ИТМО ВТ

Основы профессиональной деятельности
Часть третья (не последняя).

Клименков С.В.

2018-2019 уч. год

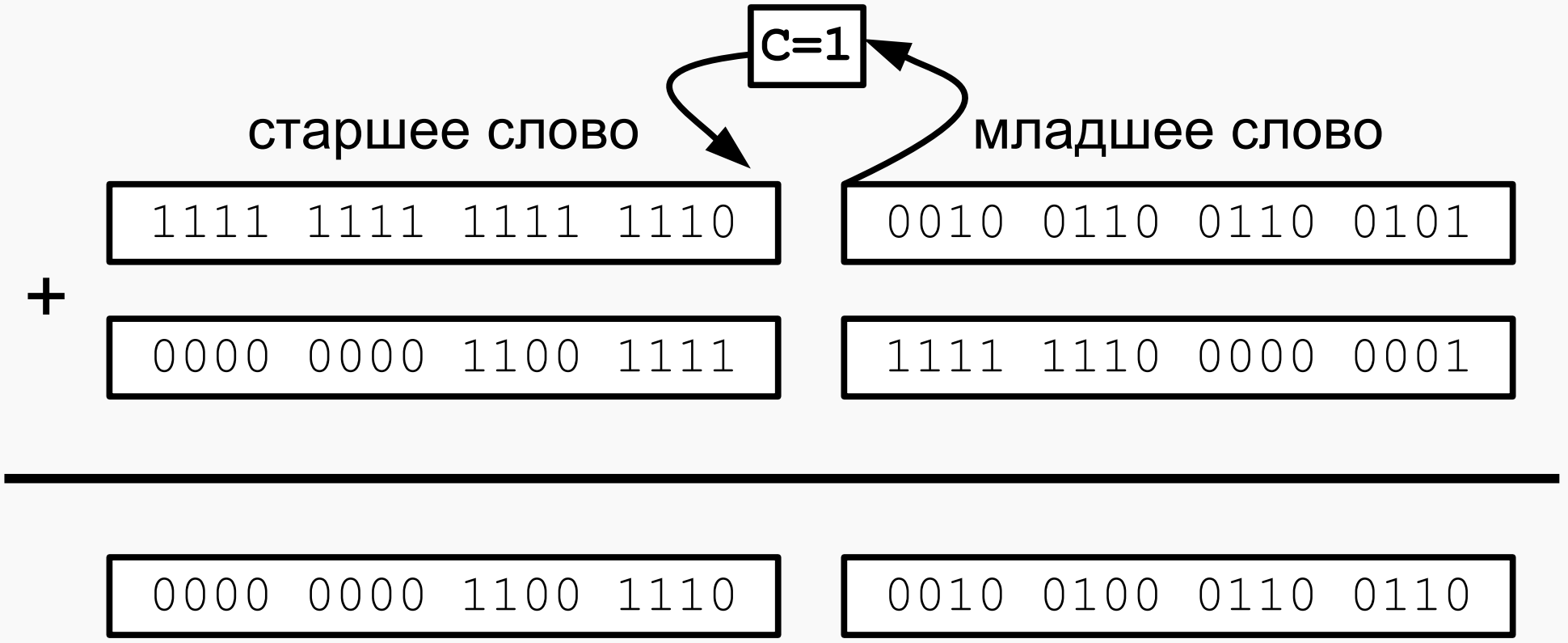
v.1.23 от 01.03.2019

ИТМО ВТ

Выполнение вычислений

1

БЭВМ: 32-х разрядные числа



fffe2665 + cffe01 = ce2466

-121243 + 13630977 = 13509734

ИТМО ВТ

Суммирование 32-х разрядных чисел

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	F200	**CLA**	
011	4019	**ADD 19**	Младшее слово 1-го числа
012	401B	**ADD 1B**	Младшее слово 2-го числа
013	301D	**MOV 1D**	Сохранение младшего слова суммы
014	F200	**CLA**	
015	501A	**ADC 1A**	Старшее слово 1-го числа, перенос между словами
016	401C	**ADD 1C**	Старшее слово 2-го числа
017	301E	**MOV 1E**	Сохранение старшего слова суммы
018	F000	**HLT**	Останов
019	2665	X	Две ячейки 1-го числа
01A	FFFE	X	
01B	FE01	Y	Две ячейки 2-го числа
01C	00CF	Y	
01D	2466	R	Две ячейки результата
01E	00CE	R	

fffe2665 + cffe01 = ce2466

ИТМО ВТ

Изменение знака 16-ти разрядного числа

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	F200	**CLA**	
011	4016	**ADD 16**	X в аккумуляторе
012	F400	**CMA**	Вычисление дополнения (инверсия битов числа)
013	F800	**INC**	Инкремент
014	3017	**MOV 17**	Сохранение результата
015	F000	**HLT**	
016	0002	X	X
017	FFFE	R	R

R = -X

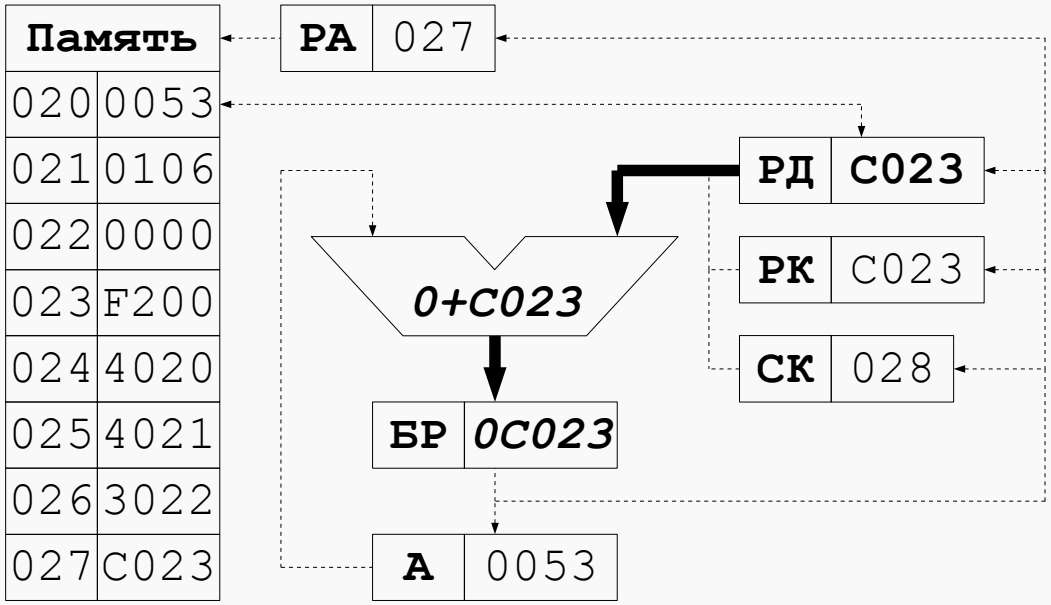
ИТМО ВТ

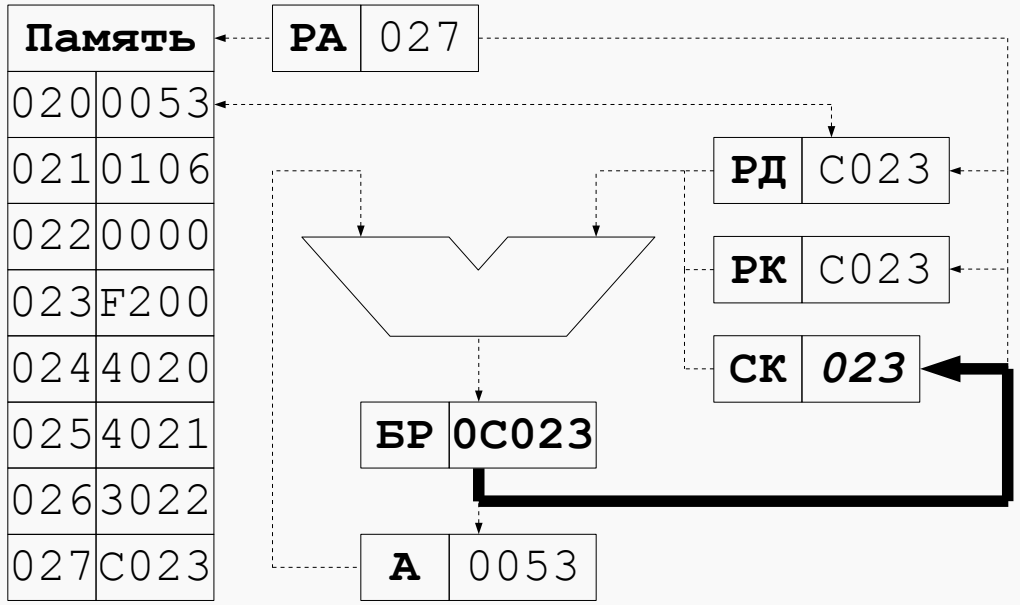
Изменение знака 32-х разрядного числа

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	F200	**CLA**	
011	401E	**ADD 1E**	Старшее слово X в аккумуляторе
012	F400	**CMA**	Вычисление дополнения (инверсия битов числа)
013	3020	**MOV 20**	Сохранение старшего слова R
014	F200	**CLA**	
015	401D	**ADD 1D**	Младшее слово X в аккумуляторе
016	F400	**CMA**	Вычисление дополнения (инверсия битов числа)
017	F800	**INC**	Инкремент (получение доп. кода младшего слова)
018	301F	**MOV 1F**	Сохранение младшего слова
019	F200	**CLA**	
01A	5020	**ADC 20**	Добавление возможного переноса к старшему слову
01B	3020	**MOV 20**	Сохранение старшего слова
01C	F000	**HLT**	
01D	2665	X	X = -121243
01E	FFFE	X	**R = -X**
01F	D99B	R	R = 121243
020	0001	R	

Управление вычислительным процессом

Переход, если перенос	BCS M	8XXX	Если (C) = 1, то M ➔ СК
Переход, если плюс	BPL M	9XXX	Если (N)= 0, то M ➔ СК
Переход, если минус	BMI M	AXXX	Если (N) = 1, то M ➔ СК
Переход, если ноль	BEQ M	BXXX	Если (Z) = 1, то M ➔ СК
Безусловный переход	BR M	CXXX	M ➔ СК





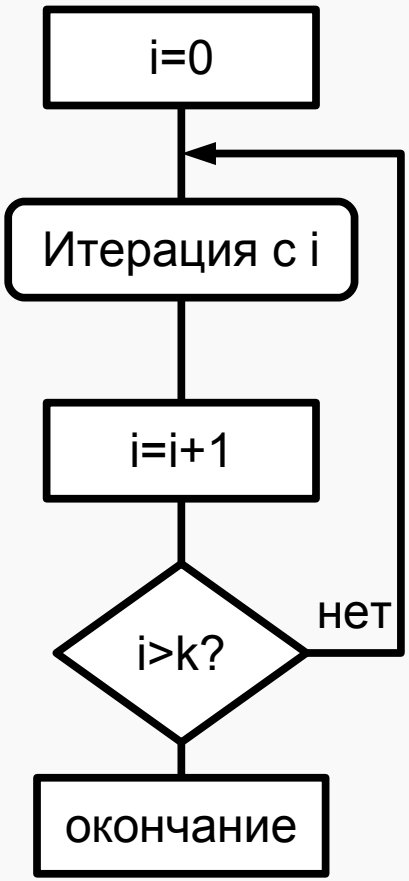
Программа вычисления модуля числа

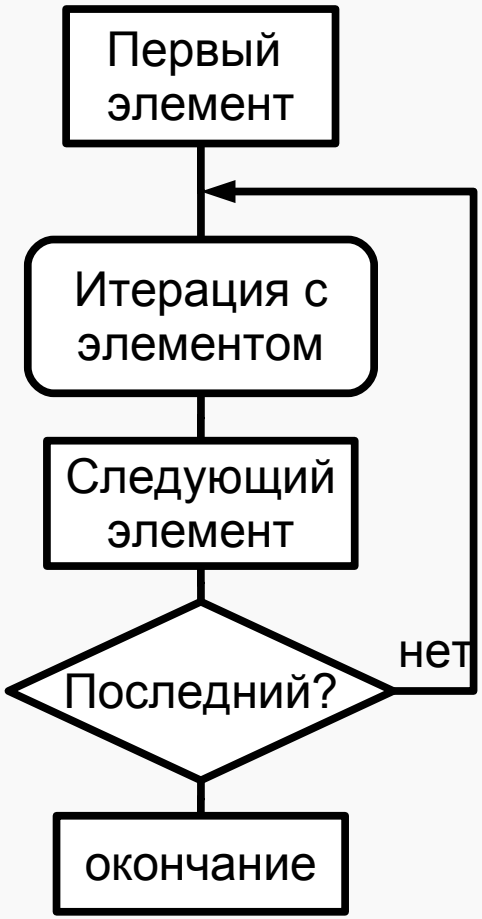
Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	F200	**CLA**	
011	4017	**ADD 17**	X в аккумуляторе
012	9015	**BPL 15**	Если X>=0 то переход к адресу 15
013	F400	**CMA**	Вычисление дополнения (инверсия битов числа)
014	F800	**INC**	Инкремент
015	3018	**MOV 18**	Сохранение результата
016	F000	**HLT**	
017	FFFE	X	X
018	0002	R	R = |X|

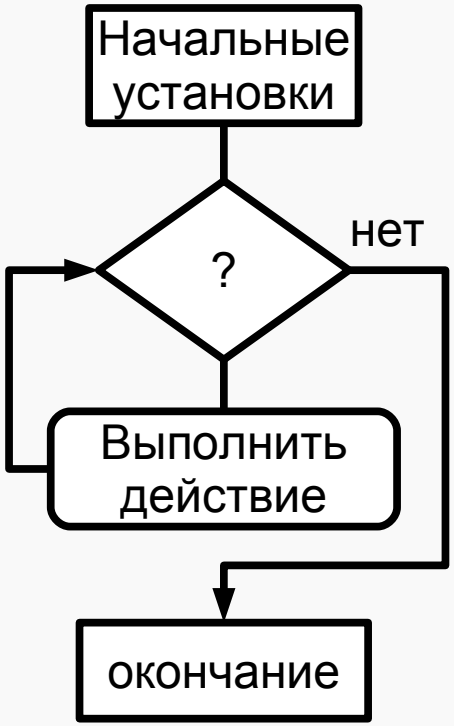
R = |X|

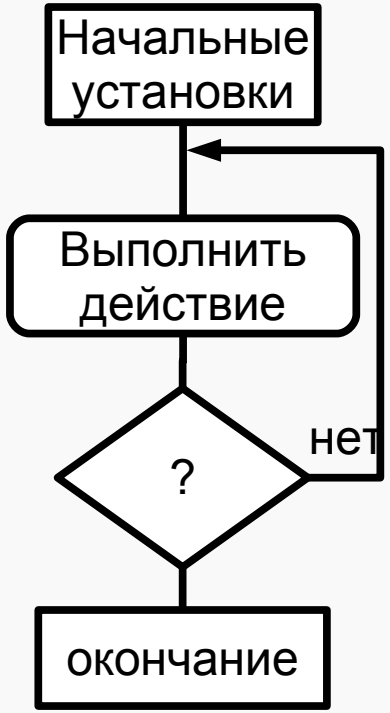
Отступление: Циклические программы

- for, foreach, while , do-while









R=50Y (вар 1)

ОДЗ!

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
005 006 007 008	0042 0000 0032 0000	Y Z M C	Множимое Ячейка для накопления и хранения результата Множитель $50=(32)_{16}$ Счетчик циклов
010 011 012 013	F200 4006 4005 3006	**CLA** **ADD 6** **ADD 5** **MOV 6**	К промежуточному результату, находящемуся в ячейке 6, добавляется ещё одно значение множимого Y
014 015 016 017	F200 4008 F800 3008	**CLA** **ADD 8** **INC** **MOV 8**	Содержимое счетчика циклов C увеличивается на 1, а его копия пока сохраняется в аккумулятор
18 19 1A	6007 A010 F000	**SUB 7** **BMI 10** **HLT**	Если C < M, то продолжаем добавление Y к R Останов, когда все 50 операций выполнены

ИТМО ВТ

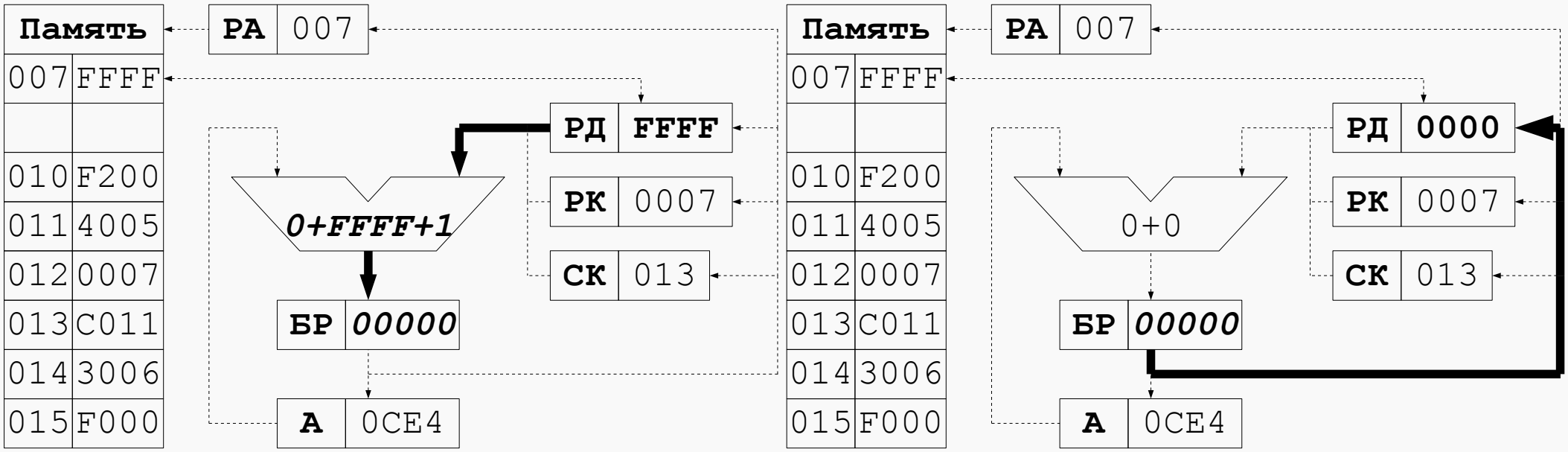
R=50Y (ISZ)

Приращение и пропуск	ISZ M	0XXX	M + 1 ➔ M, если (M) >= 0,то (CK) + 1 ➔ CK

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

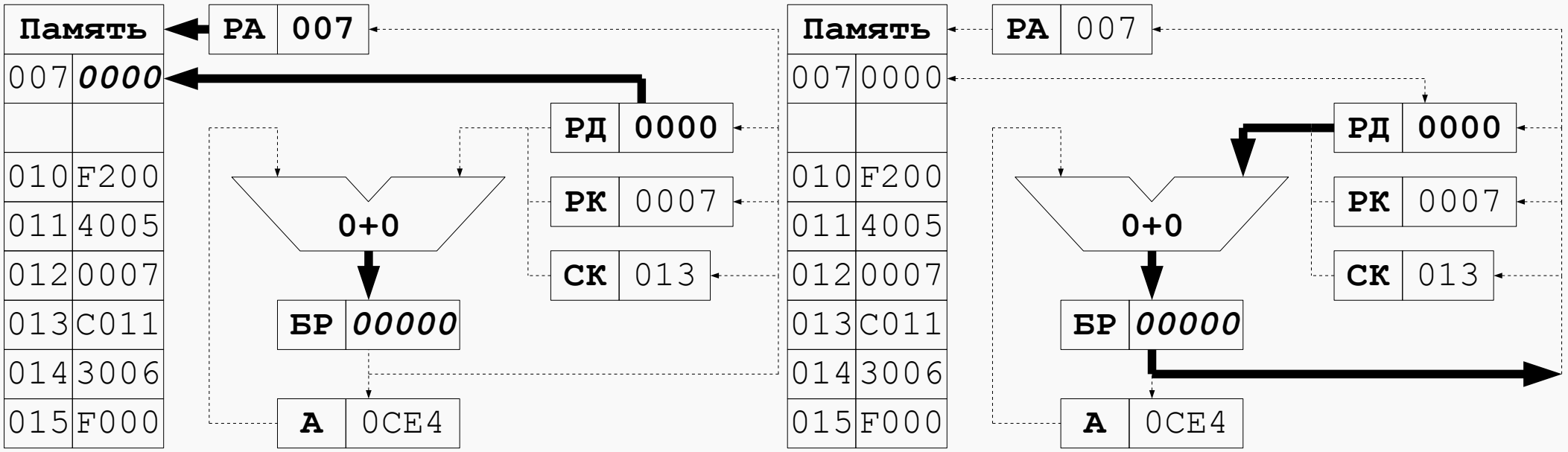
Адрес	Содержимое		Комментарии
	Код	Мнемоника	
005 006 007	0042 0000 FFCE	Y R M	Множимое Ячейка для накопления и хранения результата Отрицательное значение множителя (-50)
010 011 012 013	F200 4005 0007 C011	**CLA** **ADD 5** **ISZ 7** **BR 11**	К содержимому аккумулятора добавляется Y M наращивается на 1, в случае если M<0 выполняется переход на 11 адрес. Если M=0, то BR пропускается.
014 015	3006 F000	**MOV 6** **HLT**	Содержимое счетчика циклов C увеличивается на 1, а его копия пока сохраняется в аккумулятор

Цикл исполнения ISZ



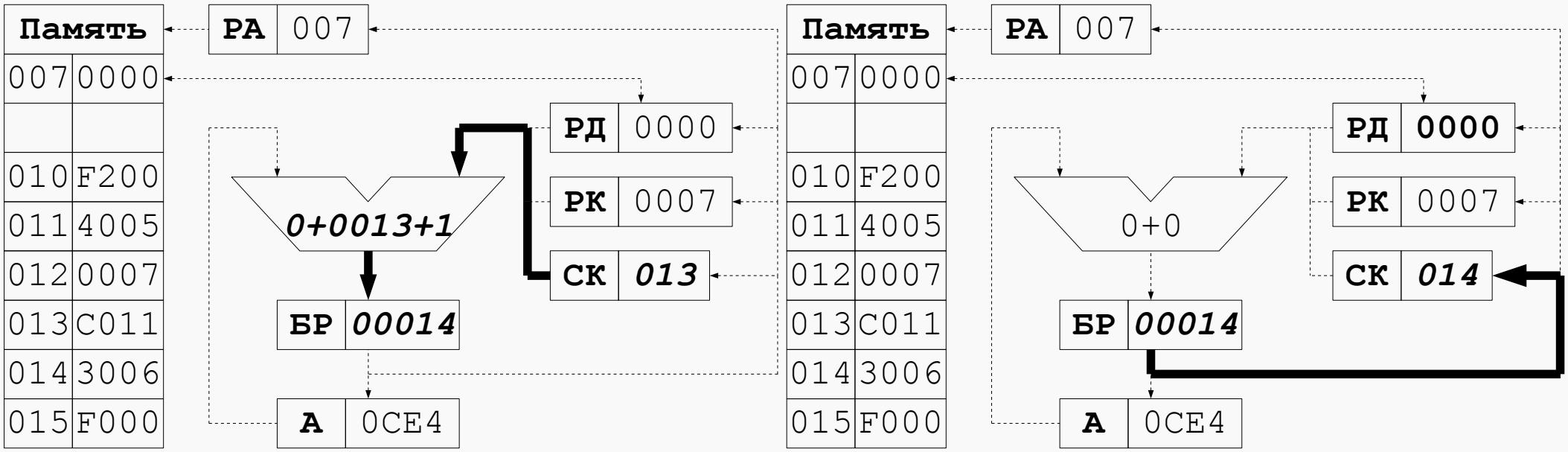
- $42_{16}=66_{10}$; $50*66=3300_{10}=CE4_{16}$
- Перед циклом непосредственного исполнения в РД находится содержимое ячейки (FFFF) адресной части (007) команды ISZ
- Содержимое РД увеличивается на 1 и сохраняется через БР обратно в РД

Цикл исполнения ISZ



- Содержимое РД записывается в память, БР обнуляется
- 15 бит РД подается на схему сравнения, где проверяется на 1

Цикл исполнения ISZ



- Так как 15 бит РД равен 0, то содержимое СК увеличивается на 1

R=50Y («китайский», правильный)

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
005	0078	Y	Множимое
006	0000	R	Ячейка для накопления и хранения результата
007	0000	R’	Промежуточный результат Y*16
010	F200	**CLA**	
011	4005	**ADD 5**	Y
012	F600	**ROL**	Y*2 после ADD C=0. В CLC нет необходимости
013	3006	**MOV 6**	Сохраним в ячейке R
014	F300	**CLC**	
015	F600	**ROL**	Y*4
016	F300	**CLC**	
017	F600	**ROL**	Y*8
018	F300	**CLC**	
019	F600	**ROL**	Y*16
01A	3007	**MOV 7**	Сохраним в ячейке R’
01B	F300	**CLC**	
01C	F600	**ROL**	Y*32
01D	4007	**ADD 7**	Добавим к аккумулятору R и R’, таким образом, что
01E	4006	**ADD 6**	R=32Y+16Y+2Y=(32+16+2)Y
01F	3006	**MOV 6**	Сохраняем результат
020	F000	**HLT**	

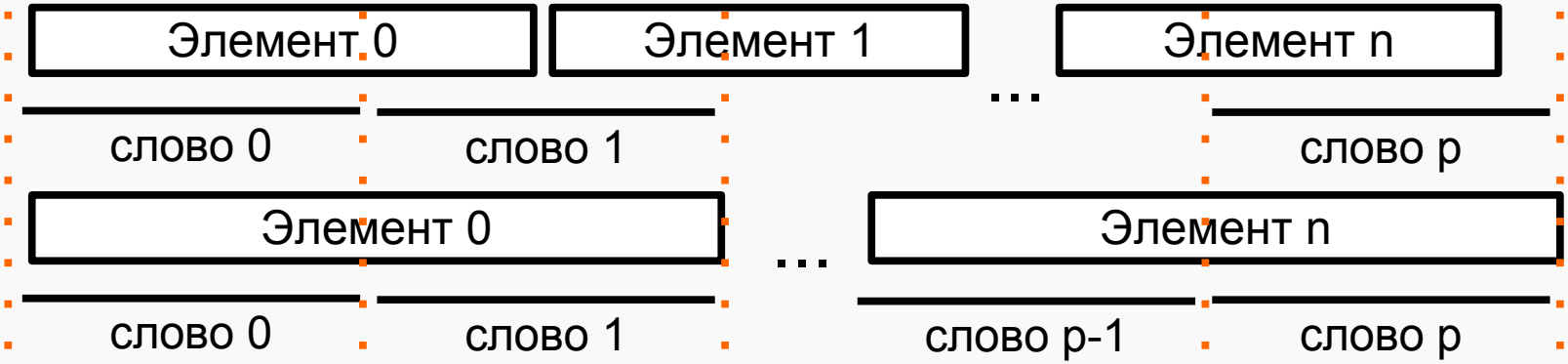
$$\mathbf{50_{10}=32_{16}=00110010_{2}}$$

Представление одномерных массивов данных

1. Элемент массива занимает ровно слово

2. Элемент меньше слова

3. Элемент больше слова



ИТМО ВТ

Суммирование элементов массива (переадресация)

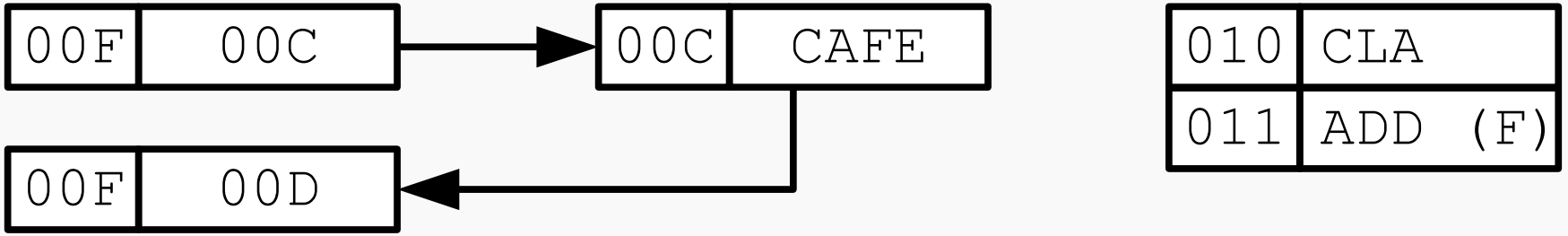
Адрес	Содержимое		Комментарии
	Код	Мнемоника	
005 006	0000 FFE0	S C	Ячейка, отведенная для накопления результата. Отрицательное число элементов массива (-32)
010 ... 02F			Элементы массива
030 031 032 033	F200 4005 4010 3005	**CLA** **ADD 5** **ADD ?** **MOV 5**	Предыдущий результат (S) складывается с ячейкой, указанной в коде команды по адресу 32 и записывается в S
034 035 036 037 038 039 03A	F200 4032 F800 3032 0006 C030 F000	**CLA** **ADD 32** **INC** **MOV 32** **ISZ 6** **BR 30** **HLT**	 Взять код команды по адресу 32 Увеличить его на 1 Записать в ячейку 32. Элементы закончились? Нет – переход на 30 адрес Да - останов

Косвенная адресация и индексные ячейки

1. Прямая адресация

2. Косвенная адресация

3. Косвенная автоинкрементная адресация (яч. 8-F)



ИТМО ВТ

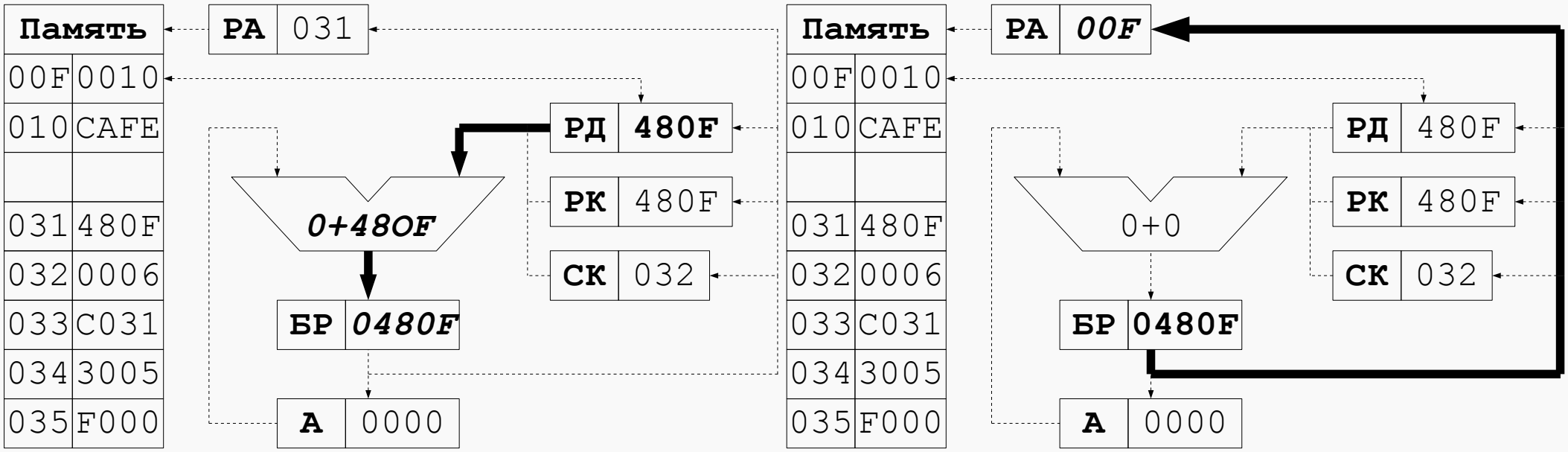
Суммирование (косвенная адресация)

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
005	0000	S	Ячейка, отведенная для накопления результата.
006	FFE0	C	Отрицательное число элементов массива (-32)
007	0010	I	Текущий элемент массива
010 ... 02F			Элементы массива
030	F200	**CLA**	
031	4807	**ADD (7)**	Акумулятор содержит следующий элемент массива
032	0007	**ISZ 7**	Увеличить адрес элемента на 1 не изменяя акк.
033	F100	**NOP**	Эта команда никогда не будет выполнена
034	0006	**ISZ 6**	Мы сложили все элементы?
035	C031	**BR 31**	Нет – продолжим складывать
036	3005	**MOV 5**	Результат в ячейке 5
037	F000	**HLT**	

Суммирование (косвенная автоинкрементная адресация)

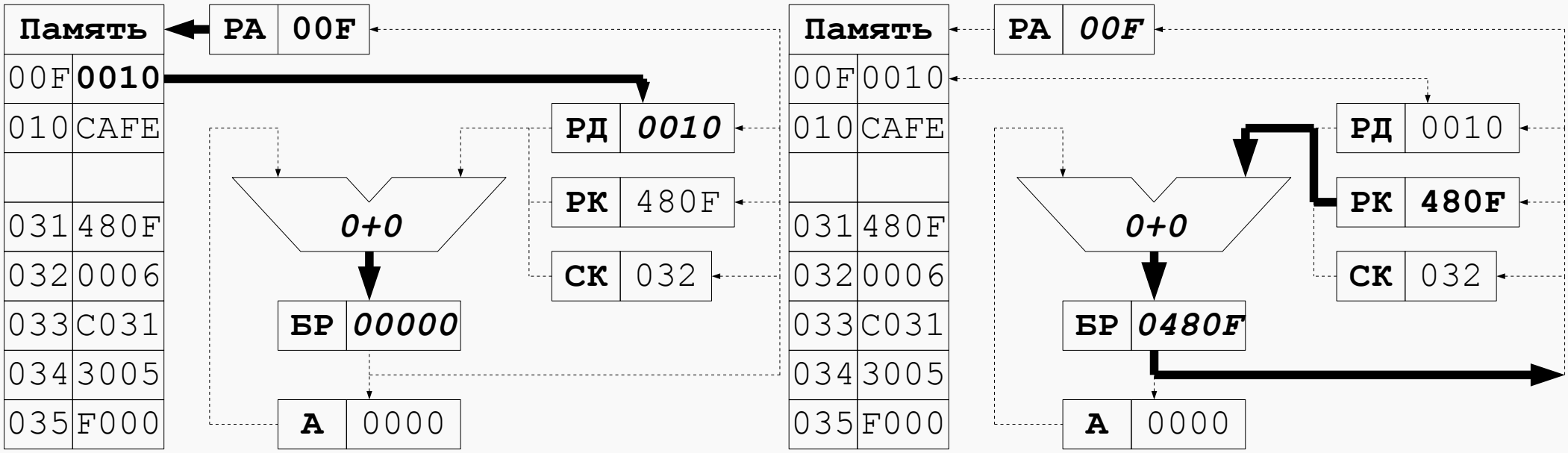
Адрес	Содержимое		Комментарии
	Код	Мнемоника	
005 006 ... 00F	0000 FFE0 0010	S C I	Ячейка, отведенная для накопления результата. Отрицательное число элементов массива (-32) Текущий элемент массива
010 ... 02F			Элементы массива
030 031 032 033 034 035	F200 480F 0006 C031 3005 F000	**CLA** **ADD (F)** **ISZ 6** **BR 31** **MOV 5** **HLT**	Акумулятор содержит следующий элемент массива Содержимое ячейки F увеличивается на 1 Мы сложили все элементы? Нет – продолжим складывать Результат в ячейке 5

Цикл выборки адреса



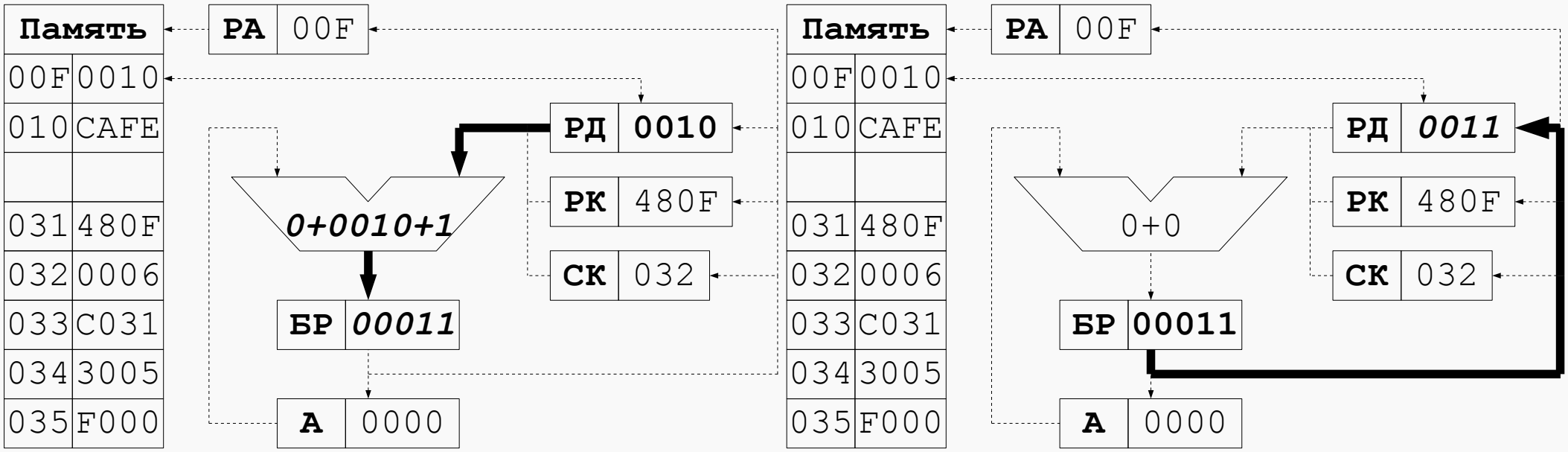
- Цикл выборки команд завершился увеличением СК записью в РД и РК кода `480F` исполняемой команды `ADD (F)`
- Далее младшие 11 разрядов содержимое РД пересылается в РА через БР

Цикл выборки адреса



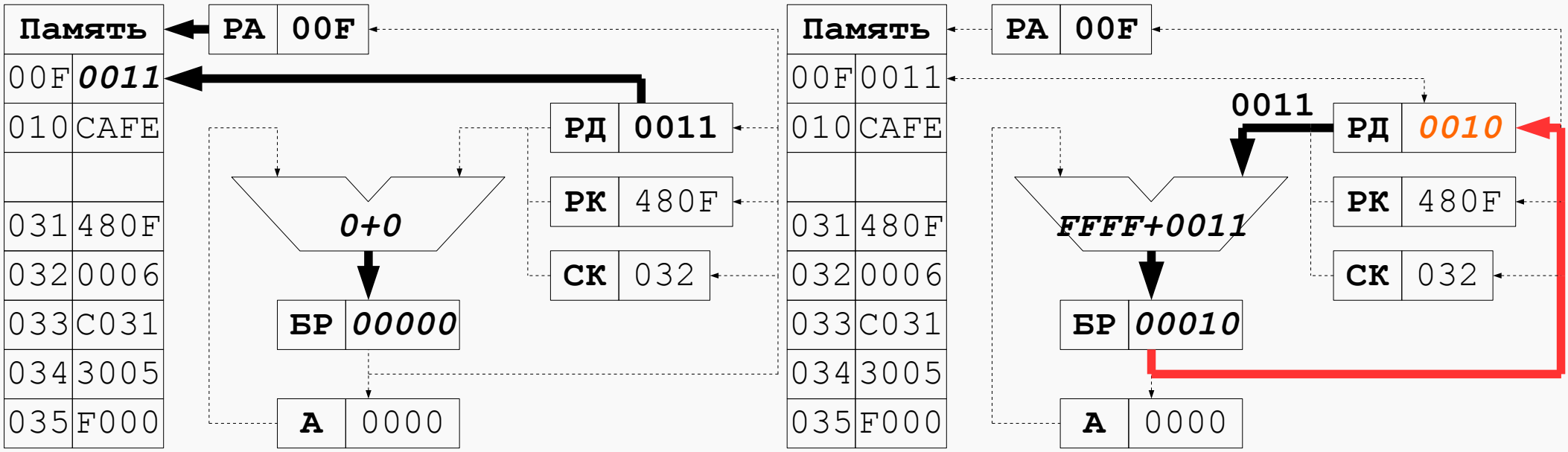
- Содержимое по адресу в РА загружается в РД
- Далее последовательно проверяются младшие разряды регистра команд на соответствие маске 000 0000 1XXX, т.е является ли ячейка индексной (008-00F)

Цикл выборки адреса



- Если ячейка индексная, то ее содержимое, находящееся в РД необходимо увеличить на 1 и поместить в БР
- Увеличенное содержимое из БР необходимо переслать в РД

Цикл выборки адреса



- Новое, увеличенное на 1, содержимое индексной ячейки записывается в память
- Необходимо вернуть предыдущее значение для дальнейшей выборки операнда, поэтому содержимое РД уменьшается на 1 и записывается в БР, а в **следующем** такте в РД (РД+COM(0)=РД-1 → **БР→ РД**)


ПОБЕДА

ИТМО ВТ

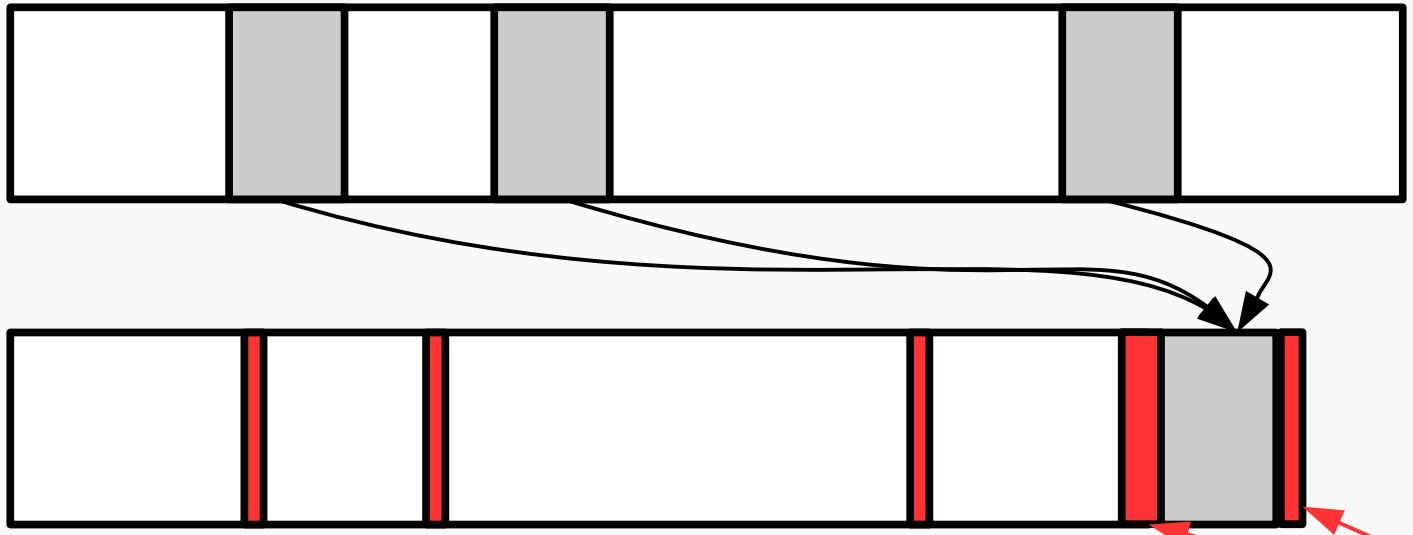
Подпрограммы

2

ИТМО ВТ

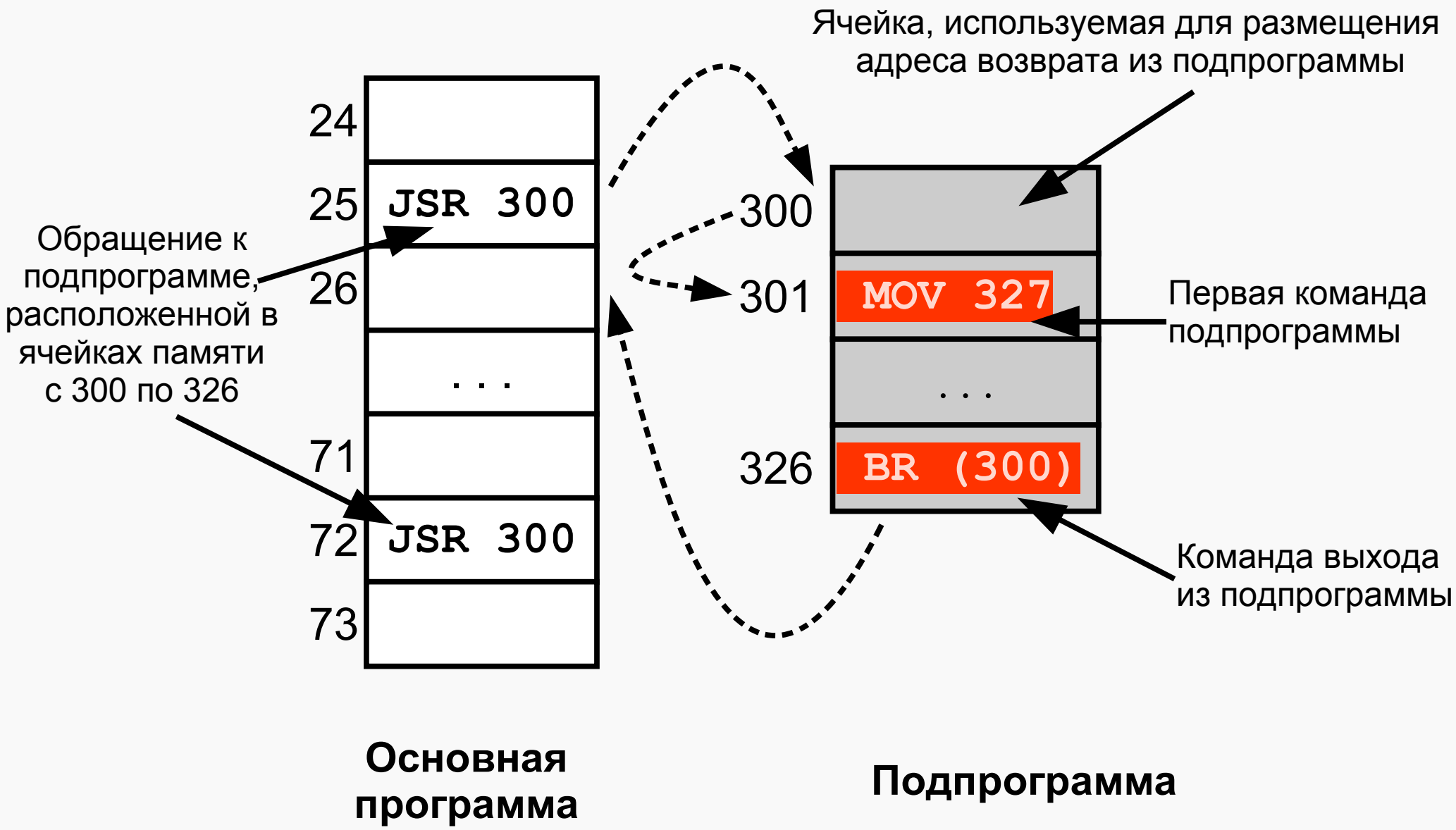
Мотвивация

- 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



ИТМО ВТ

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

ИТМО ВТ

Подпрограмма вычисления удвоенного модуля для чисел в ячейках 58,63,71 с размещением результатов в ячейках 74,77,82

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
010	F200	**CLA**	Первое число
011	4058	**ADD 58**	Аргумент в 58
012	2030	**JSR 30**	
013	3074	**MOV 74**	Результат в 74
014	F200	**CLA**	Второе число
015	4063	**ADD 63**	Аргумент в 63
016	2030	**JSR 30**	
017	3077	**MOV 77**	Результат в 77
18	F200	**CLA**	Третье число
19	4071	**ADD 71**	Аргумент в 71
1A	2030	**JSR 30**	
1B	3082	**MOV 82**	Результат в 82
1C	F000	**HLT**	

Подпрограмма вычисления R=2|X|

Адрес	Содержимое	
	Код	Мнемоника
030	0000	Адрес возв.
031	9034	**BPL 34**
032	F400	**CMA**
033	F800	**INC**
034	F300	**CLC**
035	F600	**ROL**
036	C830	**BR (30)**

Комплекс программ: передача через адреса ячеек памяти

Подпрограмма вычисления R=2|X|

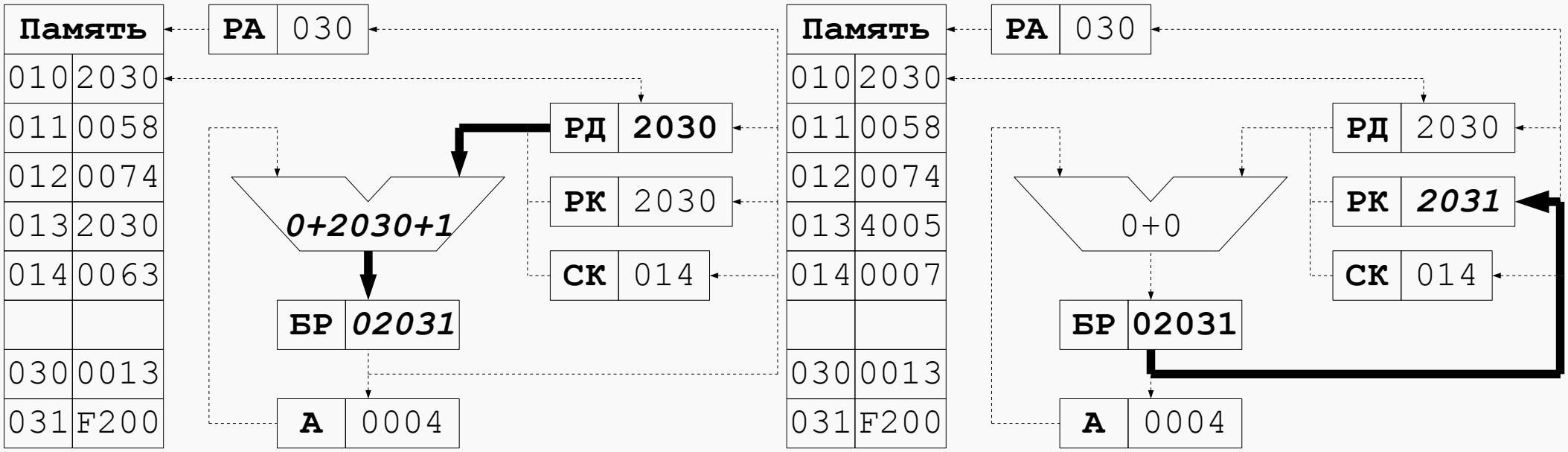
Адрес	Содержимое	
	Код	Мнемоника
030	0000	Адрес возв.
031	F200	**CLA**
032	4830	**ADD (30)**
033	3035	**MOV 35**
034	0030	**ISZ 30**
035	0000	Адрес Xi
036	F200	**CLA**
037	4830	**ADD (30)**
038	303A	**MOV 3A**
039	0030	**ISZ 30**
03A	0000	Адрес Ri
03B	F200	**CLA**
03C	4835	**ADD (35)**
03D	9040	**BPL 40**
03E	F400	**CMA**
03F	F800	**INC**
040	F300	**CLC**
041	F600	**ROL**
042	383A	**MOV (3A)**
043	C830	**BR (30)**

Комментарии
По адресу возврата выбрать адрес Xi и записать в яч. 035 Адрес возврата++ СК>0 всегда!
По адресу возврата выбрать адрес Ri и записать в яч. 03A Адрес возврата++ СК>0 всегда!
Выбрать Xi
Xi<0, меняем знак
Умножаем на 2
Записываем Ri
Возврат

Основная программа

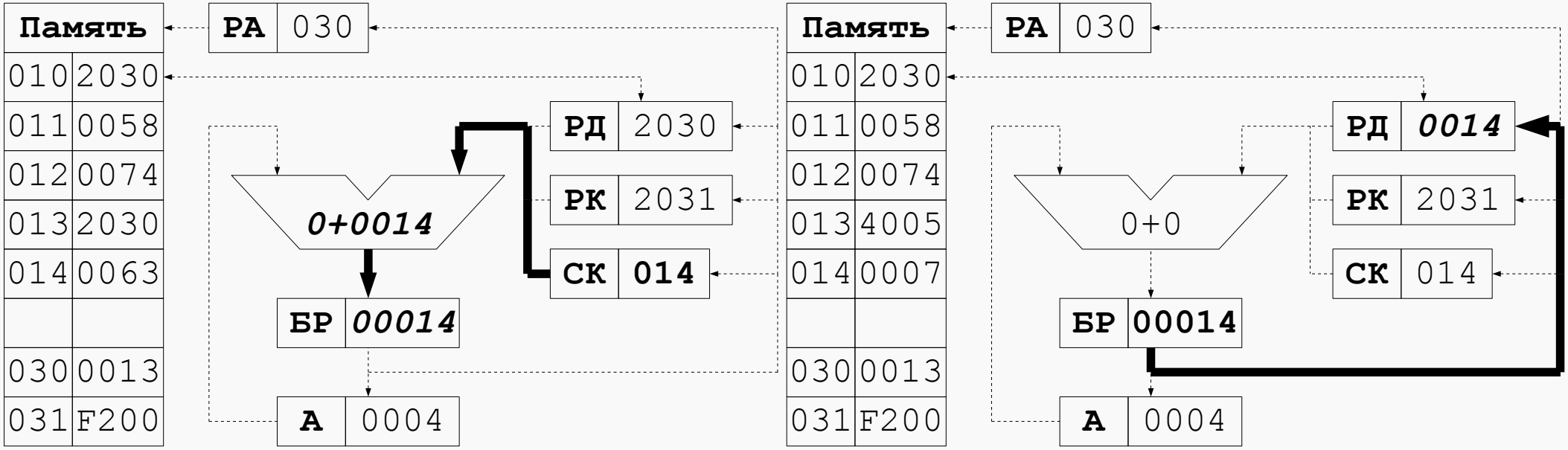
Адрес	Содержимое	
	Код	Мнемоника
010	2030	**JSR 30**
011	0058	Адрес X1
012	0074	Адрес R1
013	2030	**JSR 30**
014	0063	Адрес X2
015	0077	Адрес R2
016	2030	**JSR 30**
017	0071	Адрес X3
018	0082	Адрес R3
019	F000	**HLT**

Цикл исполнения JSR



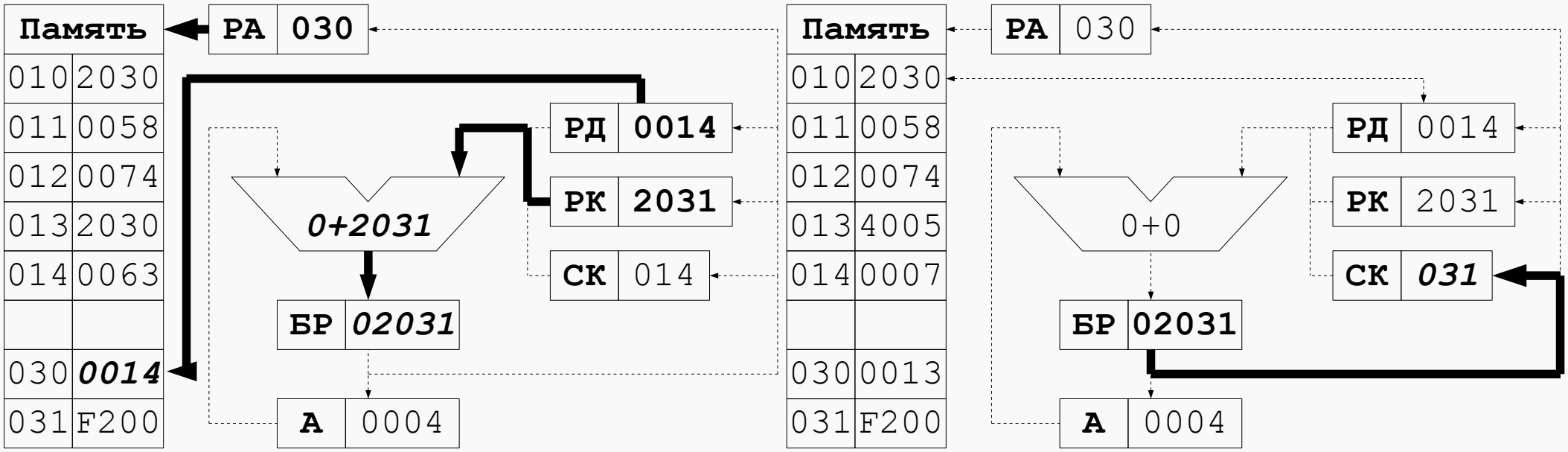
- Окончание декодирования адресной команды записывает в РА адресную часть команды
- Содержимое РД увеличивается на 1 и через БР помещается на временное хранение в РК, который не будет больше использоваться в декодировании

Цикл исполнения JSR



- Адрес возврата из подпрограммы (014 — содержимое СК) записывается через БР в РД

Цикл исполнения JSR



- Содержимое адреса возврата записывается в первую ячейку подпрограммы, одновременно РК переписывается в БР
- Младшие 11 бит содержимого БР записывается в СК, происходит переход к первой команде подпрограммы

ИТМО ВТ

Цикл исполнения BR (адр)

Чем отличается от
цикла исполнения **`BR адр`**
из лекции 5?

ИТМО ВТ

Рекурсивность

- Рекурсивность — способность подпрограммы вызывать саму себя.
  - Подсчет факториала

```
f (n) {
    if (n==0) return 1;
    return n*f(n-1);
}
```

- В БЭВМ?

Адрес	Содержимое	
	Код	Мнемоника
030	0000	Адрес возв.
031	9034	**BEQ 37**
032	F500	**DEC**
033	2030	**JSR 30**
034		...
040	C830	**BR (30)**

ИТМО ВТ

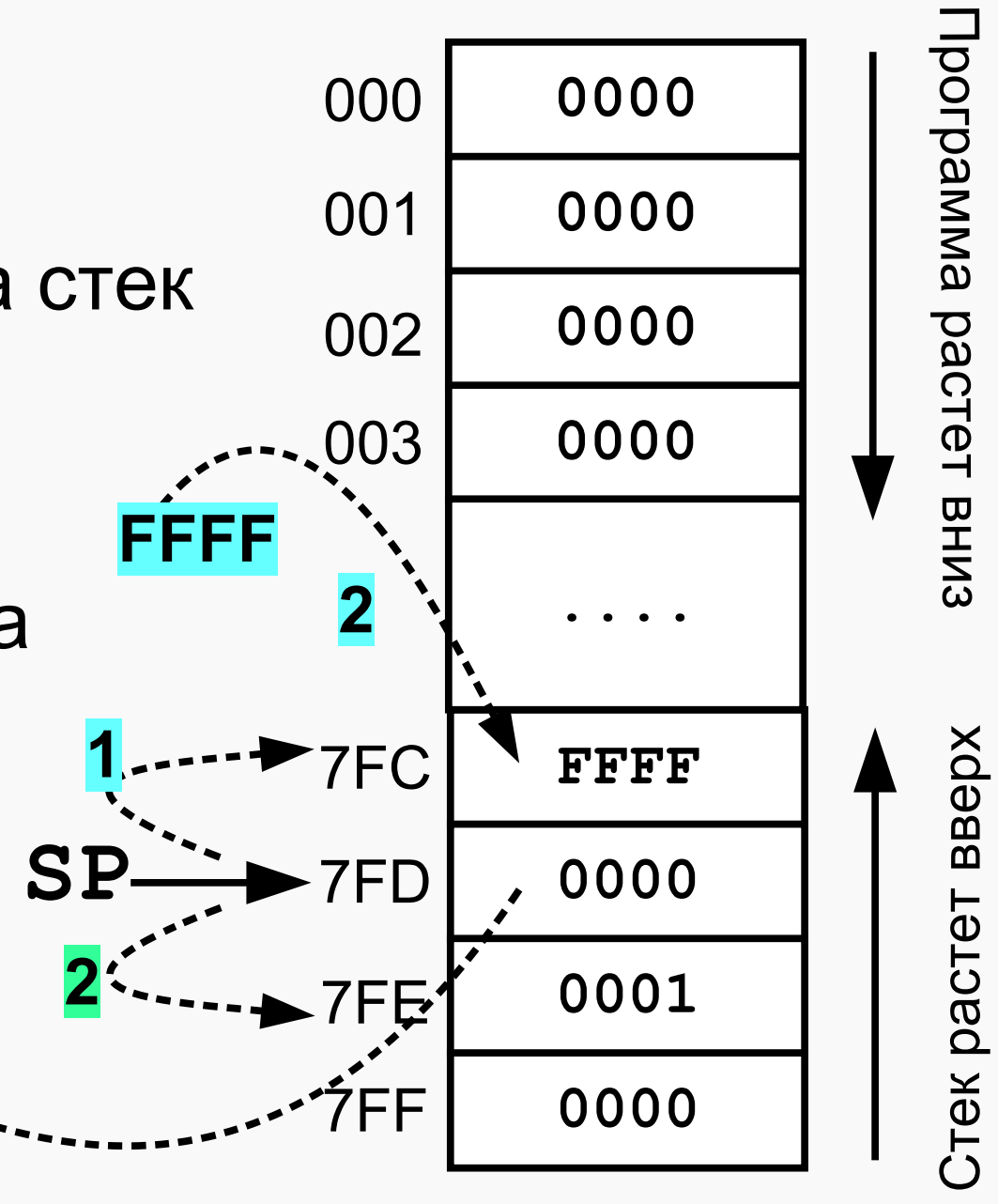
Реентерабельность R=50Y

- Реентерабельность — способность программы быть запущенной несколько раз

Адрес	Содержимое		Комментарии
	Код	Мнемоника	
005 006 007 008	0042 0000 0000 FFCE	Y R M Мнач	Множимое Ячейка для накопления и хранения результата Текущее (цикловое) значение множетеля Начальное отрицательное значение множителя (-50)
00C 00D 00E 00F	F200 3006 4008 3007	**CLA** **MOV 6** **ADD 8** **MOV 7**	 Очистка ячейки для накопления результата Инициализация текущего множителя начальным значением
010 011 012 013	F200 4005 0007 C011	**CLA** **ADD 5** **ISZ 7** **BR 11**	К содержимому аккумулятора добавляется Y M наращивается на 1, в случае если M<0 выполняется переход на 11 адрес. Если M=0, то BR пропускается.
014 015	3006 F000	**MOV 6** **HLT**	Содержимое счетчика циклов C увеличивается на 1, а его копия пока сохраняется в аккумулятор

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)

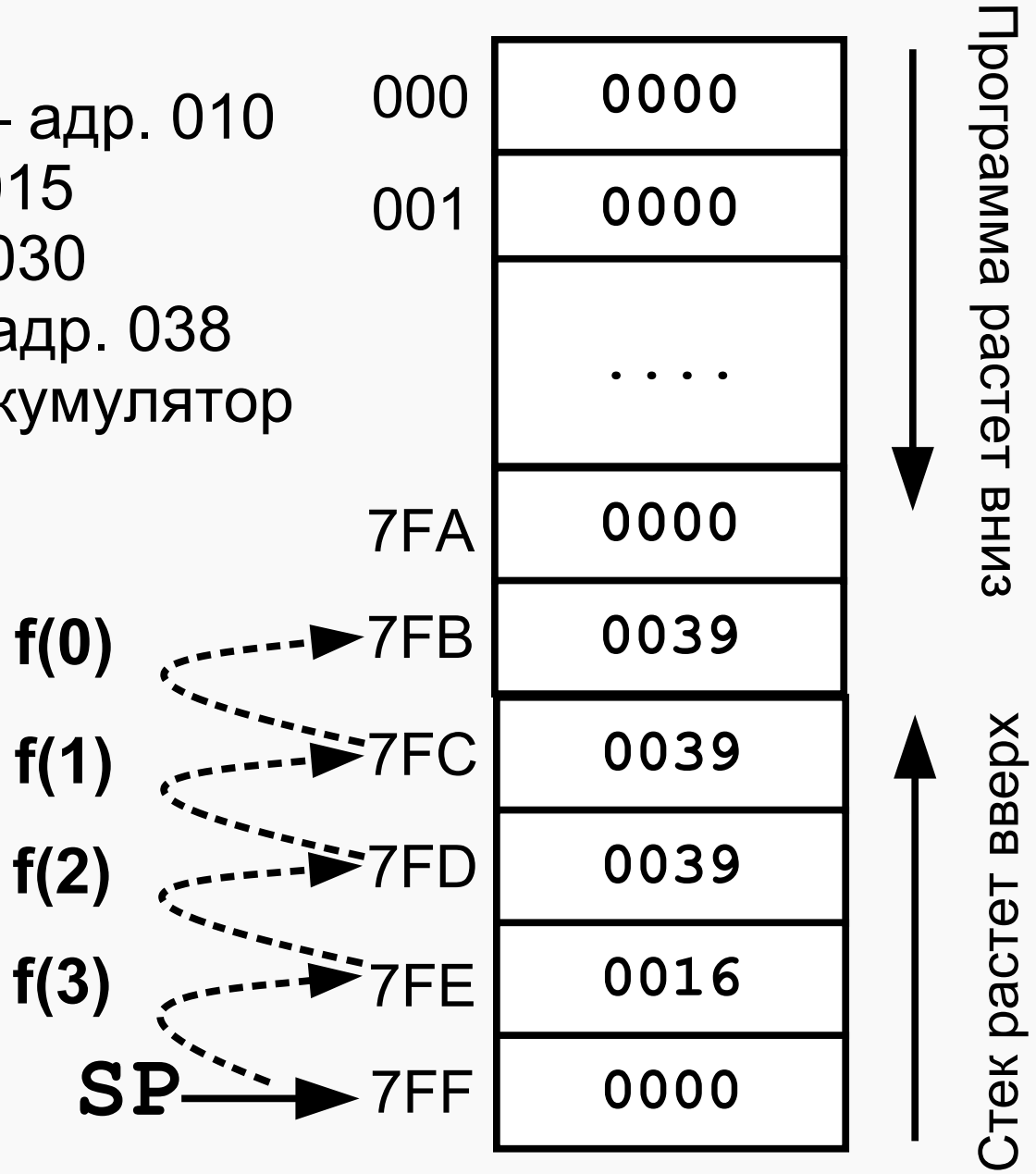


Вызов подпрограмм с использованием стека

Основная программа — адр. 010
Первый вызов — адр. 015
Подпрограмма — адр. 030
Рекурсивный вызов — адр. 038
Параметры — через аккумулятор

```
f (n) {
    if (n==0) return 1;
    return n*f(n-1);
}
```

- f(3)

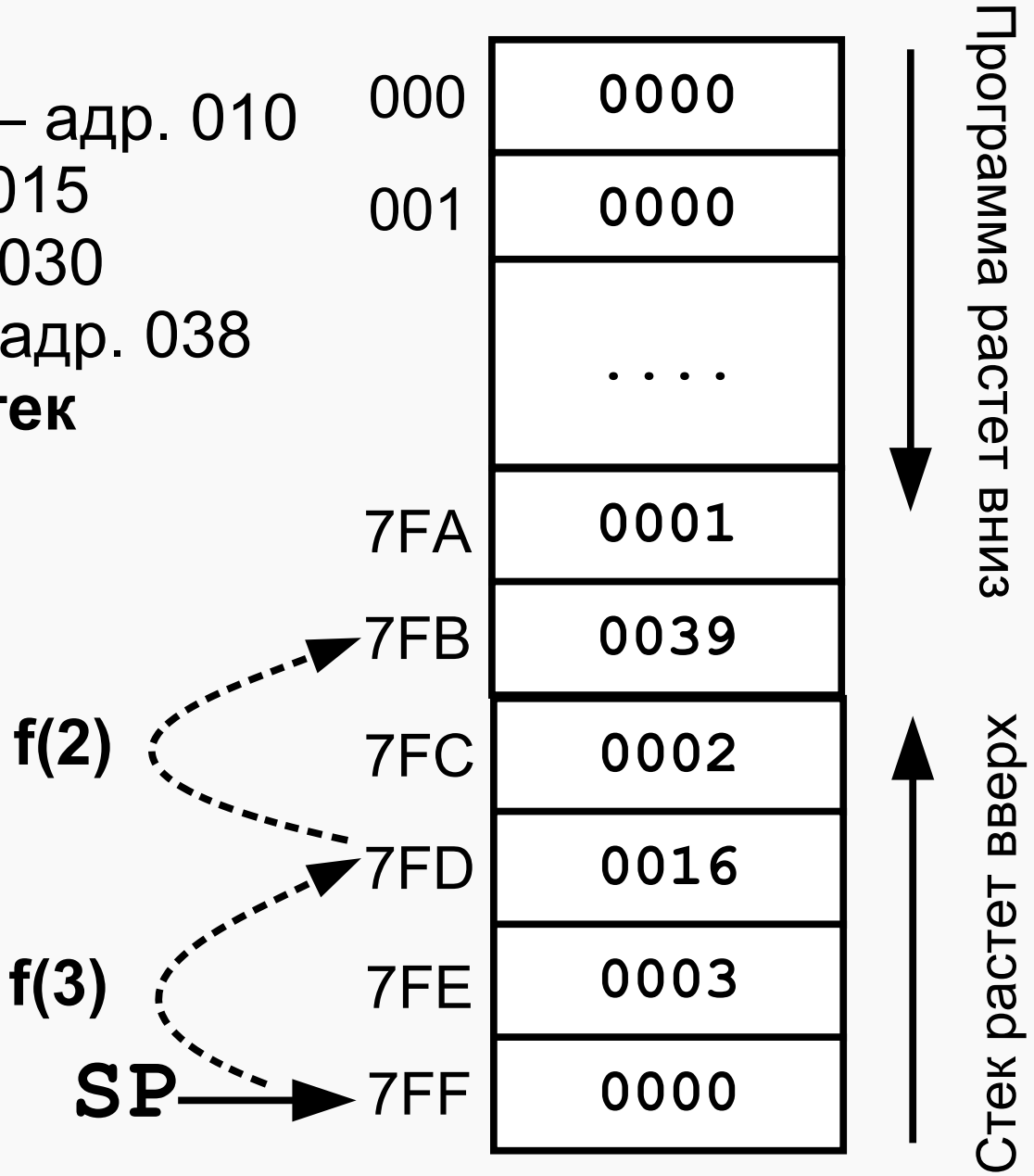


Передача параметров с использованием стека

Основная программа — адр. 010
Первый вызов — адр. 015
Подпрограмма — адр. 030
Рекурсивный вызов — адр. 038
Параметры — через **стек**

```
f (n) {
    if (n==0) return 1;
    return n*f(n-1);
}
```

- f(3)
  - push A
  - call f
  - pop A



ИТМО ВТ

Ассемблер БЭВМ

Назначение	Синтаксис	Пример использования
Размещение в памяти	ORG адрес	ORG 10
Прямая адресация	[метка:] МНЕМОНИКА АРГУМЕНТ	MOV R
Косвенная адресация	[метка:] МНЕМОНИКА (АРГУМЕНТ)	ADD (K)
Безадресная команда	[метка:] МНЕМОНИКА	BEGIN: CLA
Команда ввода-вывода	[метка:] МНЕМОНИКА АДРЕСВУ	OUT 3
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     00D
K:      WORD    ? ; Адрес первого элемента массива
N:      WORD    ? ; Количество элементов массива
R:      WORD    ? ; Результат
BEGIN:  CLA       ; Первая команда — адрес 010
        MOV     R
        ADD     (K)
        BPL     SKIP
        CLA
        ADD     R
        INC
        MOV     R
SKIP:   ISZ     N
        BR      BEGIN
        HLT

ORG     020
X:      WORD    6 DUP (?) ; Элементы массива
```

PIC - Position Independent Code (перемещаемый код)

- Код, который работает относительно того адреса на который загружен

Адрес	Содержимое	
	Код	Мнемоника
010	F200	**CLA**
011	4017	**ADD 17**
012	9015	**BPL 15**
013	F400	**CMA**
014	F800	**INC**
015	3018	**MOV 18**
016	F000	**HLT**
017	FFFE	X
018	0002	R=|X|

- Относительная (СК) адресация
  - ADD +5 ;(12+5=17)
  - BPL +2
- Адресация относительно базового адреса
  - ADD 17+0
  - MOV 17+1
- Сегментная адресация
  - Code-, Data-, StackSegment
  - 17 → DS; MOV DS:1
- Выполняется транслятором ассемблера из обычных меток и загрузчиком программы

ИТМО ВТ

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

ИТМО ВТ

Библиотеки

- Набор стандартных библиотечных функций
- Разделяемые (динамически линкуемые) и архивные (статически линкуемые)
  - `# find /lib /usr/lib -name "*.so" |wc -l`

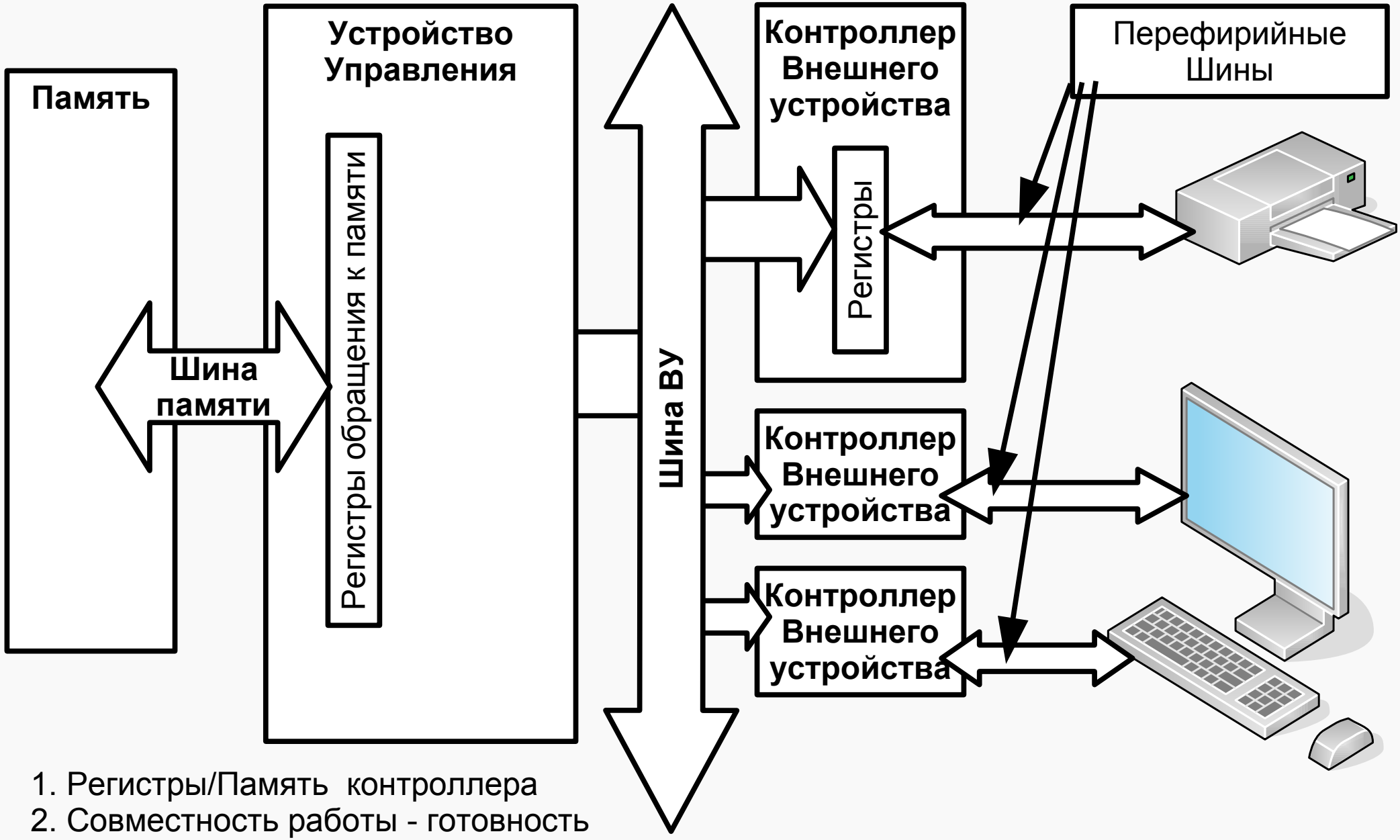
    `3510`
  - Статические связывают вызовы функций с телом функции в процессе компиляции
  - Динамические — в момент загрузки
- Если вам нужна функция — см. в библиотеки

ИТМО ВТ

Ввод-вывод

3

Подключение устройств



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

ИТМО ВТ

Драйверы

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

Ввод-вывод

Программно-управляемый

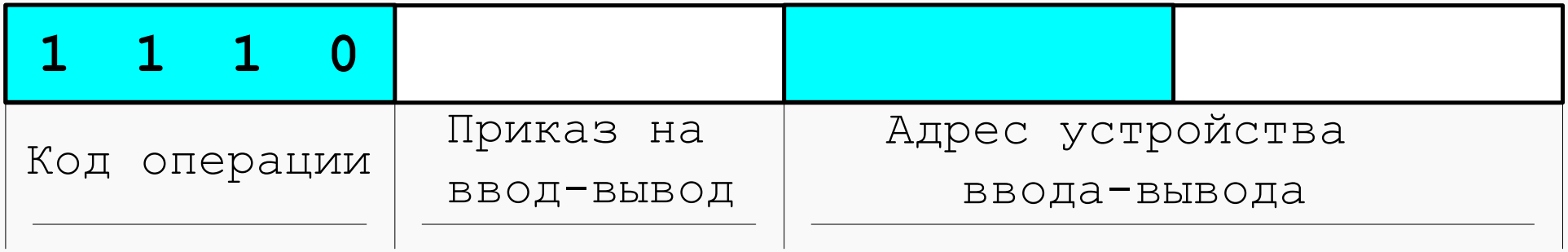
Управляемый-аппаратурой ПДП (DMA)

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

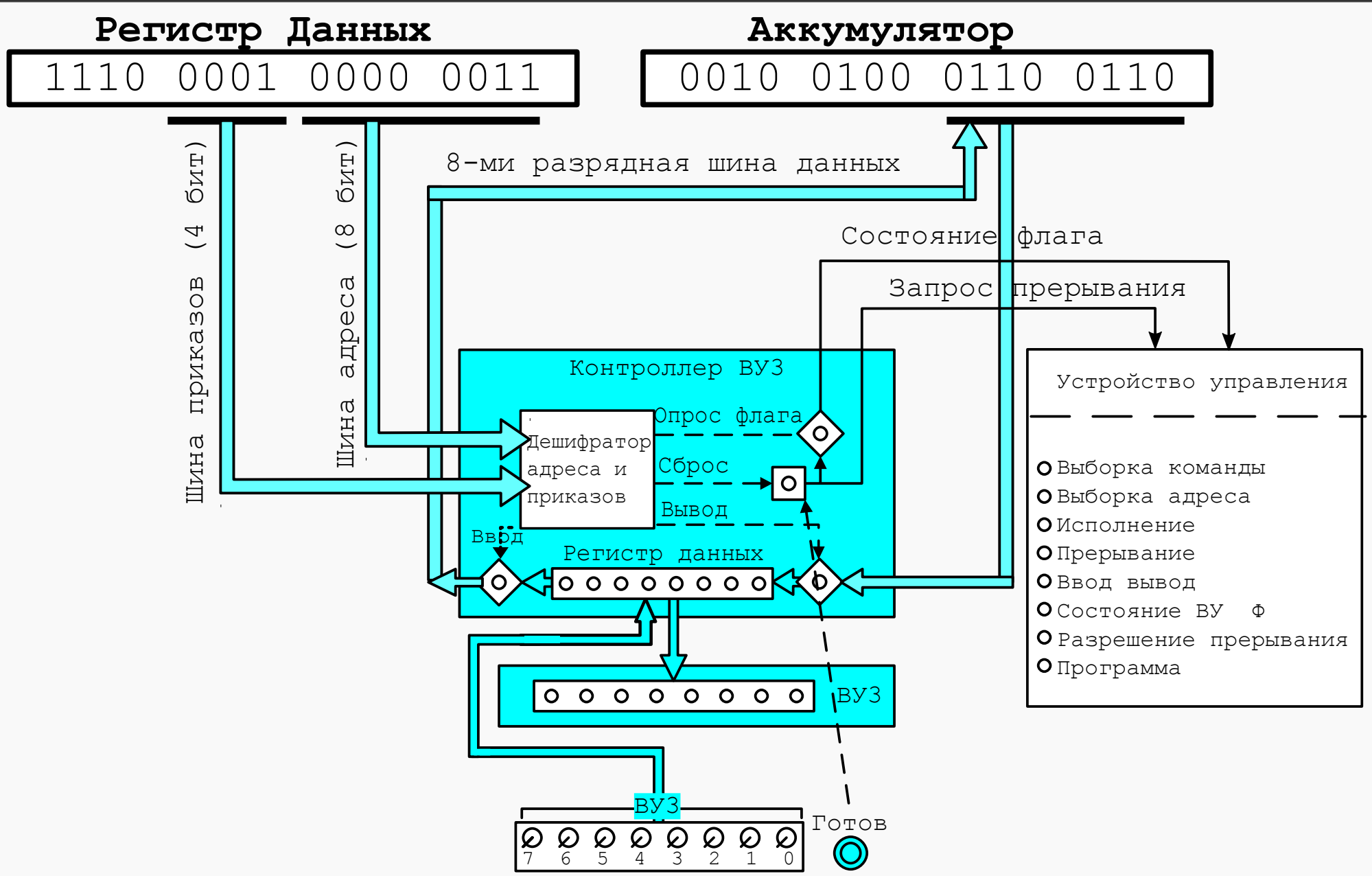
БЭВМ: Команды ввода-вывода

Наименование	Мнемон.	Код	Описание
Очистка флага	`CLF B`	E0XX	0 ➔ флаг устр.
Опрос флага	`TSF B`	E1XX	Если (флаг устр. B) = 1, то (CK) + 1 ➔ CK
Ввод	`IN B`	E2XX	(B) ➔ A
Вывод	`OUT B`	E3XX	(A) ➔ B

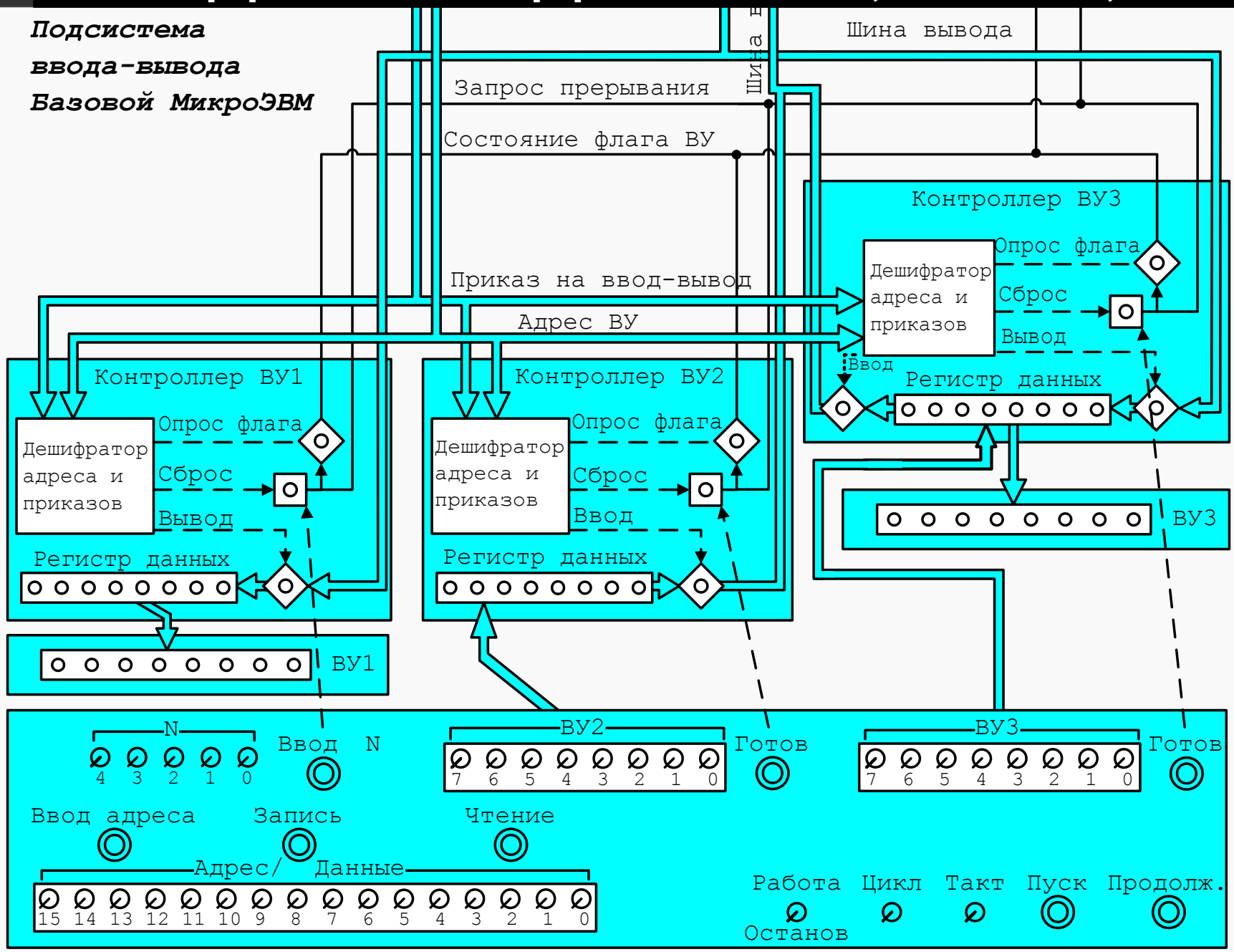
Команды ввода-вывода



Контроллер ВУ



БЭВМ: Основные устройства ввода-вывода: ВУ-1, ВУ-2, ВУ-3



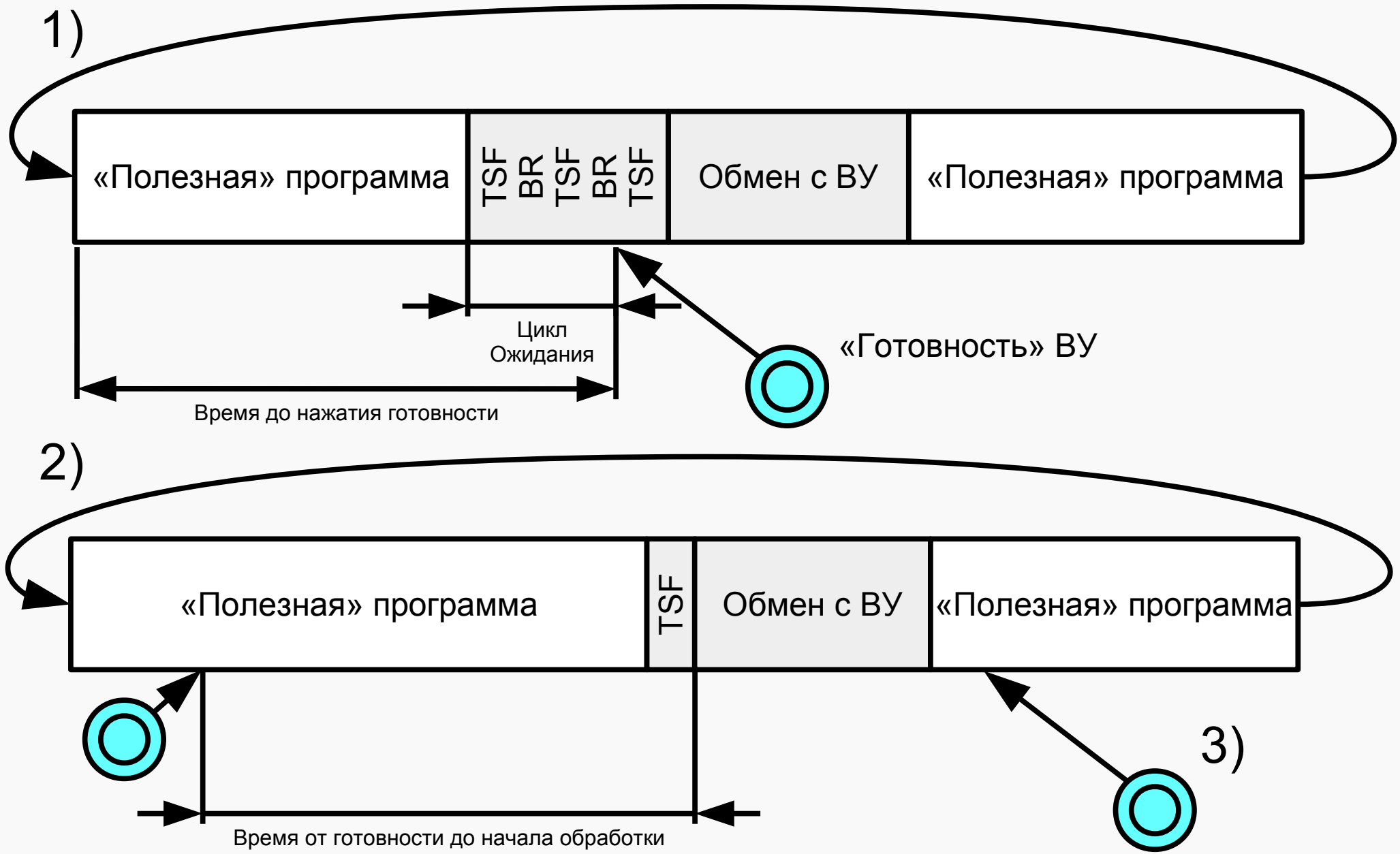
Цикл ожидания ввода

Ввод двух символов с устройства ввода

```
ORG     005
L8:     WORD    FFF8    ; -8 количество сдвигов
RES:    WORD    ?       ; Ячейка для записи слова "ДА"
ORG     020             ; Первая команда — адрес 020
BEGIN:  CLA
SPIN1:  TSF     2       ; Ожидание ввода первого символа
        BR      SPIN1   ;
        IN      2       ; Ввод первого символа
        CLF     2       ; Очистка флага. Можно вводить дальше
RL:     ROL             ; Сдвиг первого символа
        ISZ     L8      ; старший байт аккумулятора
        BR      RL
SPIN2:  TSF     2       ; Ожидание ввода второго символа
        BR      SPIN2
        IN      2       ; Ввод второго в младшие 8 разрядов A
        CLF     2
        MOV     RES
        HLT
```

Сколько циклов команд БЭВМ будет ждать ввода второго символа?

Временные издержки



ИТМО ВТ

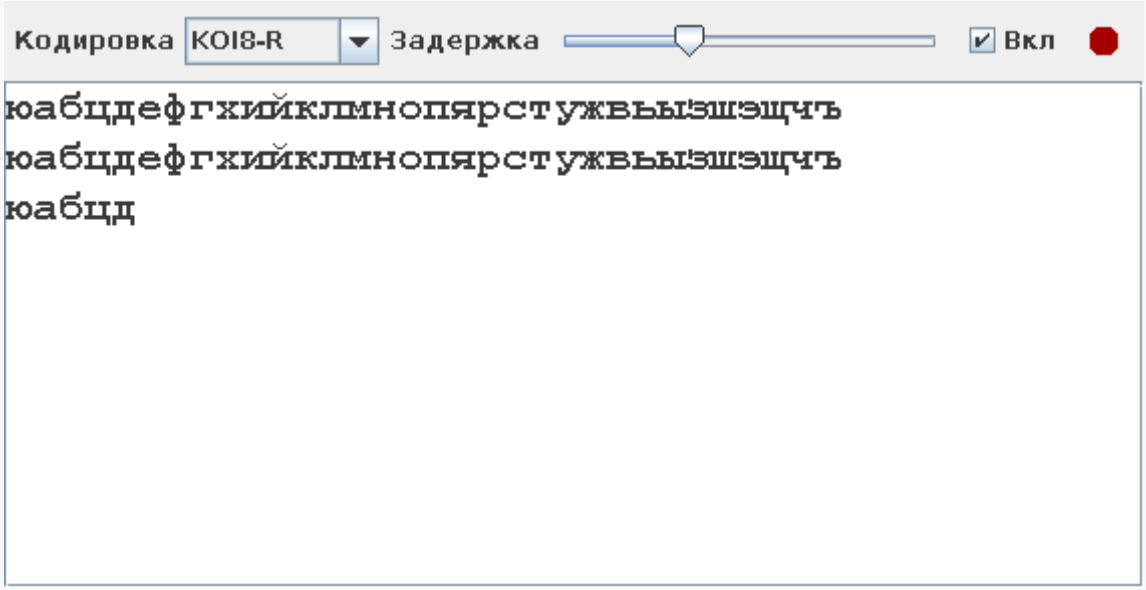
БЭВМ: ВУ-0 Таймер

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

БЭВМ: ВУ-4 Текстовый принтер

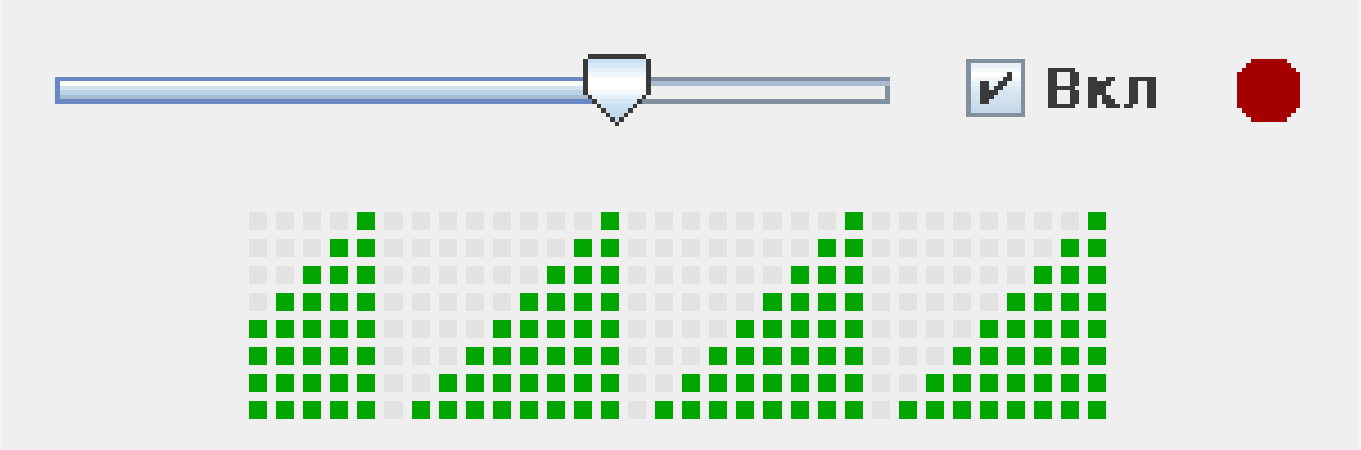
ИТМО ВТ

- 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



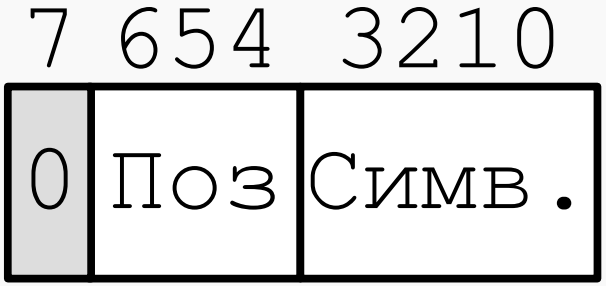
БЭВМ: ВУ-5 Бегущая строка

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

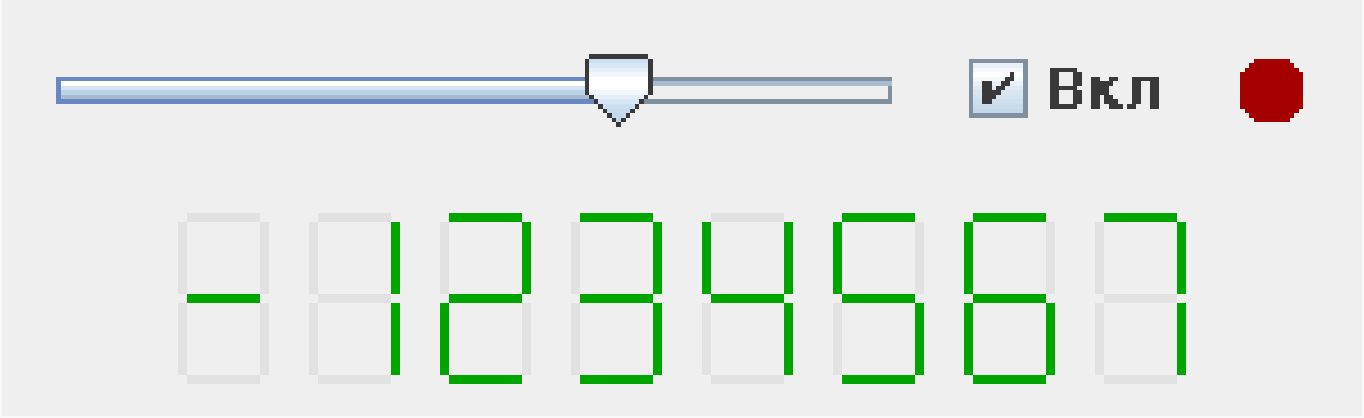


БЭВМ: ВУ-6 8-ми разрядный семисегментный индикатор

- Формат РДВУ:

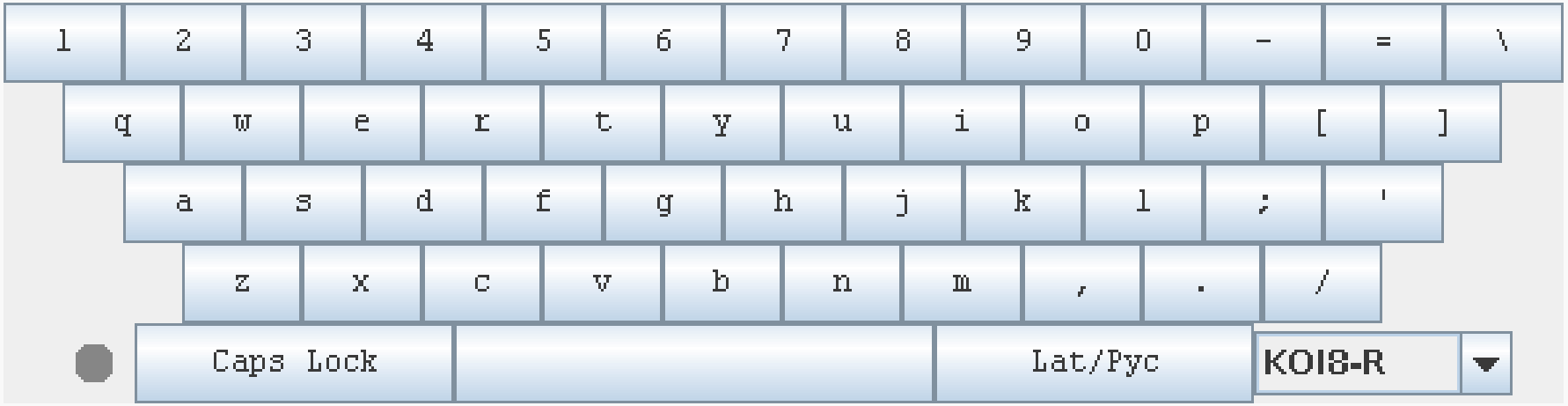


- Симв==(A_{16}) — установка в разряде знака «-»
- Симв==(B_{16}-F_{16}) — сброс разряда



БЭВМ: ВУ-7 клавиатура

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



БЭВМ: ВУ-8 Цифровая клавиатура

- При нажатии клавиши ее код помещается в РДВУ
- Клавиша 0-9 код 0-9
- Клавиша «-» код A
- «+» код B
- «/» код C
- «*» код D
- «.» код E
- «=» код F

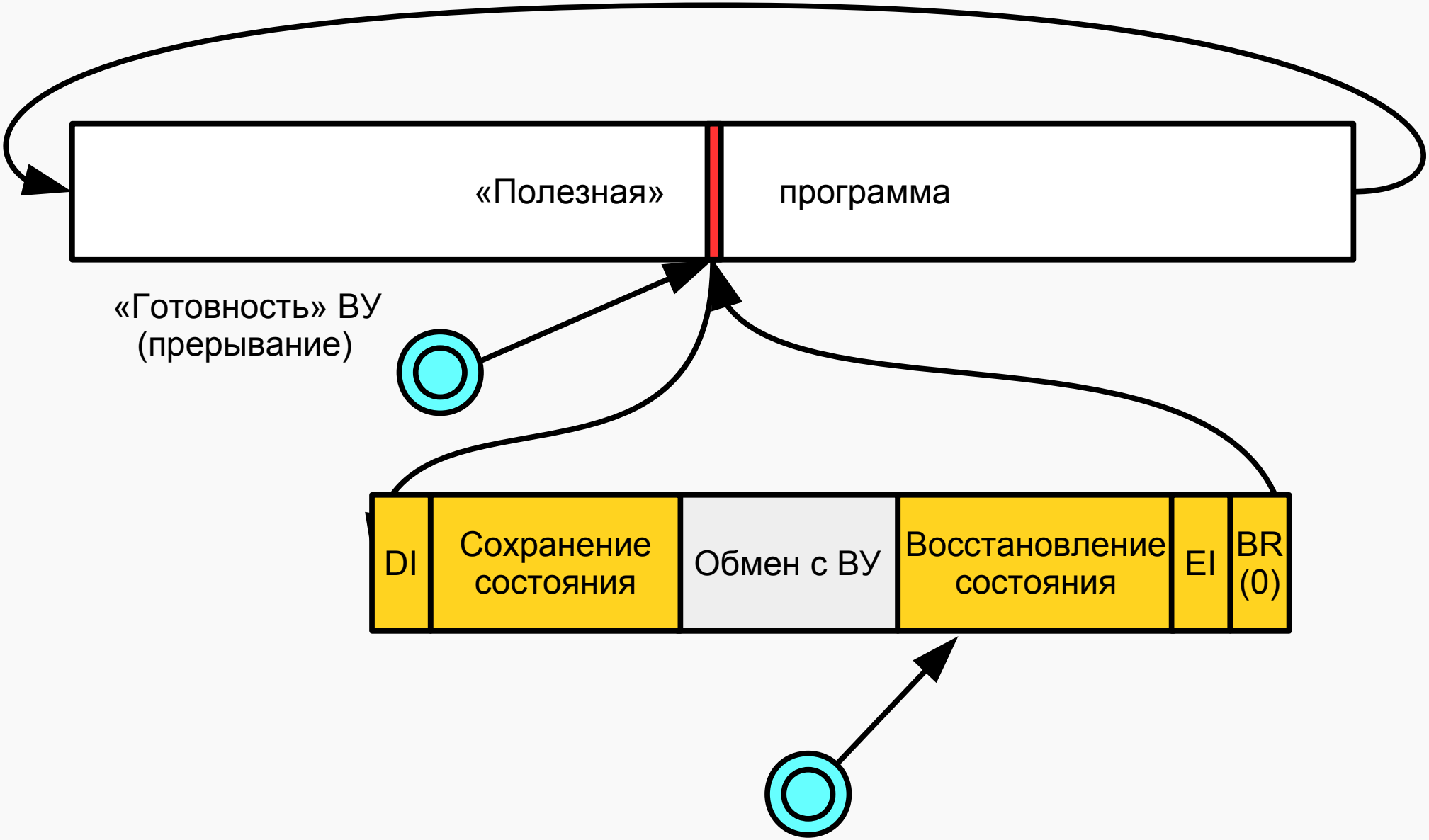
7	8	9	/
4	5	6	*
1	2	3	-
0	.	=	+

ИТМО ВТ

БЭВМ: ВУ-9 БЭВМ

- Два стандартных контроллера ввода-вывода (как ВУ-3) связаны между собой
- IN — читает информацию из РДВУ контроллера
- OUT — записывает данные в РДВУ **обоих** контроллеров
- CLF одного контроллера приводит к установке флага связанного контроллера

Инициация обмена по прерыванию



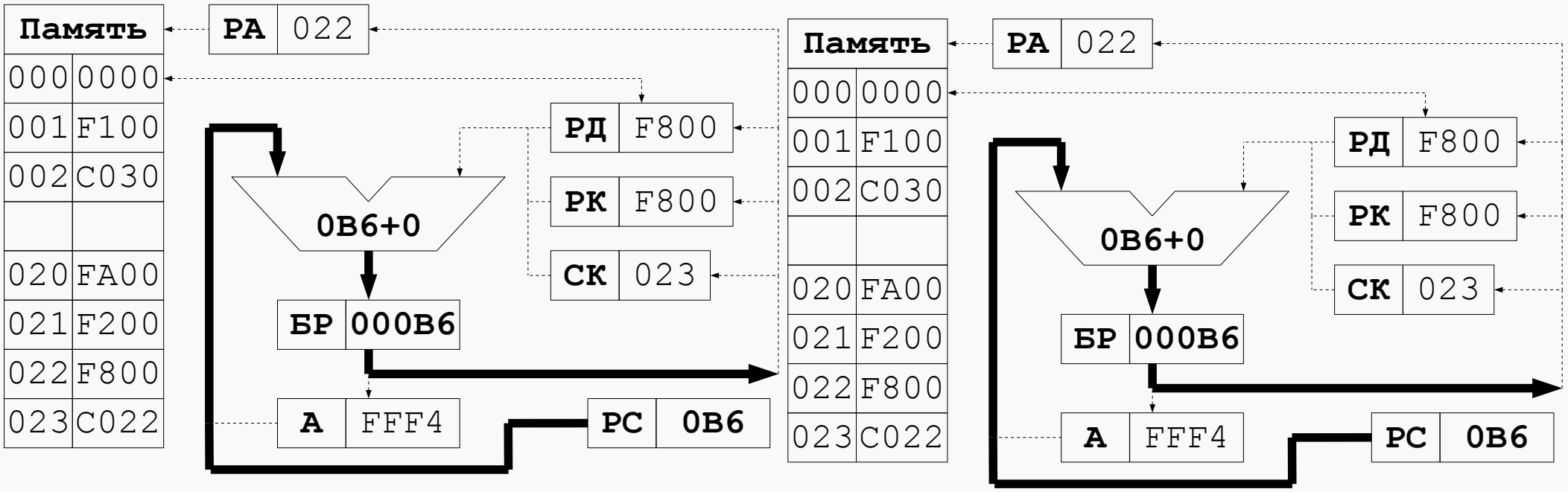
ИТМО ВТ

Регистр состояния и команды

Бит	Мнем.	Содержимое
0	C	Перенос
1	Z	Нуль
2	N	Знак
3	0	0 - используется для организации безусловных переходов в МПУ
4	EI	Разрешение прерываний
5	I	Прерывание (логическое “И” шины запроса на прерывание и бита 4 РС – “разрешение прерываний”)
6	F	Состояние ВУ (Ф)
7	W	Состояние тумблеров РАБОТА/ОСТАНОВ (1 - РАБОТА)
8	P	Программа

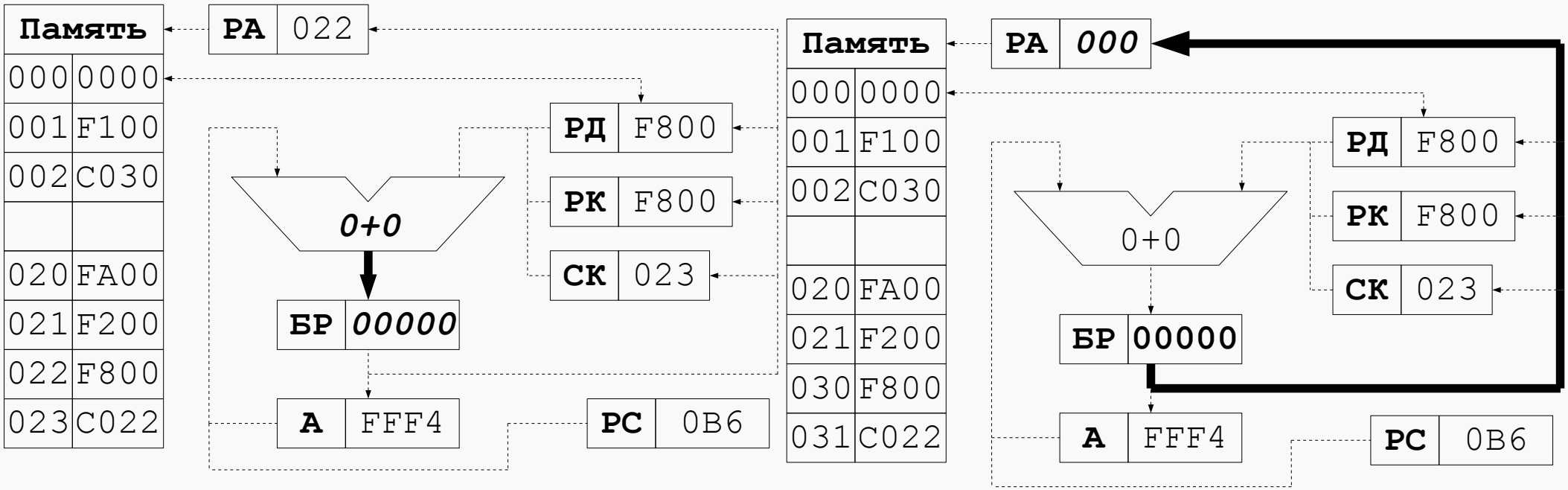
Наименование	Мнемон.	Код	Описание
Разрешение прерываний	EI	FA00	Установка бита 4 (разрешение прерываний)
Запрещение прерываний	DI	FB00	Сброс в 0 бита 4 (разрешение прерываний)
Возврат из прерывания	BR (0)	C800	Переход по адресу, сохраненному в яч. 0

Цикл прерывания



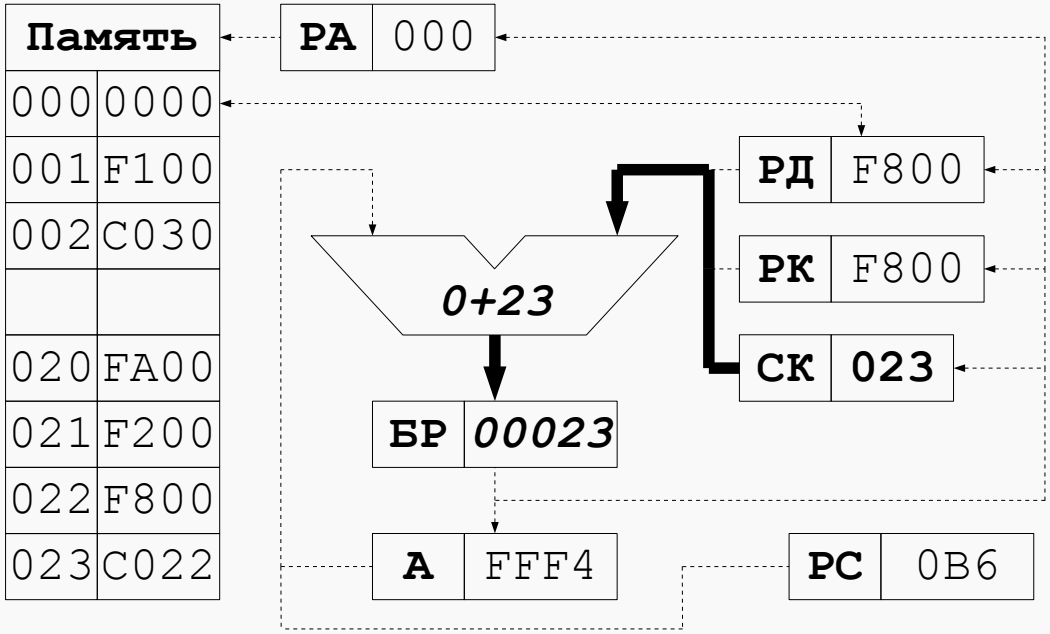
- После выполнения команды управление передается циклу прерывания
- Если бит 7 РС (Работа/Останов) == 0 то останов БЭВМ
- Если бит 5 РС (Прерывание) ==0 то переход к следующий команде (циклу выборки команды)

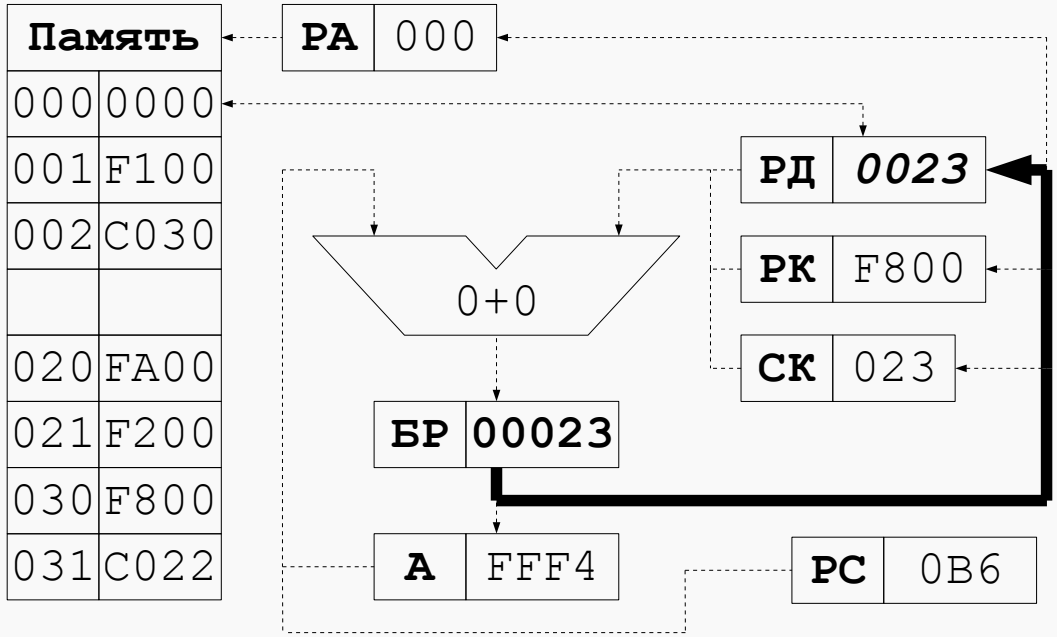
Цикл прерывания



- 0 — (ячейка сохранения адреса возврата) записывается в РА

Цикл прерывания

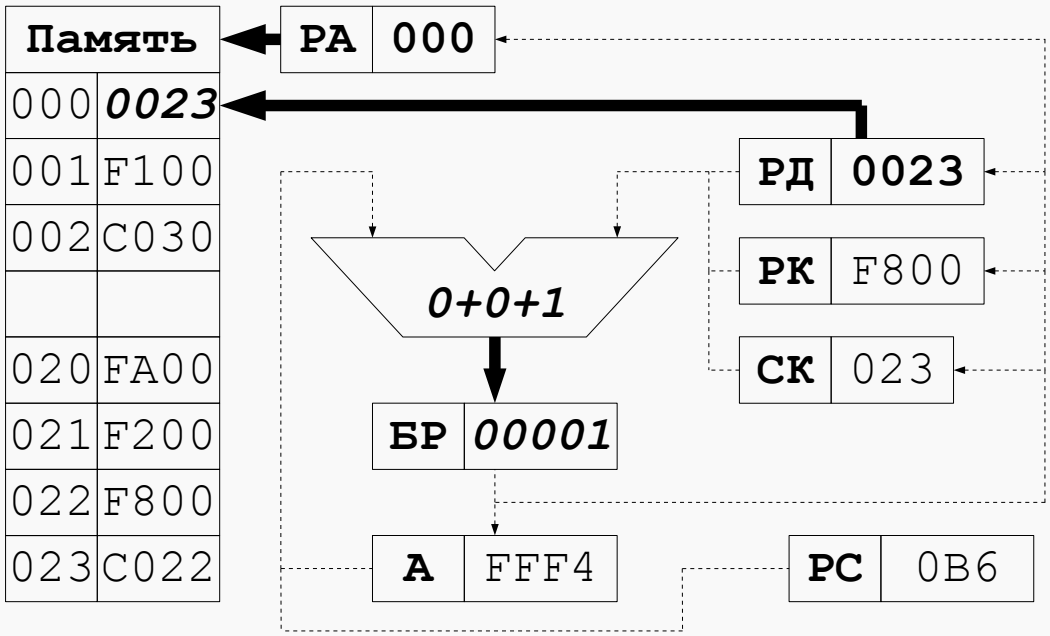


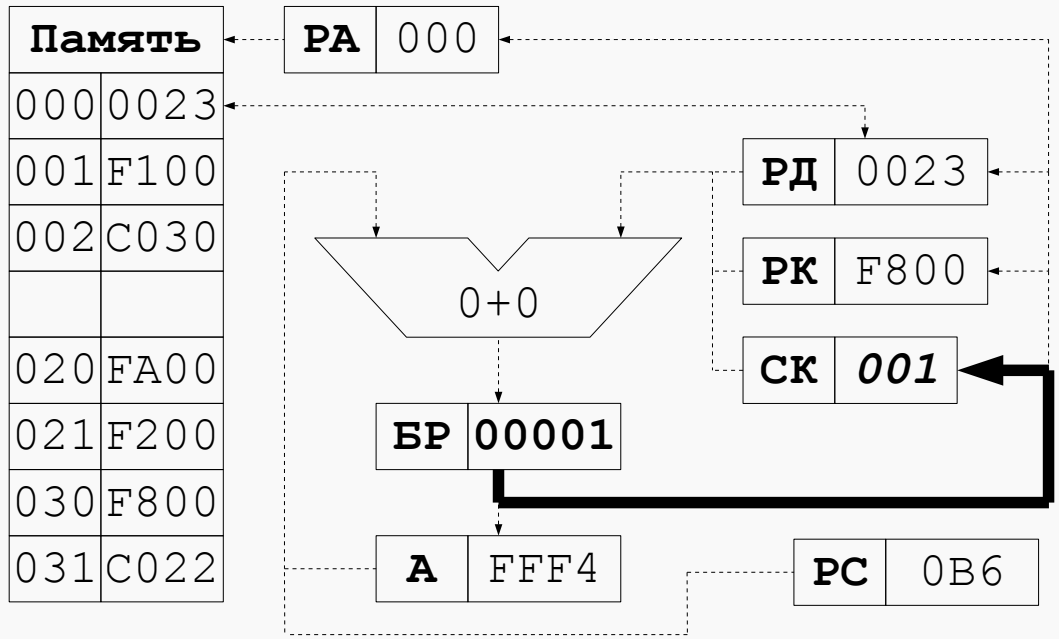


- Содержимое СК через БР переписывается в РД

ИТМО ВТ

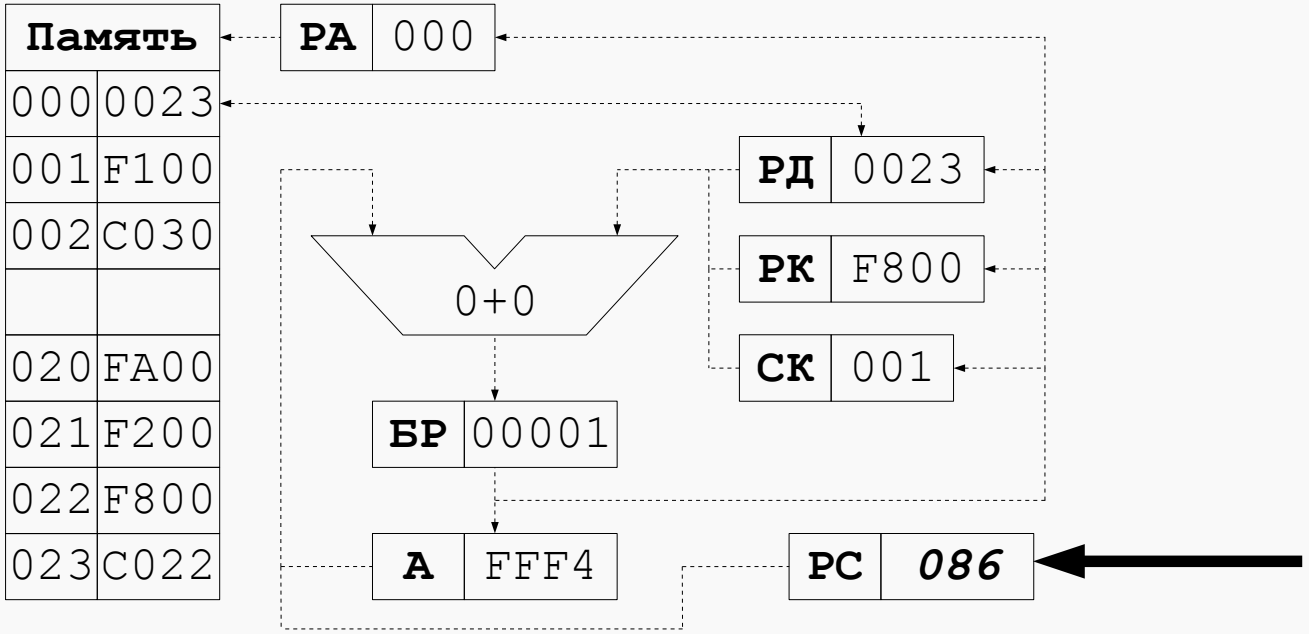
Цикл прерывания





- Содержимое РД записывается по адресу 0, тем самым обеспечивая возможность возврата из прерывания
- 1 записывается в БР и далее в СК, определяя первую исполняемую команду

Цикл прерывания



- Производится запрещение прерываний в РС, сбрасывается бит 4 «разрешение прерываний», что приводит к сбросу бита 5 «прерывание» (B->8)
- После этого переход к циклу выборки команды из ячейки 1, т.е. обработке прерывания

ИТМО ВТ

После каких команд нет цикла прерывания?

- DI
  - Нет необходимости. Прерывания запрещены
- HLT
  - Нет необходимости. Программа остановлена
- EI
  - Приведет к зацикливанию обработчика прерывания
    Завершение обработчика:

```
EI
BR (0)
```

Обработка прерываний: Основная программа

```
                    Обработка прерываний
  Готовность ВУ1: 2*A→РДВУ1, Готовность ВУ3: РДВУ3→ яч.29
        ORG 000
RET:    WORD ?   ; Ячейка адреса возврата
        NOP      ; Ячейка отладочной точки
                 ; останова (NOP/HLT)
        BR INT   ; Переход к обработке прерываний

        ORG 020  ; Основная программа
BEGIN:  EI       ; Установка состояния разрешения прерывания
        CLA      ; Первоначальная очистка аккумулятора
LOOP:   INC      ; Цикл для наращивания
        BR LOOP  ; содержимого аккумулятора


        ORG 029  ; Ячейка для хранения кодов,
IO3:    WORD  ?  ; поступающих с ВУ-3
```

Обработка прерываний: Сохранение и восстановление

Готовность ВУ1: 2*A→РДВУ1, Готовность ВУ3: РДВУ3→ яч.29

```
        ORG   030
INT:    MOV   SAVED_A  ; Сохранение содержимого A и C
        ROL
        MOV   SAVED_C
```

Основная программа обработки прерывания
(см. следующий слайд)

```
RESTORE:NOP           ; отладочная точка останова (NOP/HLT)
        CLA           ; Восстановление содержимого C и A
        ADD SAVED_C
        ROR
        CLA
        CMA
        AND SAVED_A
        EI            ; Возобновление состояния
                      ; разрешения прерывания
        BR (RET)      ; Возврат из обработки прерывания
SAVED_A:WORD ?
SAVED_C:WORD ?
```

Обработка прерываний: Прерывание

Готовность ВУ1: 2*A→РДВУ1, Готовность ВУ3: РДВУ3→ яч.29

```
          TSF 3        ; Опрос флага ВУ-3
          BR CHECK1    ; Если сброшен → к опросу флага ВУ-1
          BR READY3    ; Переход на ввод данных из ВУ-3
CHECK1:   TSF 1        ; Опрос флага ВУ-1
          BR READY2    ; Если сброшен → к опросу флага ВУ-2
          BR READY1    ; Переход на вывод данных в ВУ-1
READY3:   CLA          ; Ввод данных из ВУ-3
          IN 3
          CLF 3        ; Сброс флага ВУ-3
          MOV IO3      ; Пересылка значения в ячейку 29
          BR RESTORE
READY1:   CLA
          ADD SAVED_A
          ROL
          OUT 1        ; Вывод на ВУ-1 8-ми
          CLF 1        ; Сброс флага ВУ-1
          BR RESTORE
READY2:   CLF 2
```

ИТМО ВТ

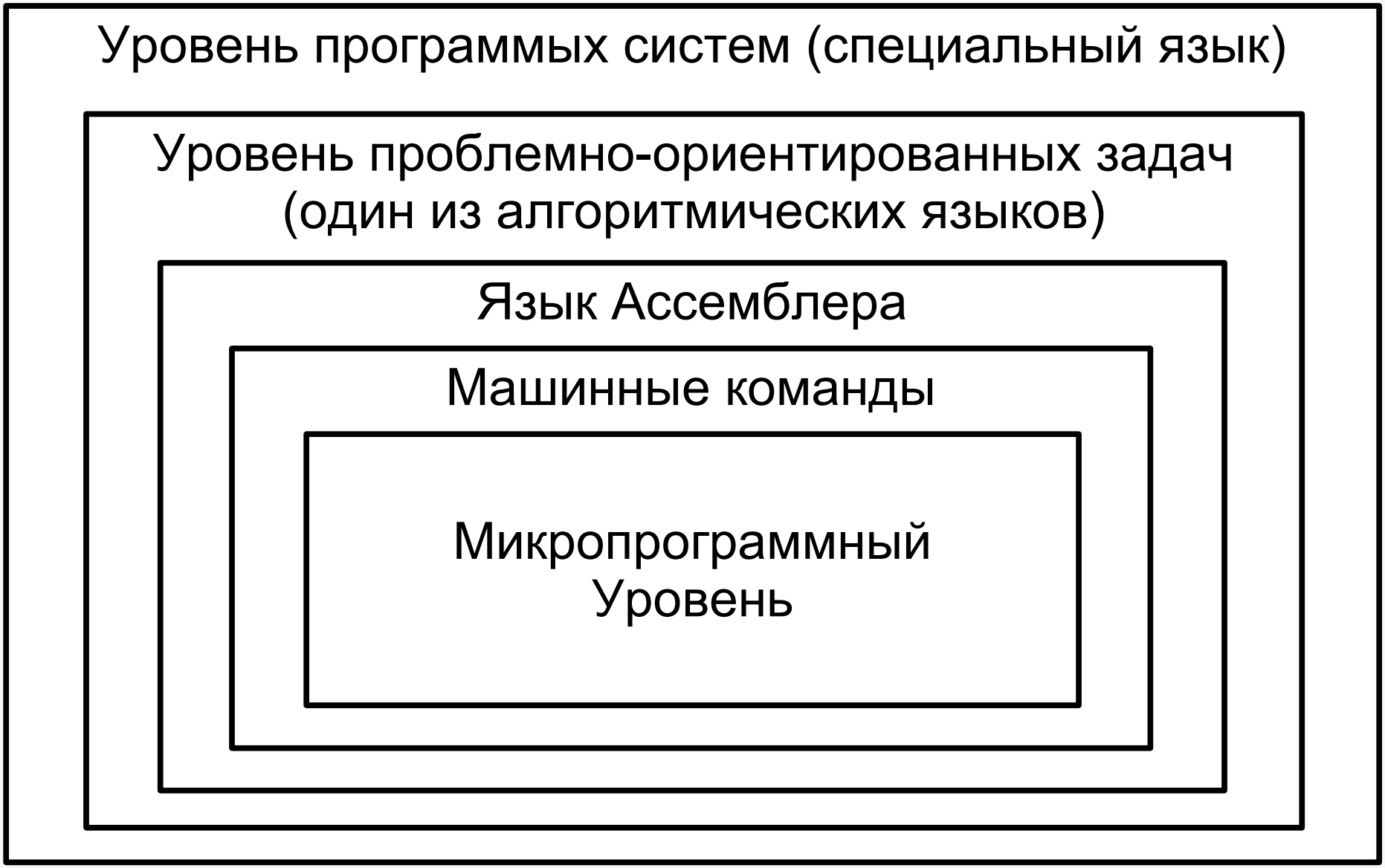
Ввод-вывод

Микропрограммное устройство управления

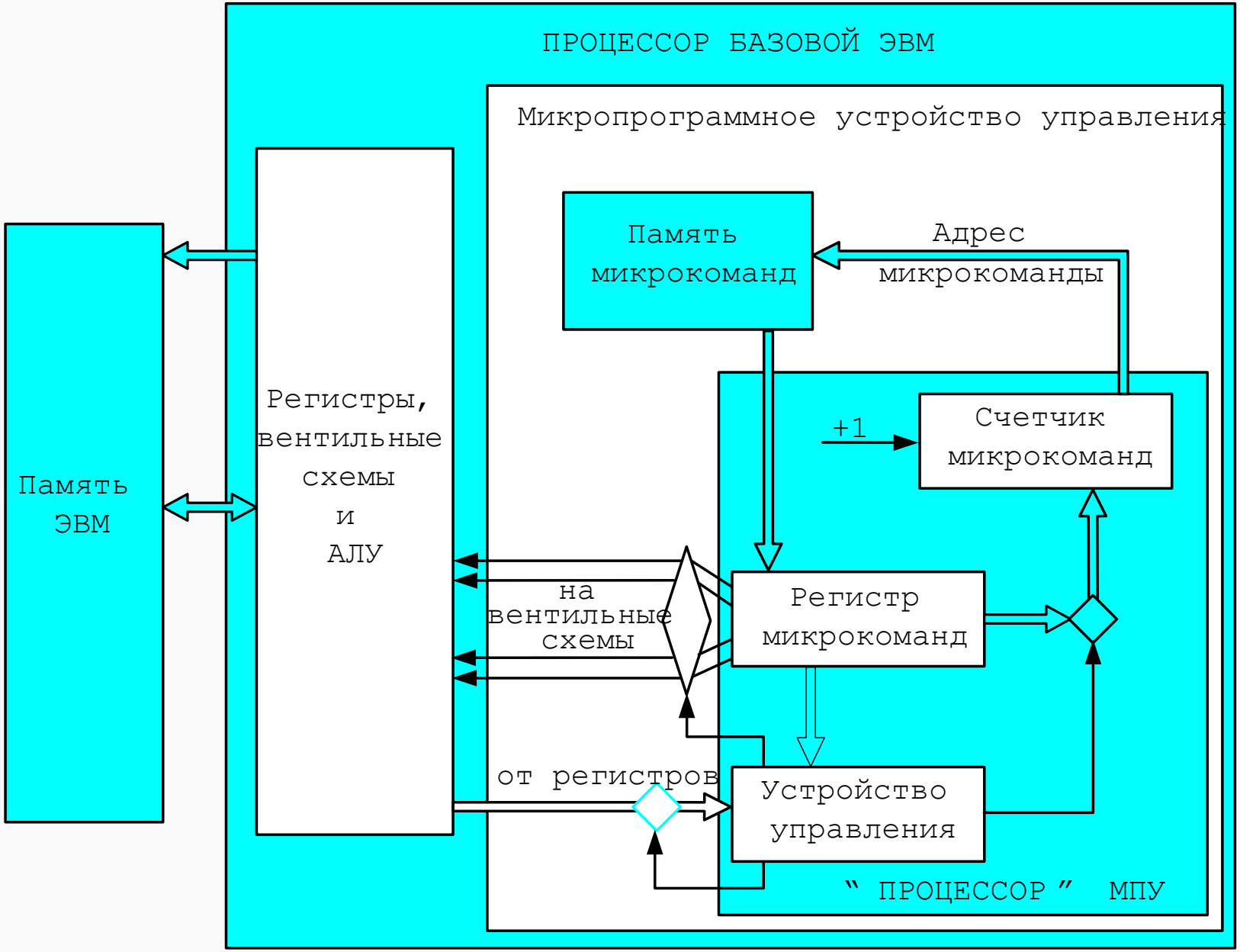
4

ИТМО ВТ

Многоуровневая ЭВМ

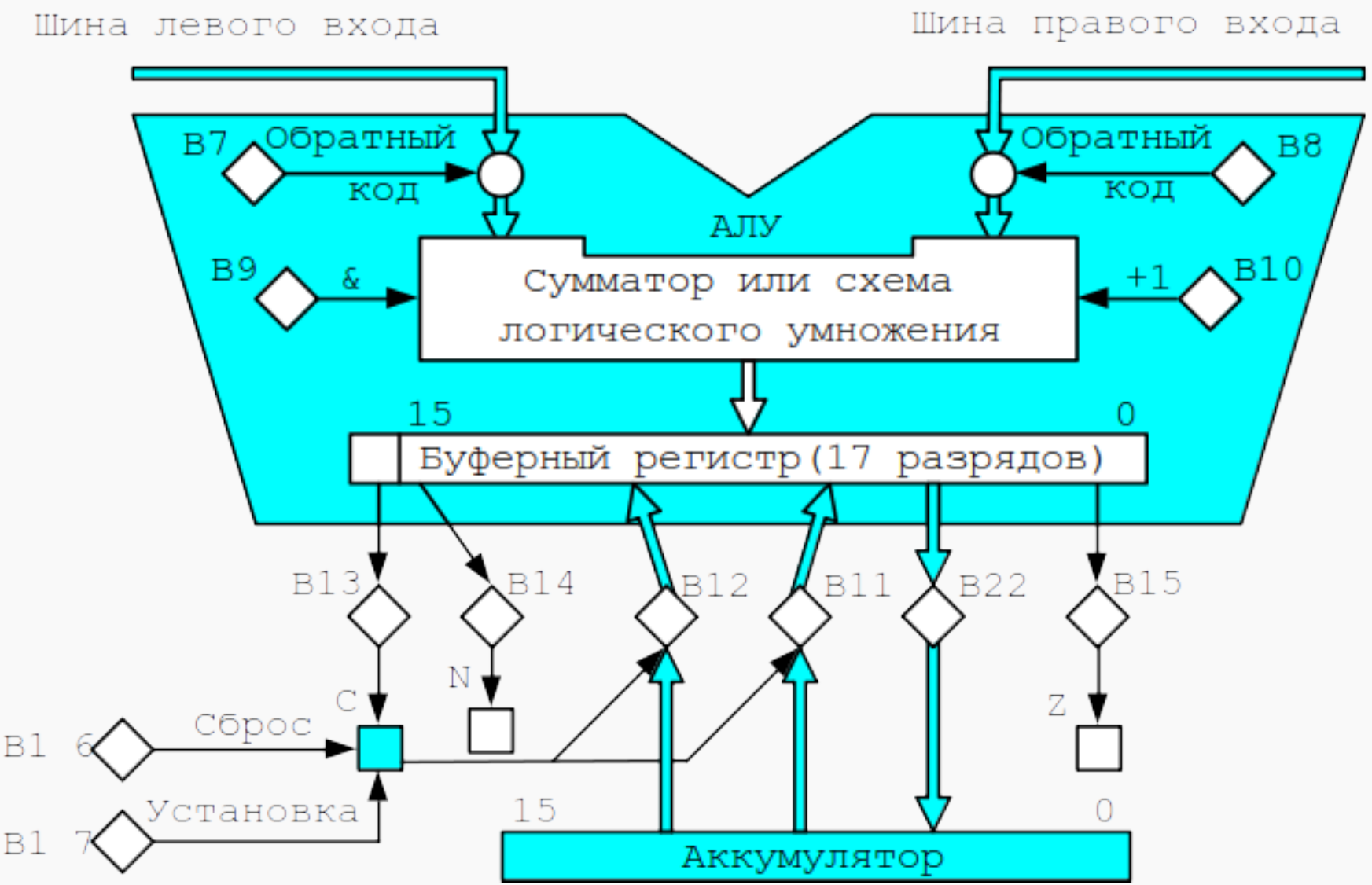


МПУ


ПРОЦЕССОР БАЗОВОЙ ЭВМ
Микропрограммное устройство управления
Память микрокоманд
Адрес микрокоманды
Регистры, вентильные схемы и АЛУ
Память ЭВМ
+1
Счетчик микрокоманд
на вентильные схемы
Регистр микрокоманд
от регистров
Устройство управления
" ПРОЦЕССОР " МПУ

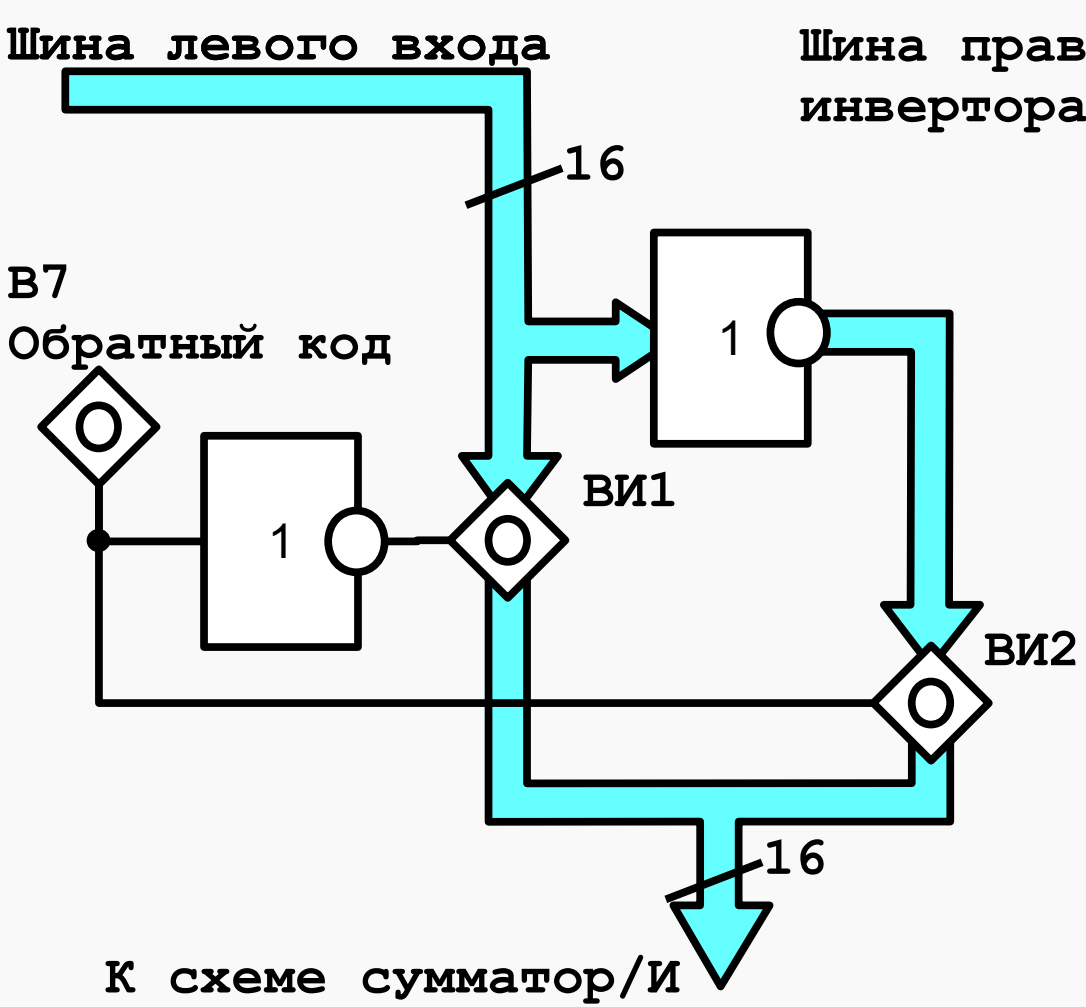
ИТМО ВТ

АЛУ

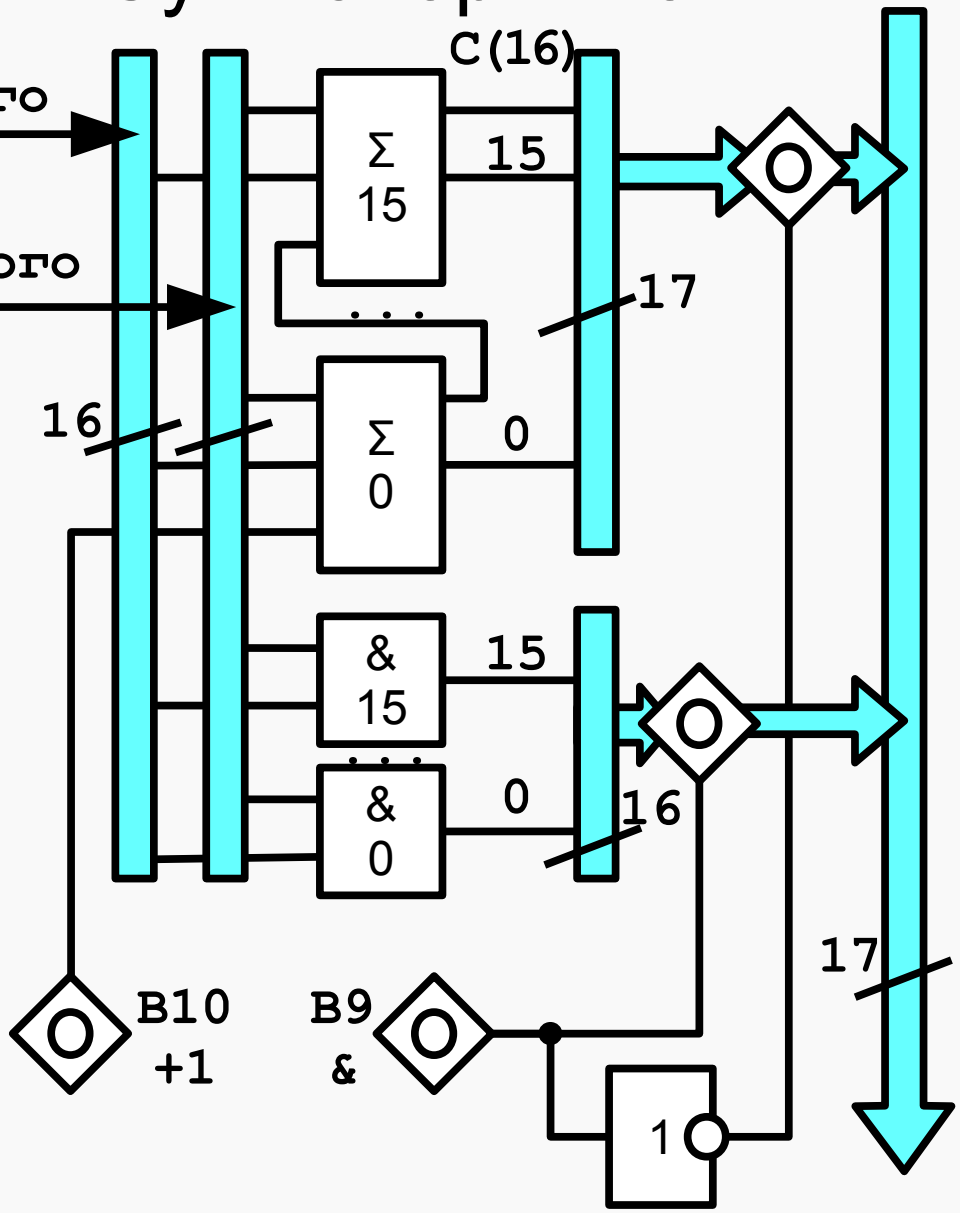

Шина левого входа
Шина правого входа
B7
Обратный код
Обратный код
B8
АЛУ
B9
&
Сумматор или схема логического умножения
+1
B10
15
0
Буферный регистр(17 разрядов)
B13
B14
B12
B11
B22
B15
C
N
Z
B1 6
Сброс
B1 7
Установка
15
0
Аккумулятор

АЛУ: Обратный код, сумматор и логическое «И»

Схема обратного кода



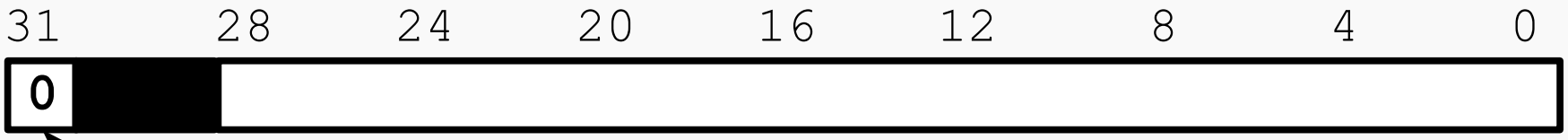
Сумматор и лог. «И»



Горизонтальные микрокоманды

ИТМО ВТ

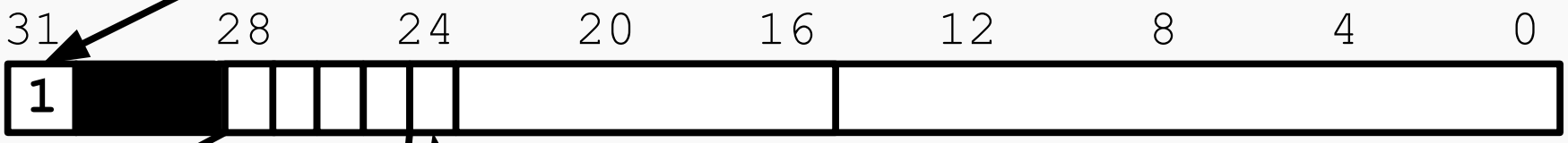
Операционная микрокоманда



Биты управления отдельными вентильными схемами

Код операции

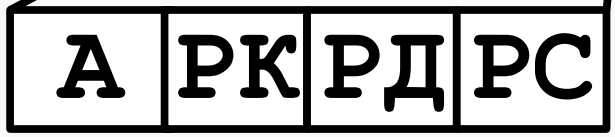
Управляющая микрокоманда



Адрес перехода

Поле выбора проверяемого бита

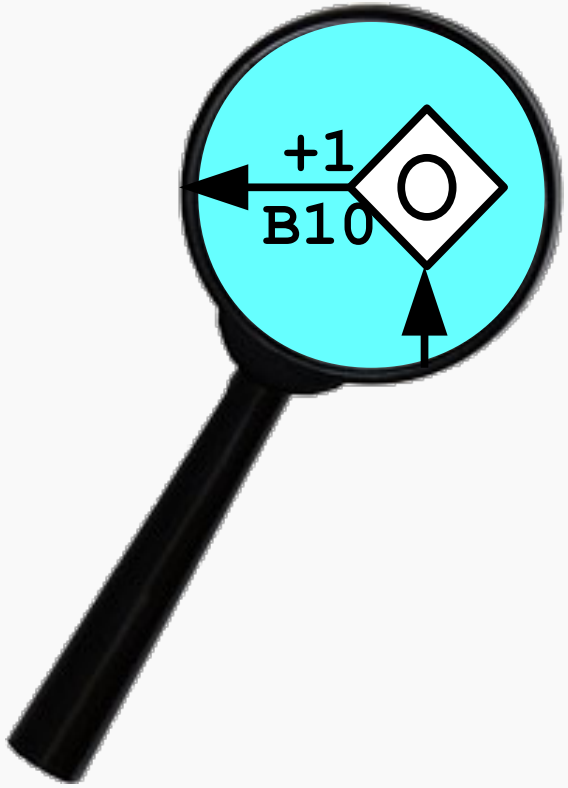
Однобитовое поле сравнения



Поле выбора проверяемого бита

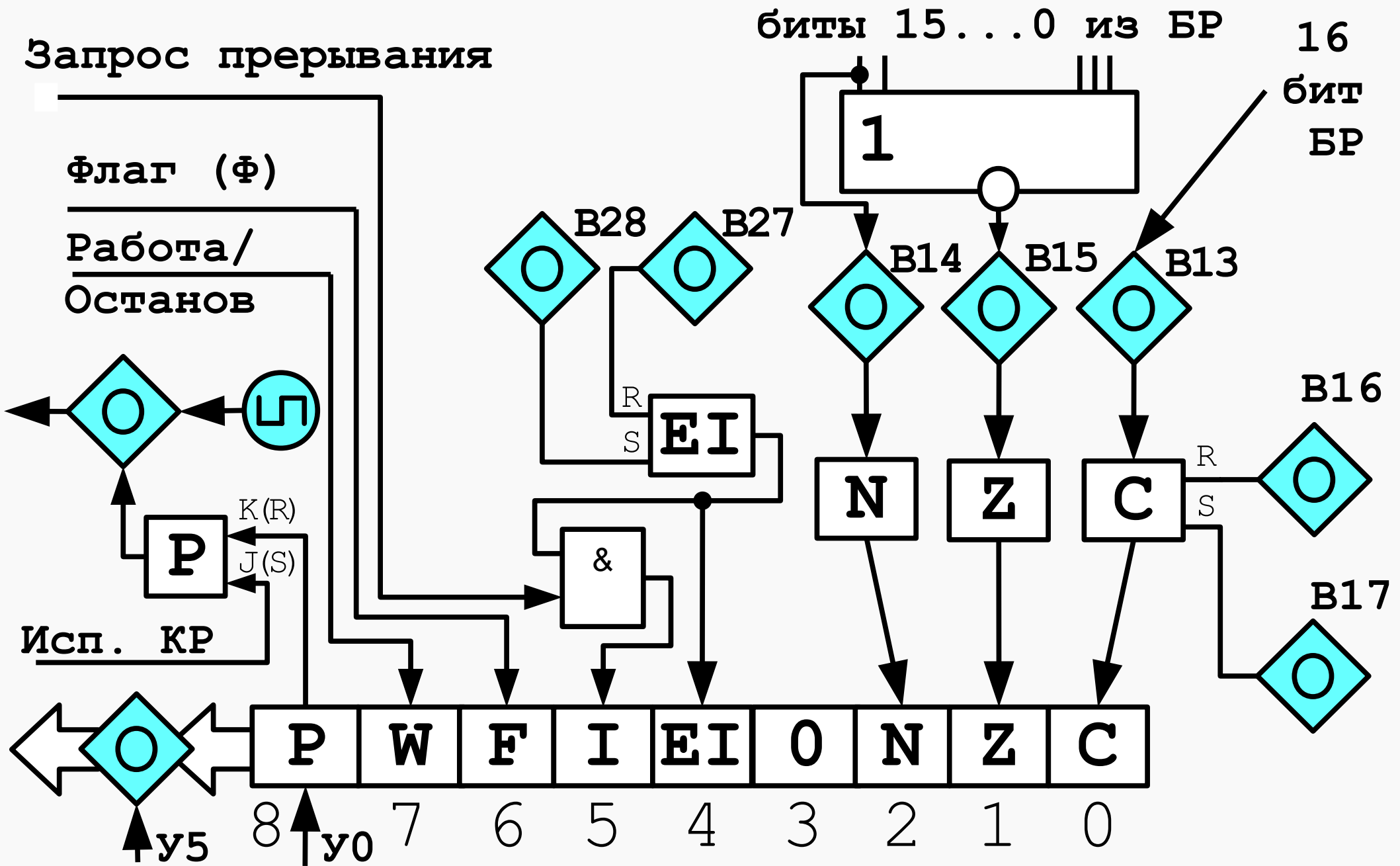
ИТМО ВТ

МПУ



Регистр адреса
B18
У18
B23
У23
B24
У24
Регистр данных
У19
B19
B1
У1
Регистр команд
У20
B20
B2
У2
У3
Счетчик команд
B21
B3
У21
Устройство управления
Регистр микрокоманд
ВР0
ВР1
ВВ0-ВВ15
У0
У1
У2
У3
У4
У5
У6
У7
У8
У9
У10
У11
У12
У13
У14
У15
У16
У17
У18
У19
У20
У21
У22
У23
У24
У25
У26
У27
У28
Схема сравнения
=
ПАМЯТЬ МИКРО-КОМАНД
B7
Обратный код
У7
АЛУ
Обратный код
B8
У8
&
B9
У9
Сумматор или схема логического умножения
+1
B10
У10
Буферный регистр АЛУ
Бит сравнения
Адрес перехода
НЕ
B13
У13
B14
У14
N
B12
У12
B11
У11
B15
У15
Z
У16
Сброс
B16
C
Установка
B17
У17
B4
У4
Аккумулятор
B22
У22
ИЛИ-НЕ
НЕ
СчМК
+1
Регистр состояния
B5
Схемы управления вводом-выводом
ИНДИКАТОРЫ СОСТОЯНИЯ
Выборка команды
Выборка адреса

Регистр состояния



ИТМО ВТ

Вентильные схемы

- Входные сигналы на входы АЛУ
  - B1-B3 — левый вход АЛУ
  - B4-B6 — правый вход АЛУ
- Арифметические операции и сдвиги
  - B7-B12
- Установка признаков
  - B13-B22
- Работа с памятью
  - B23-B24
- Организация ввода-вывода информации
  - B25-B28
- Останов ЭВМ — B0

ИТМО ВТ

Интерпретатор БЭВМ

- 256 ячеек для хранения микрокоманд, включая резерв
- Содержит вертикальные микрокоманды (см. далее)
- Оформлено в виде таблицы (см. Методу!)

Ад-рес	Микрокоманды		Комментарии	
	Горизонтальная	Вертикальная	Метка	Действие
01	0000 0008	0300	НАЧ	СК→БР
02	0004 0000	4001		БР→РА
03	0080 0408	0311		ОП(РА)→РД, СК+1→БР

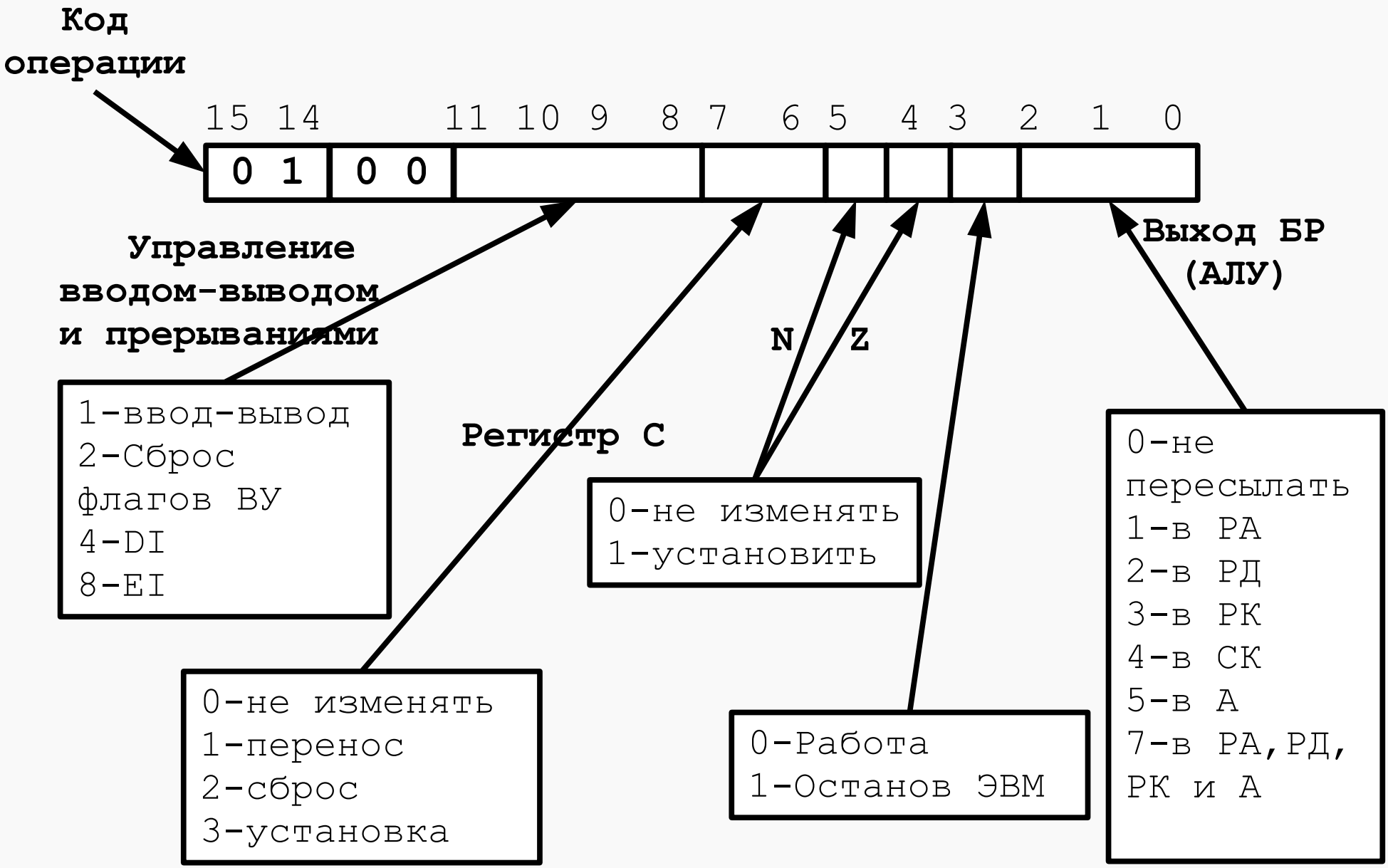
ИТМО ВТ

Интерпретатор БЭВМ

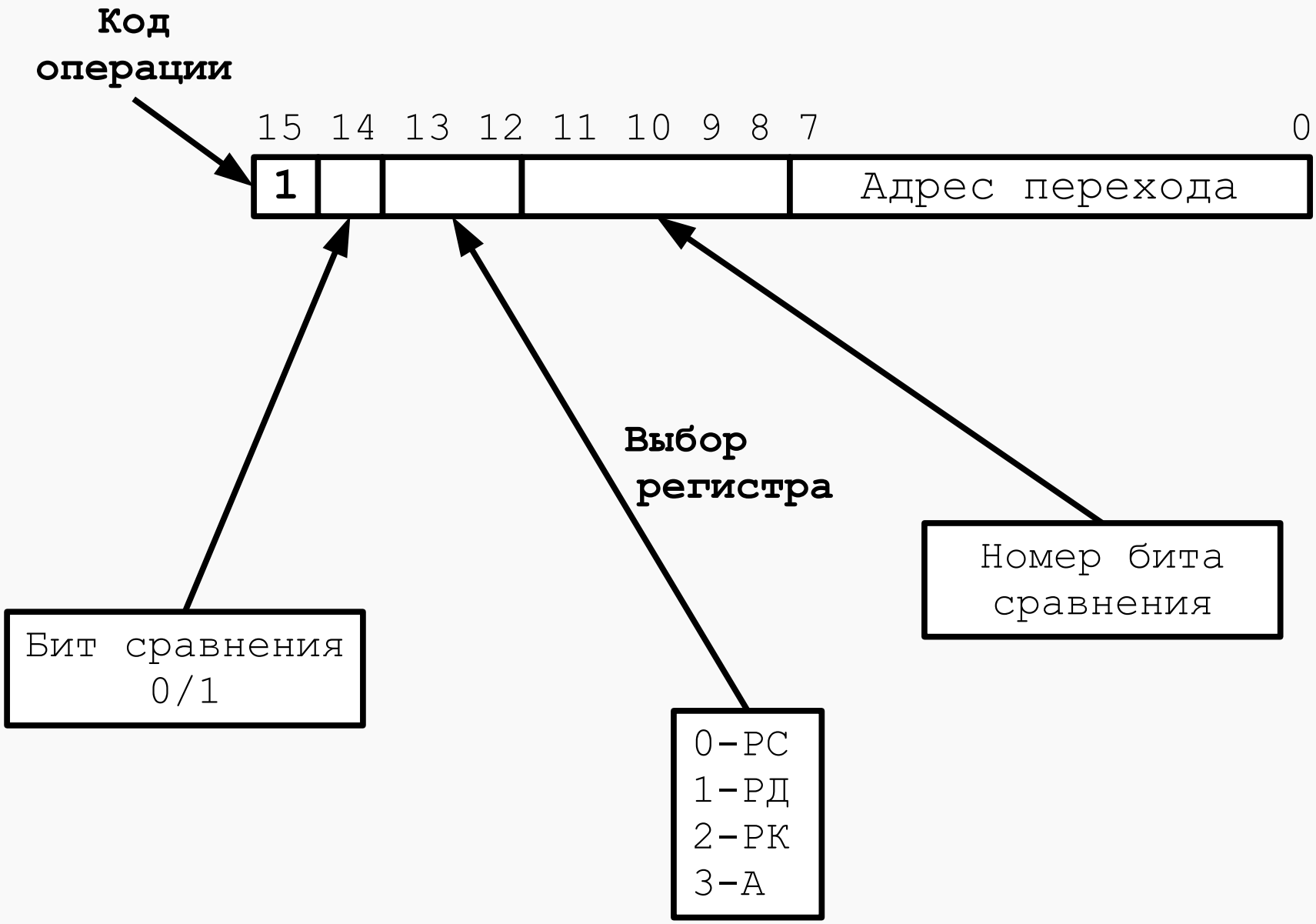
- Цикл выборки команд
- Цикл выборки адреса операнда
- Цикл исполнения адресных команд
  - Декодирование адресных команд
  - Исполнение адресных команд
- Декодирование и исполнение безадресных команд
- Декодирование и исполнение команд ввода-вывода
- Цикл прерывания
- Пультовые операции
- Свободные ячейки для:
  - Арифметической команды
  - Команды перехода
  - Безадресной команды

Вертикальные микрокоманды: ОМК0

Вертикальные микрокоманды: ОМК1



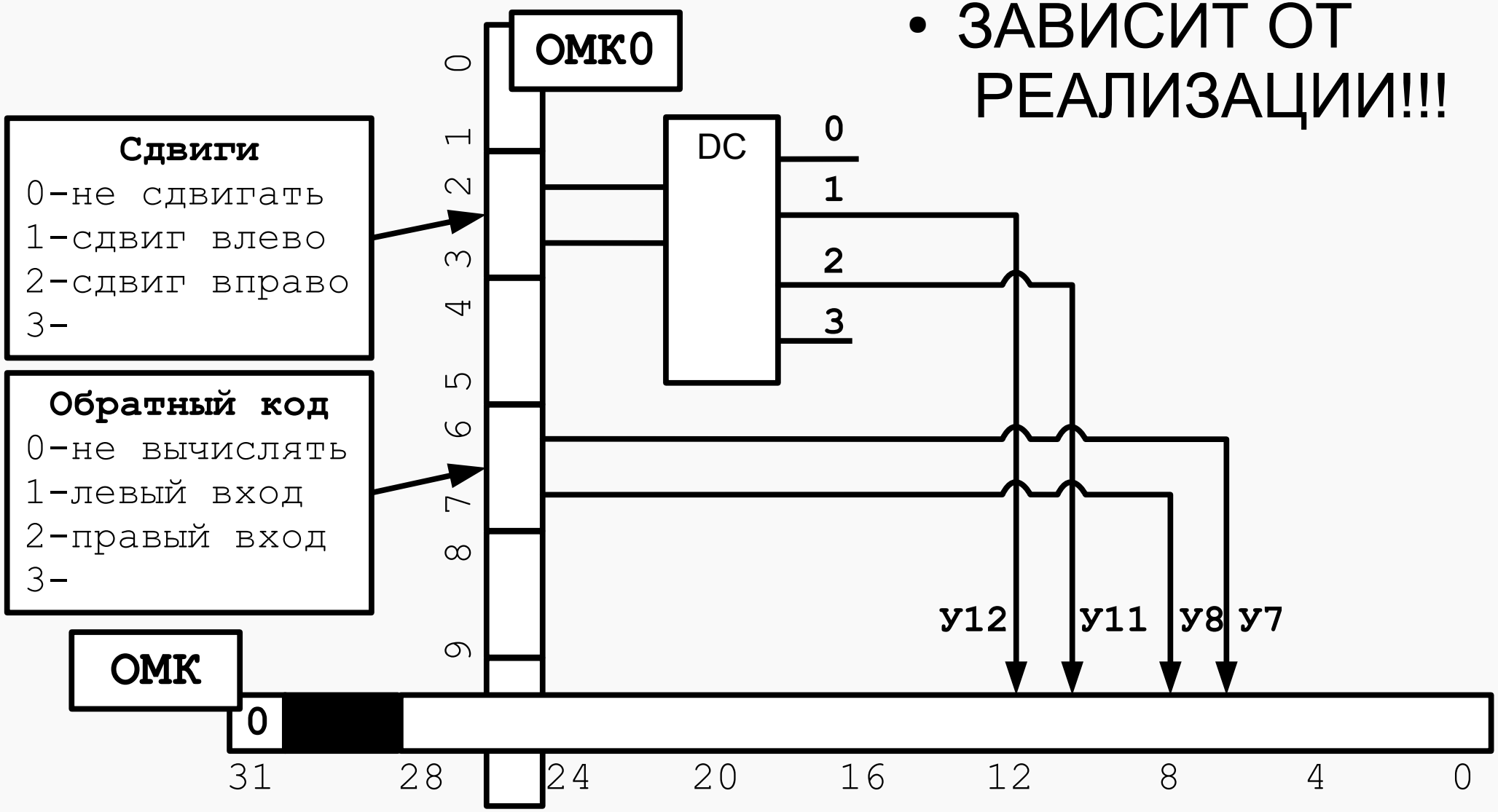
Вертикальные микрокоманды: УМК



Преобразование вертикальных микрокоманд в горизонтальные

- Задача — преобразовать коды в управляющие сигналы на вентилях

- ЗАВИСИТ ОТ РЕАЛИЗАЦИИ!!!



ИТМО ВТ

С БЭВМ — все!!!!