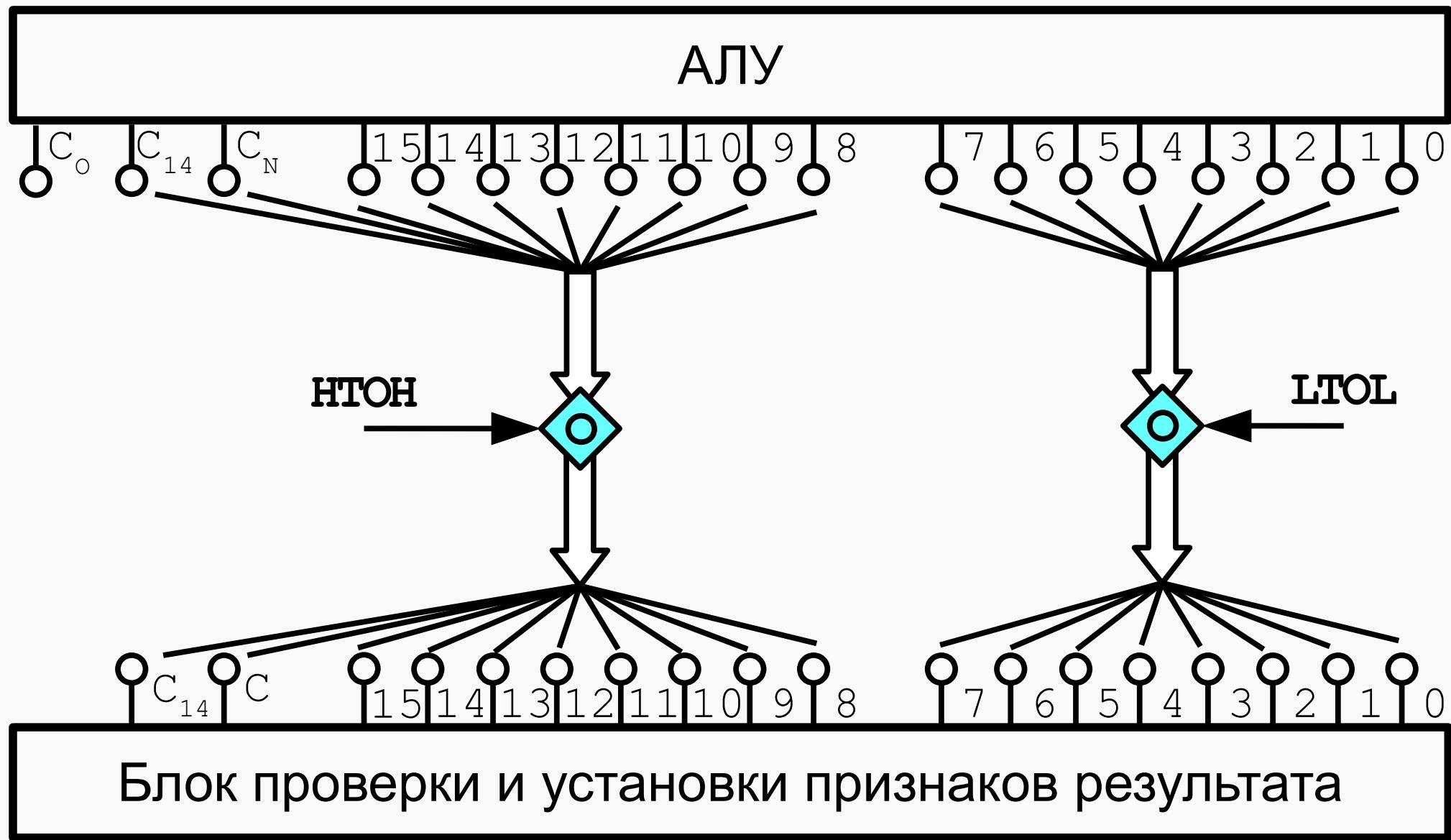
Схема обратного кода



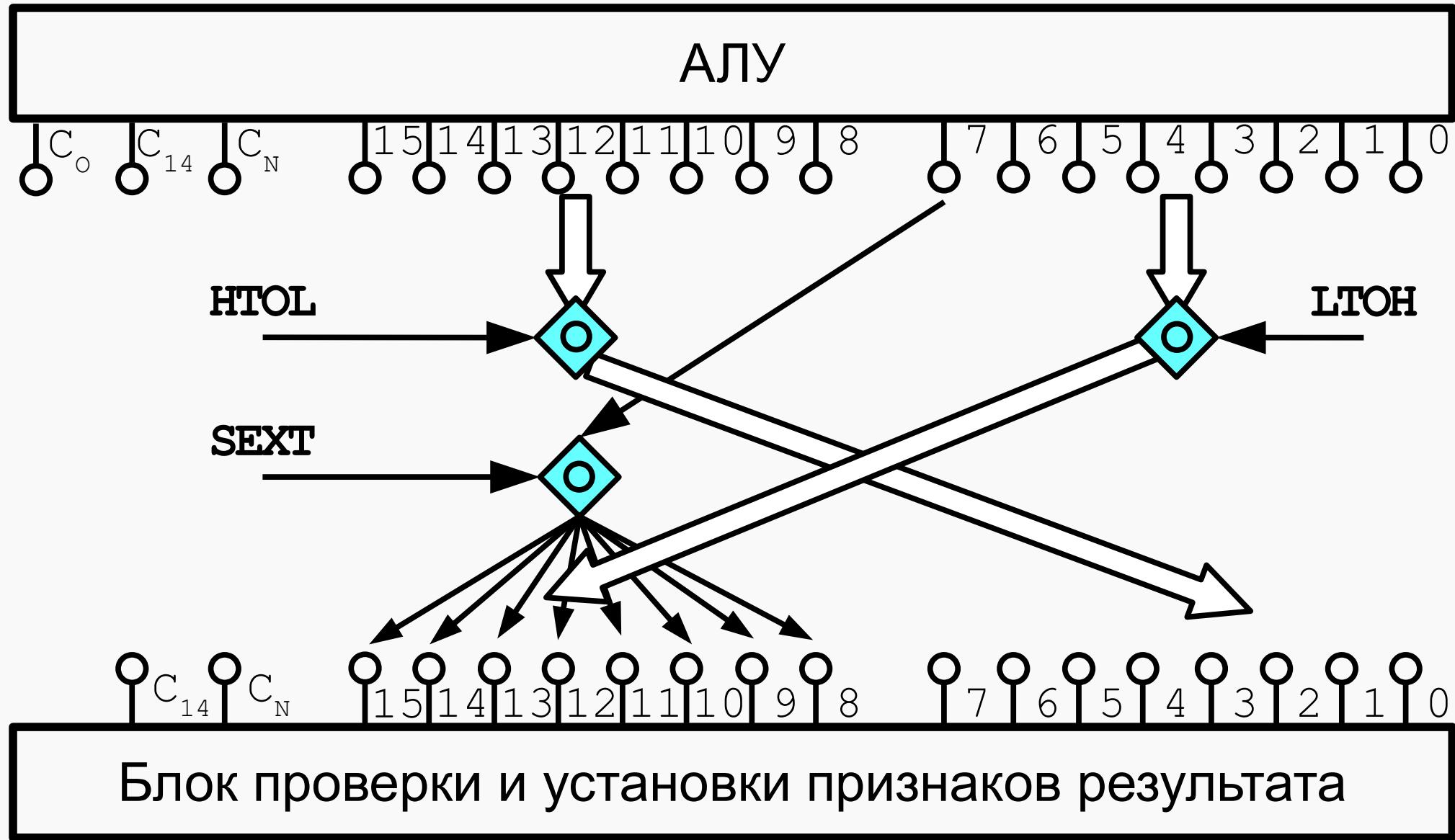
Сумматор и лог. «И»



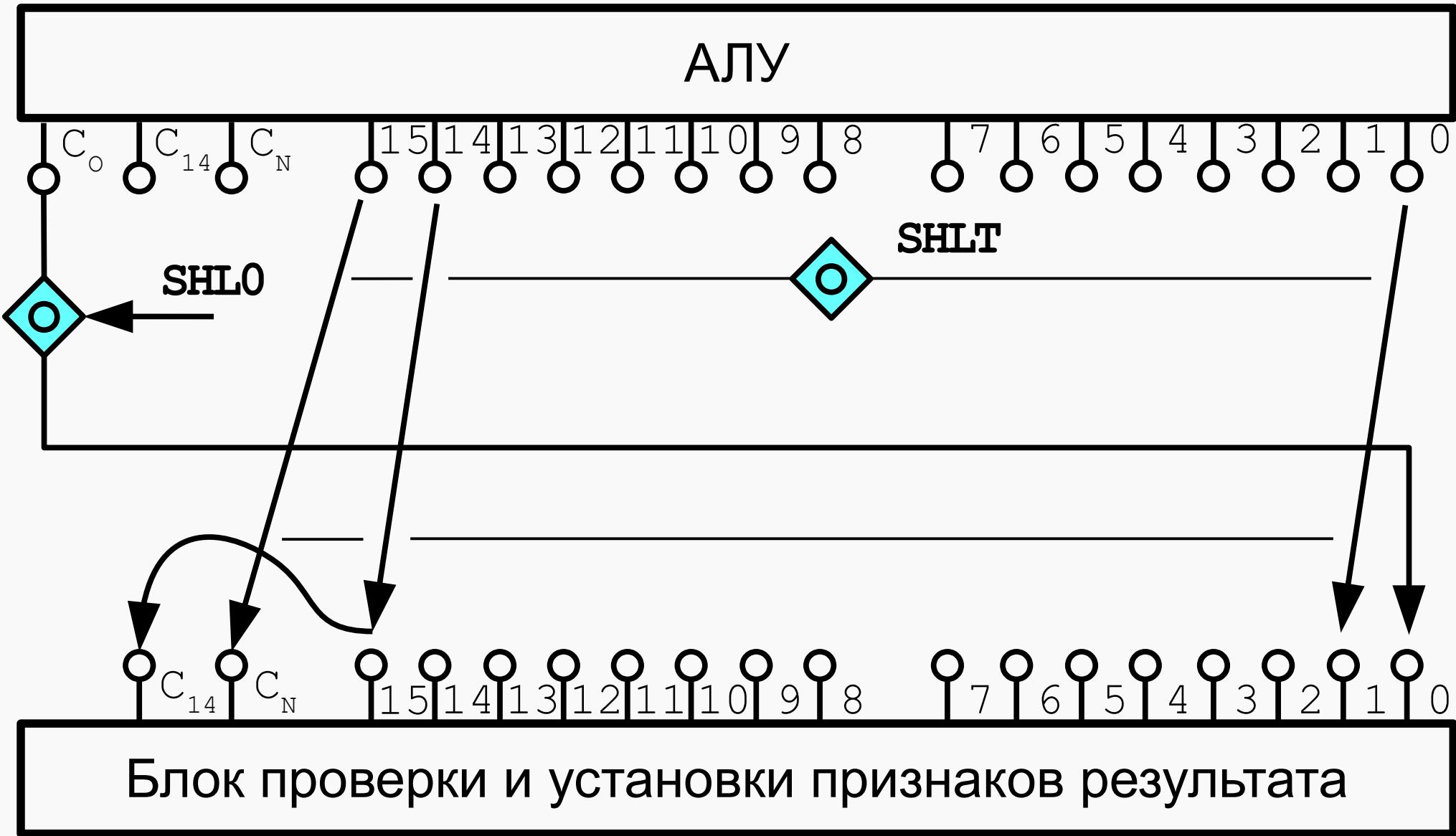
Коммутатор: Прямая передача



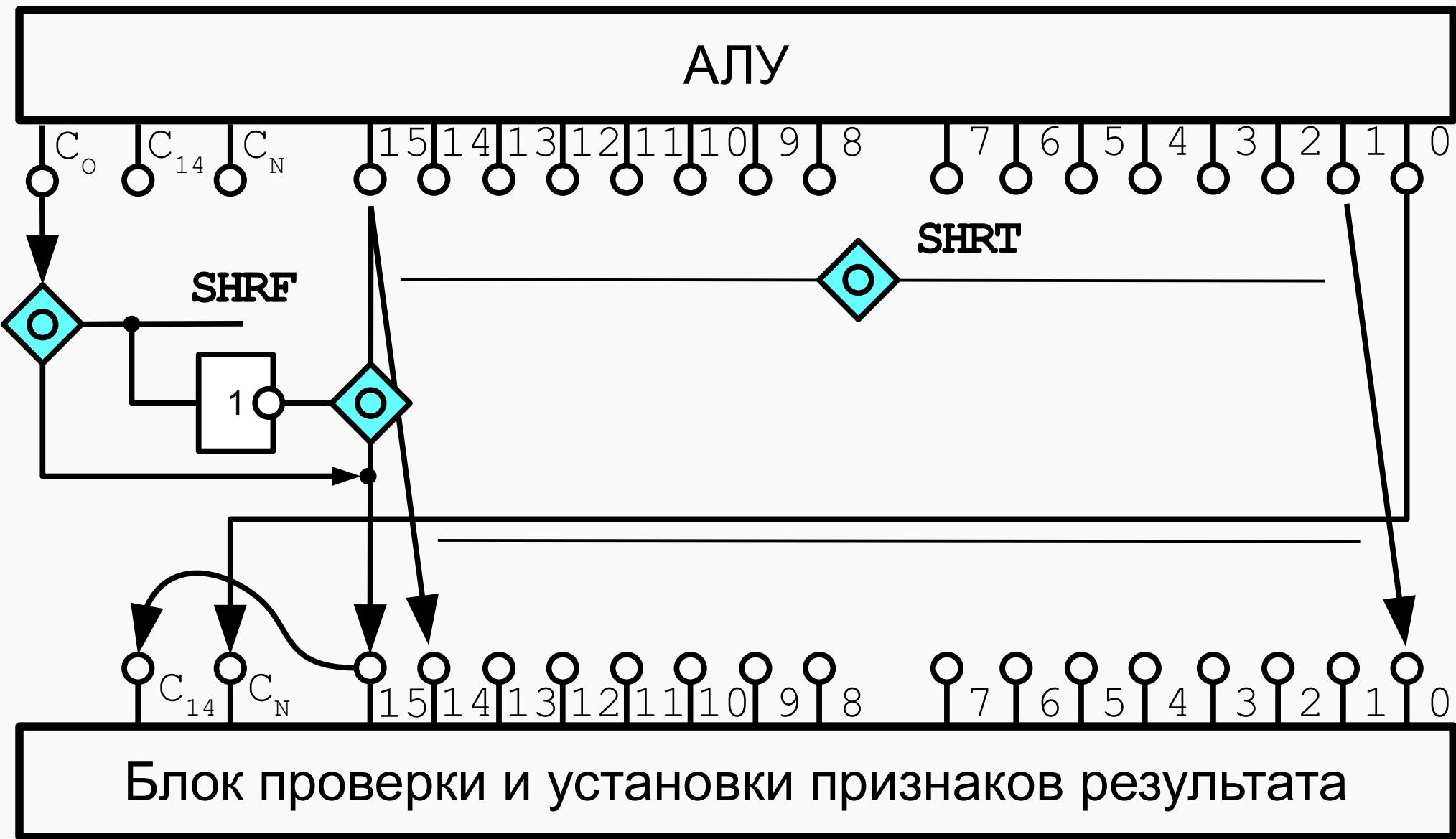
Коммутатор: обмен байтов, расширение знака



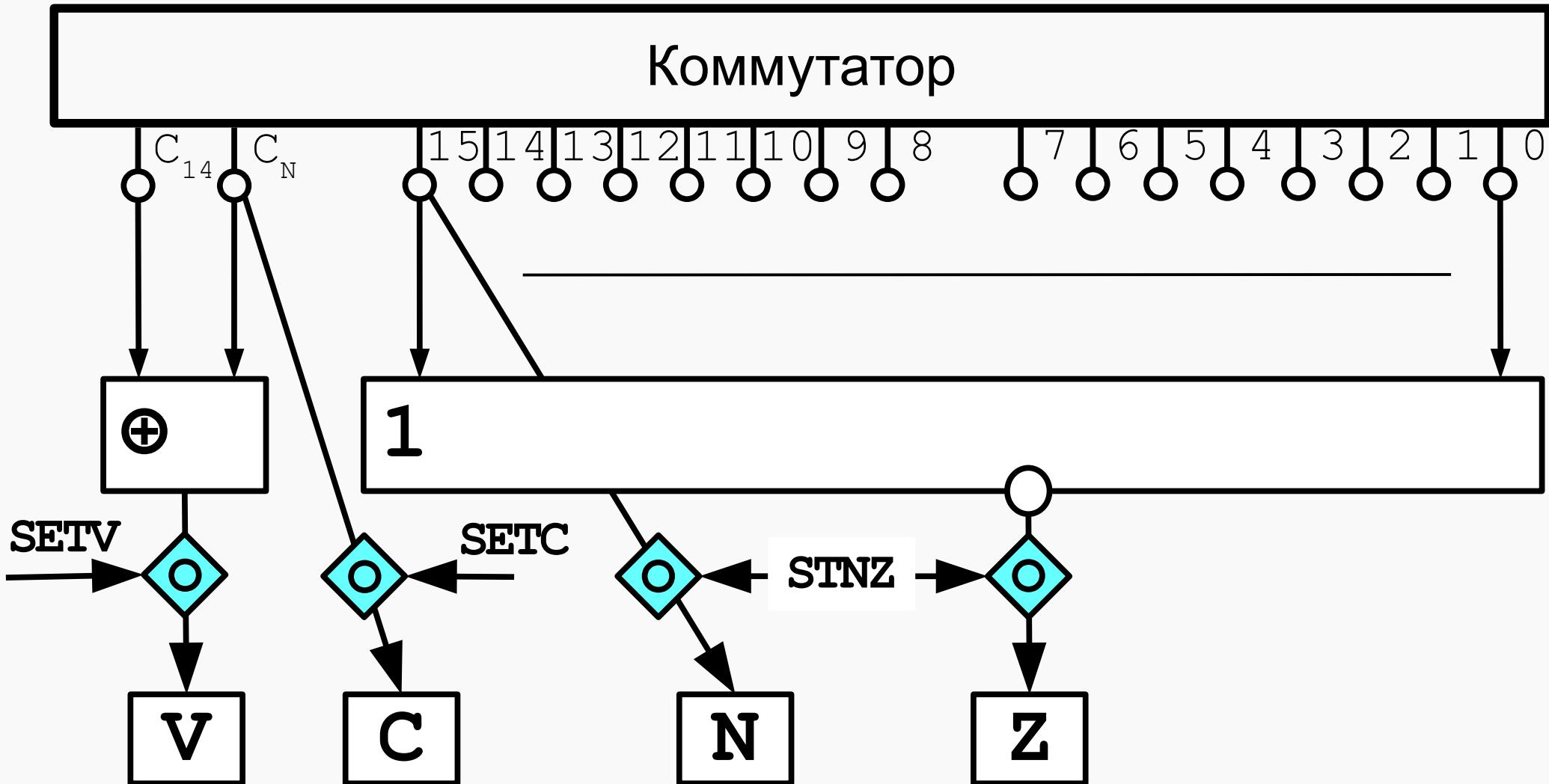
Коммутатор: ROL, ASL



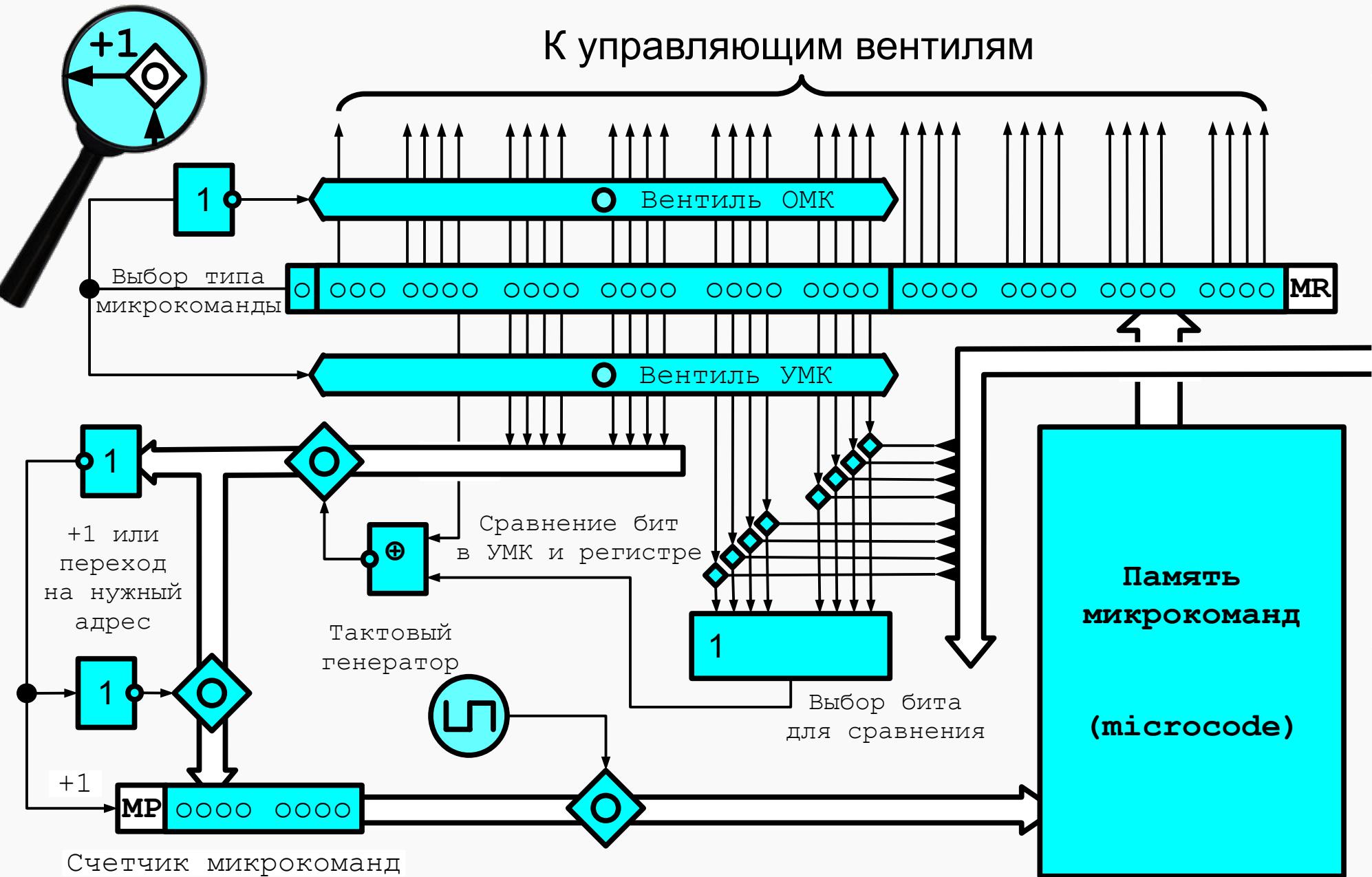
Коммутатор: ROR, ASR



Блок проверки и установки признака результата



Устройство управления

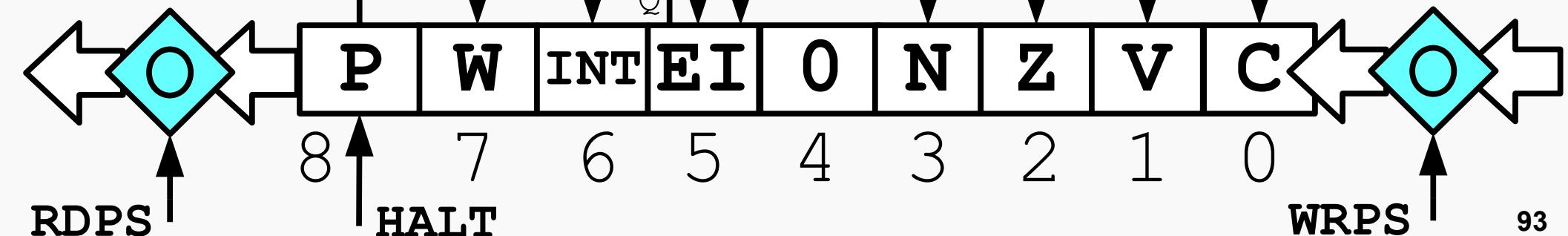
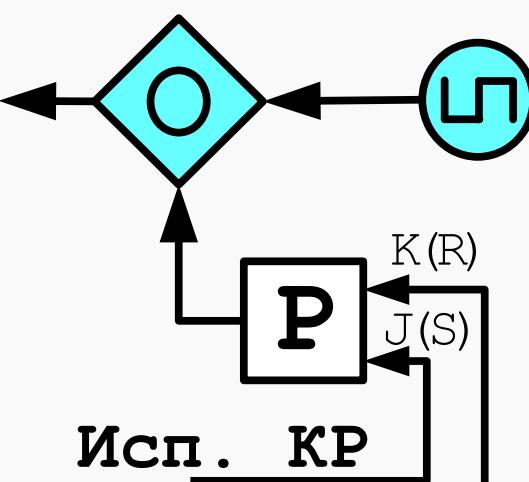


8 младших бит с коммутатора

Регистр состояния

Запрос прерыв. IntRq

Работа/
Останов



Шпаргалка. Вентильные схемы

Бит	Мнемоника	Назначение
Чтение регистров		
00	RDDR	DR (РД) → правый вход АЛУ
01	RDCR	CR (PK) → правый вход АЛУ
02	RDIP	IP (CK) → правый вход АЛУ
03	RDSP	SP (YC) → правый вход АЛУ
04	RDAC	AC (A) → левый вход АЛУ
05	RDBR	BR (БР) → левый вход АЛУ
06	RDPS	PS (PC) → левый вход АЛУ
07	RDIR	KR (KP) → левый вход АЛУ
АЛУ		
08	COMR	Обратный код правого входа АЛУ
09	COML	Обратный код левого входа АЛУ
10	PLS1	Операция A + B + 1 (PLuS 1)
11	SORA	Операция И (Sum OR And)

Шпаргалка. Вентильные схемы

Бит	Мнемоника	Назначение
Управление коммутатором		
12	LTOl	Прямая передача младшего байта
13	LTON	Передача младшего байта в старший
14	HTOl	Передача старшего байта в младший
15	HTON	Прямая передача старшего байта
16	SEXT	Расширение знака младшего байта (sign extend)
17	SHLT	Сдвиг влево (арифметический) (SHift LeFT)
18	SHLO	Передача старого значения С в младший бит (для циклического сдвига влево)
19	SHRT	Сдвиг вправо (арифметический) (SHift Right)
20	SHRF	Переключатель сдвига вправо (для циклического сдвига 15 разряд)
21	SETC	Установка С
22	SETV	Установка V
23	STNZ	Установка N и Z

Шпаргалка. Вентильные схемы

Бит	Мнемоника	Назначение
Чтение регистров		
24	WRDR	АЛУ → DR (РД)
25	WRCR	АЛУ → CR (РК)
26	WRIP	АЛУ → IP (СК)
27	WRSP	АЛУ → SP (YC)
28	WRAC	АЛУ → AC (A)
29	WRBR	АЛУ → BR (БР)
30	WRPS	АЛУ → PS (PC)
31	WRAR	АЛУ → AR (PA)
Работа с памятью		
32	LOAD	Ячейка памяти → DR (РД)
33	STOR	DR (РД) → Ячейка памяти

Шпаргалка. Вентильные схемы

Бит	Мнемоника	Назначение
Организация ввода-вывода		
34	IO	Передача адреса и приказа на ВУ
35	INTS	Предоставление прерывания
Резервные сигналы		
36		Зарезервирован
37		Зарезервирован
Останов БЭВМ		
38	HALT	Останов
Работа с памятью		
39	TYPE	Бит выбора ОМК/УМК

Интерпретатор БЭВМ

- 256 ячеек для хранения микрокоманд, включая резерв
- Содержит горизонтальные микрокоманды
- Оформлено в виде таблицы (см. Методу!), а лучше **java -Dmode=decoder -jar bcomp-ng.jar**

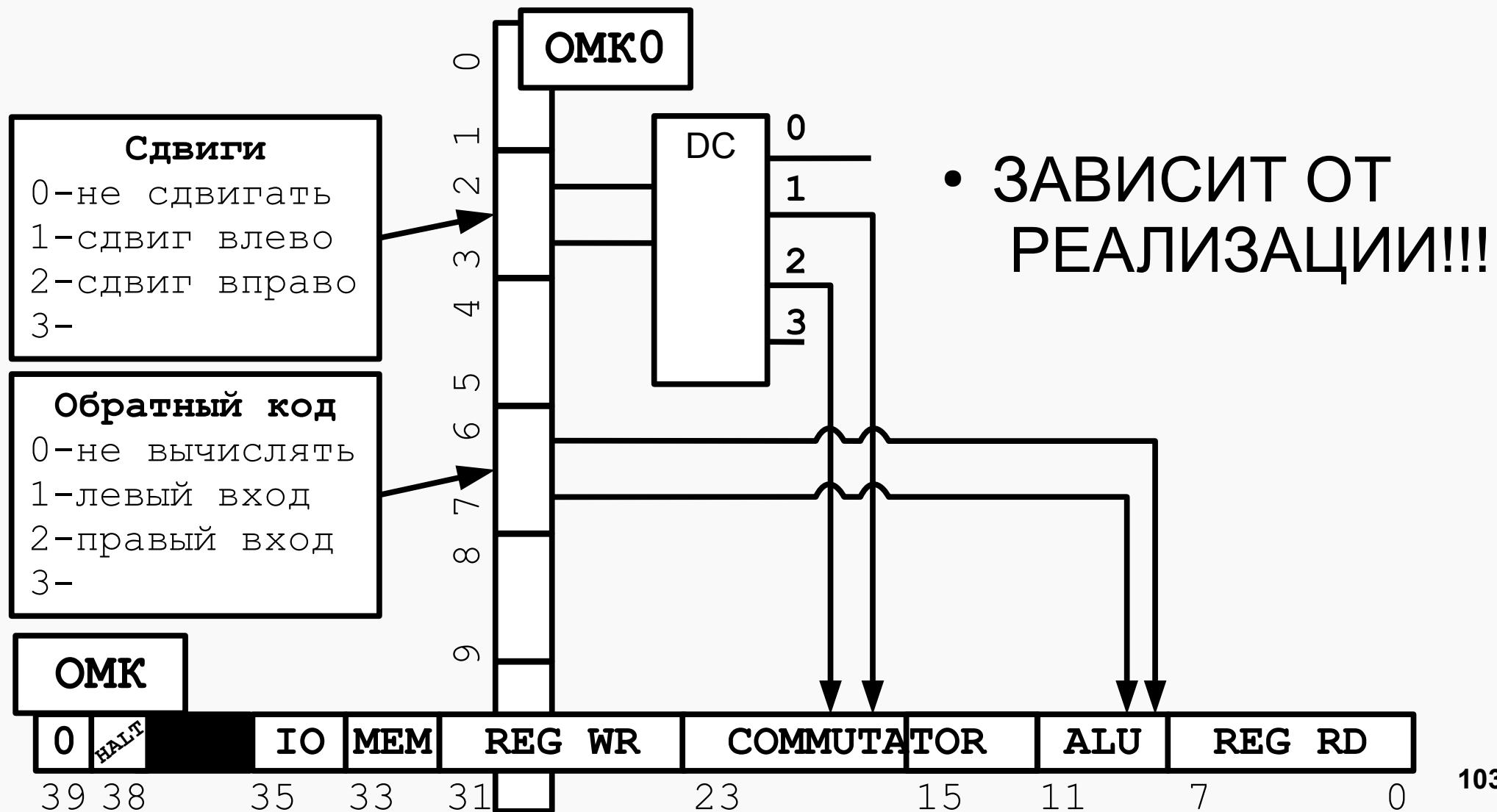
Адр- рес	Горизонтальная микрокоманда	Коментарии	
		Метка	Действие
01	00 A000 9004	INFETCH	IP → BR, AR
02	01 0400 9420		BR + 1 → IP; MEM(AR) → DR
03	00 0200 9001		DR → CR
04	81 0980 4002		if CR(15) = 1 then GOTO CHKBR @ 09

Интерпретатор БЭВМ

- Цикл выборки команд
- Цикл выборки адреса операнда и обработки режимов адресации
- Цикл выборки операнда
- Цикл исполнения
 - Декодирование и выполнение адресных команд
 - Декодирование и выполнение ветвлений
 - Декодирование и выполнение безадресных команд
- Декодирование и выполнение команд ввода-вывода
- Цикл прерывания
- Пультовые операции
- Свободные ячейки для:
 - Арифметической команды
 - Команды перехода
 - Безадресной команды

Вертикальные микрокоманды

- Задача — Сократить место в памяти микрокоманд. В БЭВМ-NG — НЕТ! (пока)



С БЭВМ — все!!!!

