

**Арифметико-логическое
устройство.
Операции над целыми
десятичными числами,
представленными
в двоично-десятичном коде.**

Фрагмент кодовой таблицы ASCII (действителен для всех кодовых таблиц)

Символ	Код (Dec)	Код (Hex)	Код (Bin)
'0'	48	30	0011 0000
'1'	49	31	0011 0001
'2'	50	32	0011 0010
'3'	51	33	0011 0011
'4'	52	34	0011 0100
'5'	53	35	0011 0101
'6'	54	36	0011 0110
'7'	55	37	0011 0111
'8'	56	38	0011 1000
'9'	57	39	0011 1001

Если с клавиатуры введена строка '117' , а переменная-приемник имеет числовой (целый) тип, то сначала получают двоично-десятичное представление числа 117 путем выделения второй тетрады из кода каждого символа и «упаковывания» полученных тетрад в промежуточную переменную: '117' → 0001 0001 0111 .

Накапливающий сумматор

Регистр

	С							A			
1)	0	0	0	1	0	0	0	1	0	1	1
2) сдв	0	0	0	0	<u>1</u>	0	0	0	<u>1</u>	0	1
3) кор			+		1	1	0	1	1	1	0
					0	1	0	1	1	0	0
4) сдв					0	0	1	0	<u>1</u>	1	0
5) кор			+		0	0	0	0	1	1	0
					0	0	1	0	1	0	1
6) сдв					0	0	0	1	0	1	0
7) сдв					0	0	0	0	<u>1</u>	0	1
8) кор			+		0	0	0	0	1	1	0
					0	1	1	1	0	1	0
9) сдв						0	0	1	1	1	0
10) сдв						0	0	0	1	1	1
11) сдв						0	0	0	0	1	1
12) С=0, конец											

Затем переводят
число из
двоично-
десятичного
представления в
двоичную
систему
счисления,
применяя
сравнительно
сложную
микропрограмму

Рисунок 1 – Пример реализации перевода числа
 $117_{10} = 1110101_2$, представленного в двоично-десятичном
 коде, в двоичную систему счисления (потребовалось 11 тактов)

Над числами, представленными в двоичной системе счисления (в двоичном коде) производятся действия. Чтобы вывести результат этих действий на экран или печать, его нужно перевести из двоичного в двоично-десятичный код (микропрограмма реализуется за 11 тактов для приведенного ниже примера), а затем получить коды символов для цифр числа (т.е. нужно выполнить обратные преобразования).

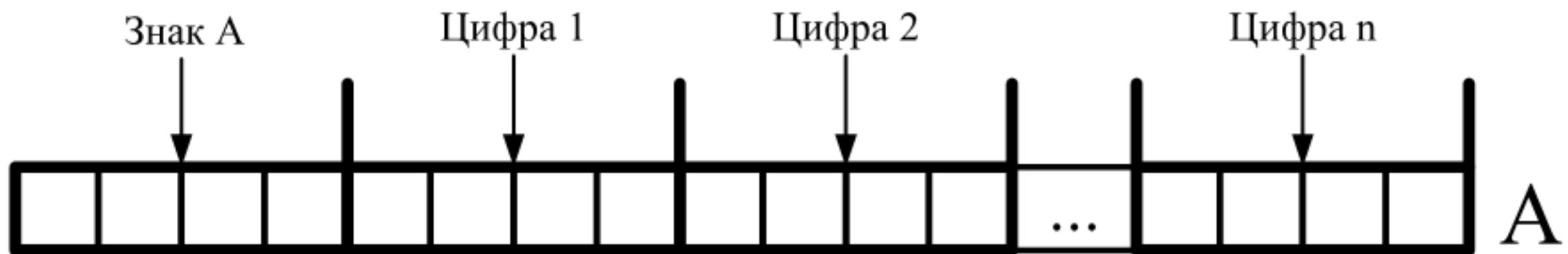
С										А									
										1 1 1 0 1 0 1 1)									
										1 1 1 0 1 0 1 2) сдв									
										1 1 1 1 0 1 0 1 3) сдв									
										1 1 1 1 0 1 0 1 4) сдв 7>4									
										1 1 1 0 1 0 1 5) сдв									
										+ 0 1 1 0 0 0 0 6) кор									
										1 0 1 0 0 1 0 1 7) сдв 9>4									
										1 0 1 0 0 1 0 1 8) сдв									
										+ 0 0 0 0 0 1 1 0 9) кор									
										0 1 0 1 1 0 0 0 1 8>4, 5>4									
										1 0 1 1 0 0 0 1 10) сдв									
										+ 0 1 1 0 0 1 1 0 11) кор									
0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	12) A=0

Рисунок 1 – Пример реализации перевода двоичного числа
 $1110101_2 = 117_{10}$ в двоично-десятичный код

Если над большим количеством десятичных чисел нужно выполнять арифметические (например, экономические) расчеты (четыре арифметические операции), то можно выполнять эти действия сразу над двоично-десятичными кодами чисел, не переводя числа в двоичное представление и обратно.

Учитывая, что микропрограммы перевода сравнительно сложные (занимают больше 10 тактов), экономия времени будет существенной!

Алгоритмы выполнения операций десятичной арифметики в АЛУ



+359 0000 0011 0101 1001

Число битов в числе должно быть кратно 4, поэтому под знак отводится 4 двоичных разряда, как и под каждую десятичную цифру числа.

Сложение двух десятичных чисел сводится к последовательной выработке сумм вида:

$$P_i \cdot C_i = A_i + B_i + P_{i+1} \quad , \text{ где}$$

A_i и B_i – четырехразрядные коды i -ых десятичных цифр слагаемых;

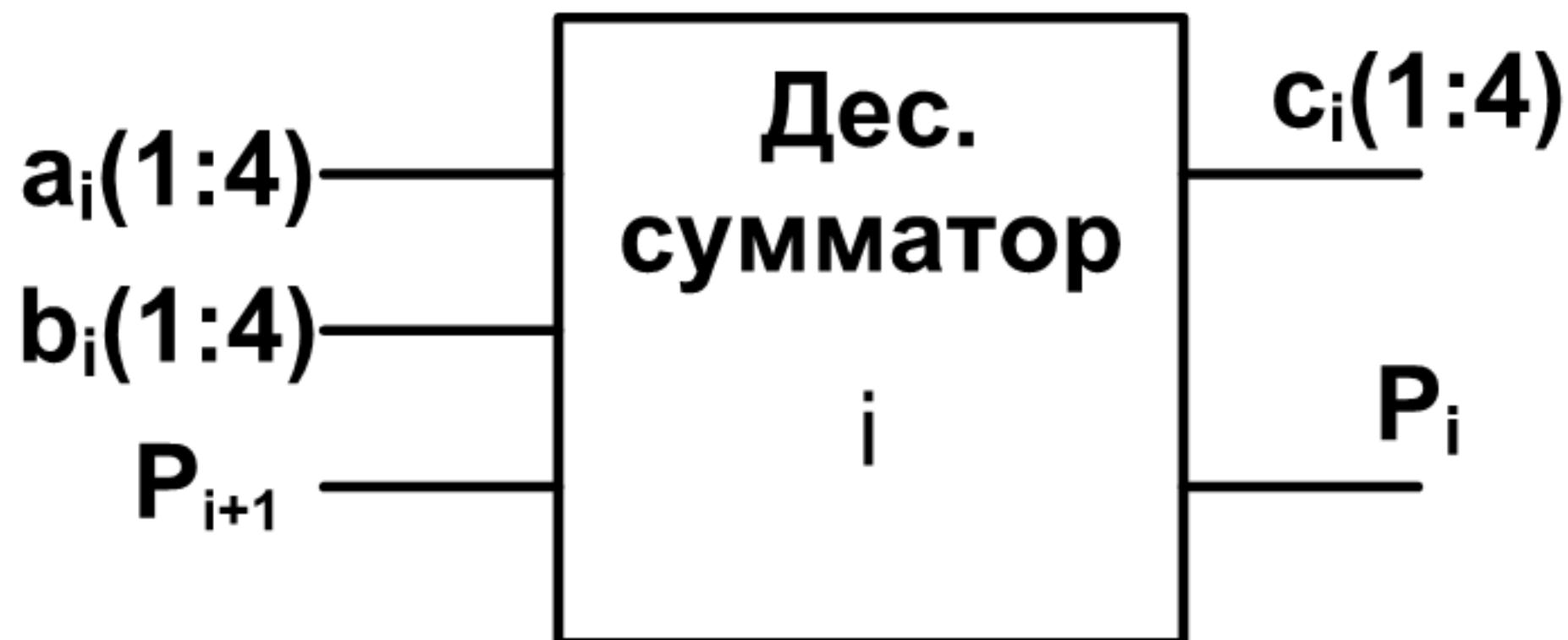
P_{i+1} – десятичный перенос (перенос с весом 10) из предыдущего младшего $(i+1)$ -го десятичного разряда суммы;

P_i – десятичный перенос из i -го разряда в следующий старший разряд суммы;

C_i – 4-х разрядный двоичный код десятичной цифры суммы

$$A_i + B_i + P_{i+1} \leq 19;$$

Одноразрядный десятичный сумматор



синтезировать не будем!

Операции над десятичными числами, представленными в двоично-десятичном формате (ДДФ), могут быть выполнены на двоичном сумматоре.

Дано: А, В – десятичные числа,
представленные в ДДФ,
 $A > 0, B > 0.$

Вычислить: С=А+В.

Основная задача: вычисление
очередной цифры суммы C_i .

$$P_i \cdot C_i = A_i + B_i + P_{i+1}$$

Возможны несколько случаев:

$$1) 0 \leq P_i \cdot C_i \leq 9$$

В этом случае ответ – правильный!

+	0	0	1	0
	0	0	1	1
<u>θ</u>	0	1	0	1
<u>P_i</u>			<u>C_i</u>	

2) $10 \leq P_i \cdot C_i \leq 15$

<u>+</u>	1	0	0	1
	0	0	1	1
<hr/>				
<u>θ</u>	1	1	0	0
<u>p_i</u>			<u>c_i</u>	

<u>+</u>	1	1	0	0
	0	1	1	0
<hr/>				
<u>I</u>	0	0	1	0
<u>p_i</u>			<u>c_i</u>	

2) $16 \leq P_i \cdot C_i \leq 19$

+	1	0	0	0
	1	0	0	1
<u>I</u>	0	0	0	1
p_i		c_i		

+	0	0	0	1
	0	1	1	0
<u>I</u>	0	1	1	1
p_i		c_i		

Если получили $P_i \cdot C_i \geq 10$, то нужна коррекция: $C_i = C_i + 6$!!!

На практике сначала делают коррекцию

$$C_i = A_i + 6, \text{ а затем}$$

$$C_i = C_i + B_i \quad (*)$$

Признак лишней коррекции (1 случай) – отсутствие переноса в (*). Тогда делается еще одна коррекция:

$$C_i = C_i - 6, \text{ что эквивалентно}$$

$$C_i = C_i + 10,$$

$$\begin{array}{r}
 0110 \\
 + 1010 \\
 \hline
 10000
 \end{array}$$

Алгоритм сложения

1) Вычисление суммы

$$C = A_1 A_2 A_3 \dots A_n + 6 6 6 \dots 6;$$

2) Вычисление суммы

$$C=C+B;$$

3) Если перенос из старшей тетрады $p_1=1$,
переполнение: $F_{\text{пп}}:=1$; \rightarrow конец;

4) Формирование корректирующего слагаемого
 $K=K_1 K_2 K_3 \dots K_n$, где

$$K_i=0000, \text{если } p_i=1;$$

$$K_i=1010, \text{если } p_i=0;$$

5) $C:=C+K$; (суммирование с разорванными
цепями межтетрадного переноса)

Конец

$$134+591=725$$

	+	0	0	0	1		0	0	1	1		0	1	0	0
		0	1	1	0		0	1	1	0		0	1	1	0
<hr/>															
		0	1	1	1		1	0	0	1		1	0	1	0

$$1) \mathbf{C} = \mathbf{A}_1 \mathbf{A}_2 \mathbf{A}_3 \dots \mathbf{A}_n + 6 \ 6 \ 6 \dots 6;$$

$$134 + 591 = 725$$

$$2) C=C+B;$$

3) Если перенос из старшей тетрады $p_1=1$,
переполнение: $F_{пп}:=1$; \rightarrow конец;

$$134+591=725$$

	+ 0 0 0 1	0 0 1 1	0 1 0 0
	0 1 1 0	0 1 1 0	0 1 1 0
+ 0 1 1 1	1 0 0 1	1 0 1 0	
0 1 0 1	1 0 0 1	0 0 0 1	
P₁= 0 1 1 0 1	0 0 1 0	1 0 1 1	
III 1 0 1 0	0 0 0 0	1 0 1 0	
нет			

- 4) Формирование корректирующего слагаемого $K = K_1 K_2 K_3 \dots K_n$, где
 $K_i = 0000$, если $r_i = 1$;
 $K_i = 1010$, если $r_i = 0$;

	+	0	0	0	1		0	0	1	1		0	1	0	0		
		0	1	1	0		0	1	1	0		0	1	1	0		
<hr/>																	
	+	0	1	1	1		1	0	0	1		1	0	1	0		
		0	1	0	1		1	0	0	1		0	0	0	1		
<hr/>																	
	+	1	1	0	1		+	0	0	1	0		+	1	0	1	1
		1	0	1	0			0	0	0	0			1	0	1	0
<hr/>																	
		0	1	1	1			0	0	1	0			0	1	0	1
			7					2						5			

5) **C:=C+K;** (суммирование с разорванными цепями межтетрадного переноса)

Последнее сложение должно осуществляться с разорванными цепями межтетрадного переноса. Недостаток: усложнение схемы двоичного сумматора за счет необходимости управлять цепями межтетрадного переноса.

$$134 + 591 = 725$$

+	0	0	0	1		0	0	1	1		0	1	0	0
	0	1	1	0		0	1	1	0		0	1	1	0
+	0	1	1	1		1	0	0	1		1	0	1	0
	0	1	0	1		1	0	0	1		0	0	0	1
+	1	1	0	1	↔--→	0	0	1	0	↔--→	1	0	1	1
	1	0	0	1		1	1	1	1		1	0	0	1
														1
	0	1	1	1		0	0	1	0		0	1	0	1
		7					2					5		

Уменьшили на 1 все цифры (десятичные) корректирующего слагаемого, тем самым обеспечив перенос через все тетрады, выходной перенос замкнули на входной перенос двоичного сумматора.

Преимущество: не нужно управлять цепями межтетрадного переноса.

$$325 + 741 = 1066$$

	+ 0 0 1 1	0 0 1 0	0 1 0 1
	0 1 1 0	0 1 1 0	0 1 1 0
	<hr/>	<hr/>	<hr/>
	+ 1 0 0 1	1 0 0 0	1 0 1 1
	0 1 1 1	0 1 0 0	0 0 0 1
	<hr/>	<hr/>	<hr/>
P ₁ =	(1) 0 0 0 0	(2) 1 1 0 0	(3) 1 1 0 0
III	1 1 1 1	1 0 0 1	1 0 0 1
III	<hr/>	<hr/>	<hr/>
	0 0 0 0	0 1 1 0	0 1 1 0
	0	6	6

Переполнение может возникнуть только при сложении чисел с одинаковыми знаками.

Вычитание

$$C = A - B = A + (-B) = A + B_{\text{доп}}$$

$$B_{\text{доп}} = 10^n - B$$

$$\begin{array}{r}
 & 1 & 0000 & 0000 & 0000 \\
 - & B_1 & B_2 & B_3 \\
 \hline
 & B_1^* & B_2^* & B_3^*
 \end{array}$$

$$B_i^* = 10 - B_i \quad \text{для } i=n;$$

$$B_i^* = 9 - B_i \quad \text{для } i=1, 2, \dots, n-1.$$

$$C_i = A_i + 6 + B_i^* = A_i + 6 + 9 - B_i = A_i + 15 - B_i$$

$$C_i = A_i + B_i \text{обр}$$

Алгоритм вычитания

$$1) \quad C = A_1 A_2 \dots A_n + \overline{B_1} \overline{B_2} \dots \overline{B_n} + 1;$$

2) Формирование корректирующего слагаемого
 $K = K_1 K_2 K_3 \dots K_n$, где

$$K_i = 1111, \text{если } p_i = 1;$$

$$K_i = 1001, \text{если } p_i = 0;$$

3) $C := C + K$; (выходной перенос сумматора замкнут на входной)

Конец

$$725 - 135 = 590$$

$$B_3^* = 10 - B_3$$

Если $A > B$, то $C = A - B$;

Если $A < B$, то $C = 10^n - (B - A)$ (результат будет получен в дополнительном коде)

$$135 - 725 = -590$$

+	0	0	0	1	0	0	1	1	0	1	0	1		
	1	0	0	0	1	1	0	1	1	0	1	0		
												1		
(0)	1	0	1	0	(1)	0	0	0	1	(1)	0	0	0	0
	1	0	0	1		1	1	1	1		1	1	1	1
													1	
0	1	0	0		0	0	0	1		0	0	0	0	
	4					1				0				

$$1000 - 410 = 590$$

Обобщенный алгоритм

(позволяет получить результат в прямом коде)

- 1) Знаку суммы присвоить знак первого слагаемого: **ЗнС:=ЗнА.**
- 2) Если знаки слагаемых одинаковы **ЗнA=ЗнB**, то выполняется микропрограмма сложения. Перенос из старшего разряда сумматора сигнализирует о ситуации переполнения.
- 3) Если знаки разные, то выполняется микропрограмма вычитания **A-B.**
- 4) Если перенос из старшего разряда сумматора равен 0, то знак суммы меняется на противоположный и вычисляется разность **B-A.**

Алгоритмы умножения и деления рассмотрены в [1].

В современных микропроцессорах (микроконтроллерах) нет аппаратного блока десятичной арифметики, поэтому рассмотренные алгоритмы (при необходимости) можно реализовать программно.

Например, на 8-битном (ширина шины данных) микроконтроллере КМ1816ВЕ51 можно программно организовать обработку многоразрядных десятичных чисел, используя для представления одного числа несколько байтов данных, а каждый байт – для хранения двух десятичных цифр числа.

При выполнении операций над такими числами удобно использовать команду десятичной коррекции аккумулятора.

Аккумулятором называется регистр АЛУ, в который загружается первый операнд и в котором после выполнения операции «появляется» результат.

Группа команд арифметических операций

Название команды	Мнемокод	КОП	Операция
Сложение аккумулятора с регистром ($n = 0 \div 7$)	ADD A, R _n	00101пг	(A) $\leftarrow (A) + (R_n)$
Сложение аккумулятора с прямоадресуемым байтом	ADD A, ad	00100101	(A) $\leftarrow (A) + (ad)$
Сложение аккумулятора с байтом из РПД ($i = 0,1$)	ADD A, @R _i	0010011i	(A) $\leftarrow (A) + ((R_i))$
Сложение аккумулятора с константой	ADD A, #d	00100100	(A) $\leftarrow (A) + \# d$
Сложение аккумулятора с регистром и переносом	ADDC A, R _n	00111пг	(A) $\leftarrow (A) + (R_n) + (C)$
Сложение аккумулятора с прямоадресуемым байтом и переносом	ADDC A, ad	00110101	(A) $\leftarrow (A) + (ad) + (C)$
Сложение аккумулятора с байтом из РПД и переносом	ADDC A, @R _i	0011011i	(A) $\leftarrow (A) + ((R_i)) + (C)$
Сложение аккумулятора с константой и переносом	ADDC A, #d	00110100	(A) $\leftarrow (A) + \# d + (C)$
Десятичная коррекция аккумулятора	DA A	11010100	Если $(A_{0-3}) > 9 \vee ((AC) = 1)$, то $(A_{0-3}) \leftarrow (A_{0-3}) + 6$, затем если $(A_{4-7}) > 9 \vee ((C) = 1)$, то $(A_{4-7}) \leftarrow (A_{4-7}) + 6$

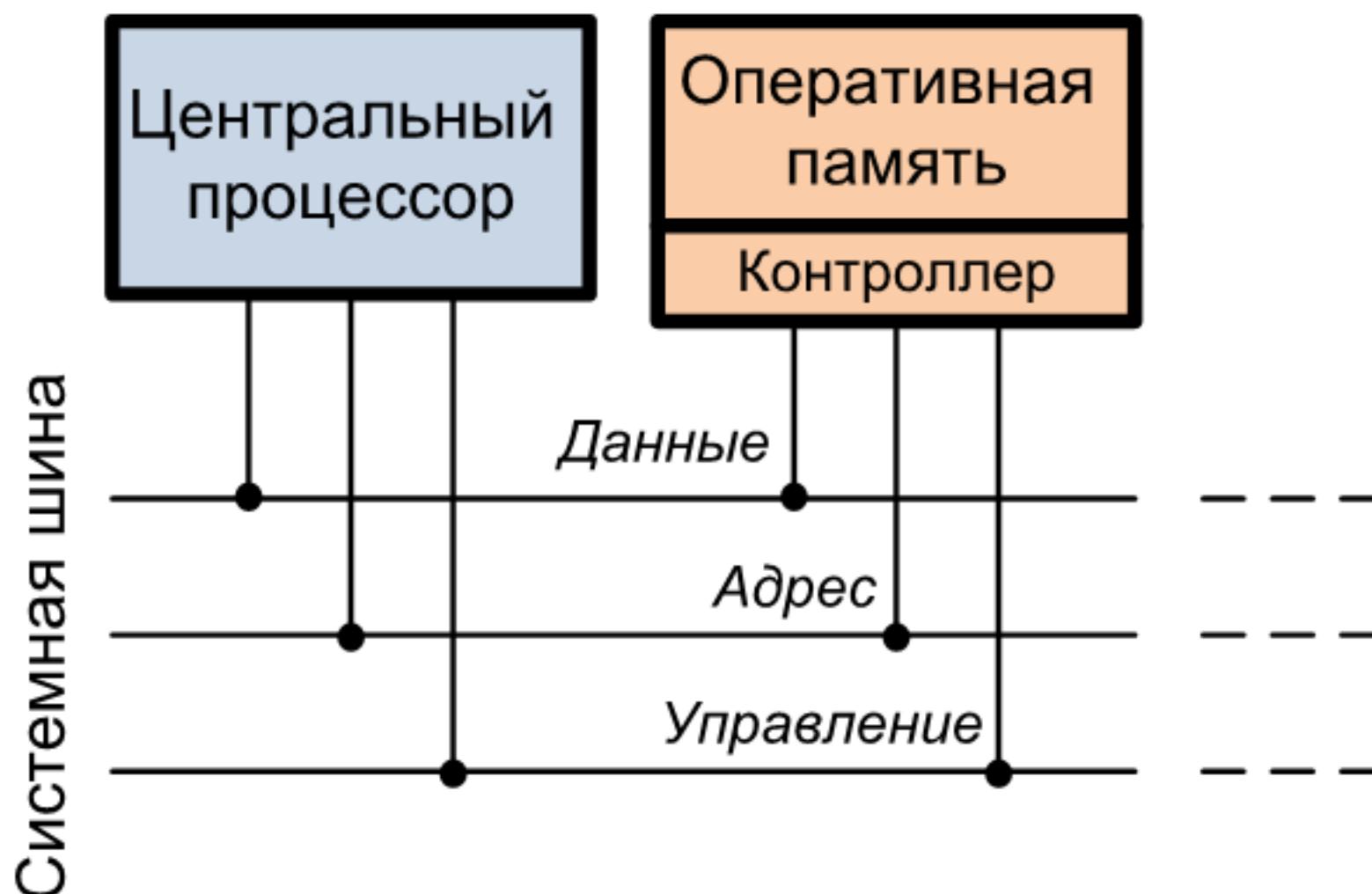
AC – флаг, в котором запоминается перенос из младшей тетрады аккумулятора, C – флаг, в котором запоминается выходной перенос аккумулятора (перенос из старшей тетрады). Старший разряд аккумулятора – седьмой.

Если $(A_{0-3}) > 9 \vee ((AC) = 1)$, то $(A_{0-3}) \leftarrow (A_{0-3}) + 6$,
затем
если $(A_{4-7}) > 9 \vee ((C) = 1)$, то $(A_{4-7}) \leftarrow (A_{4-7}) + 6$

Взаимодействие устройств ЭВМ в процессе выполнения машинной команды (рабочий цикл процессора)

В каждый момент времени компьютер выполняет определенную программу. Программа – последовательность действий (команд), направленная на преобразование некоторых входных данных в выходные (результат).

Команды и данные программы, исполняемой в текущий момент времени хранятся в оперативной памяти (ОП). Выполняет команды процессор.



Вопрос: В каком виде хранятся данные и команды?

Откуда процессор знает, где в ОП находится очередная команда и данные для нее?

Представление данных

(подробно рассматривали в теме АЛУ)

Зн $2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$

0	1	1	0	1	1	0	1
---	---	---	---	---	---	---	---

$$= 1 \cdot 2^0 + 0 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 + 0 \cdot 2^4 + 1 \cdot 2^5 + 1 \cdot 2^6 = \\ = 1 + 4 + 8 + 32 + 64 = +109$$


 0 – число положительное,
 1 – число отрицательное

Числа представляются в ЭВМ в двоичной системе счисления (форматы со знаком с фиксированной и с плавающей точкой).

Символы представляются их двоичными кодами.

Двоичные вектора (над которыми производятся поразрядные логические операции) представляются последовательностями битов, для хранения которых, как правило, используются беззнаковые целые типы.

Представление команд

Команда – это приказ компьютеру на выполнение какой-либо операции, например, операции сложения двух чисел (операндов), которые хранятся в оперативной памяти.

Можно представить команду в следующем формате:

Адрес
первого
операнда
в ОП

111...011

010...001

001...011

Код
операции

Адрес
второго
операнда
в ОП

Результат, как правило, помещается в память на место первого операнда

Каждый процессор исполняет определенный набор команд, который называется системой команд процессора. В этот набор входят арифметические и логические команды, команды пересылки информации между регистрами процессора и оперативной памятью, команды передачи управления, команды ввода-вывода и др.

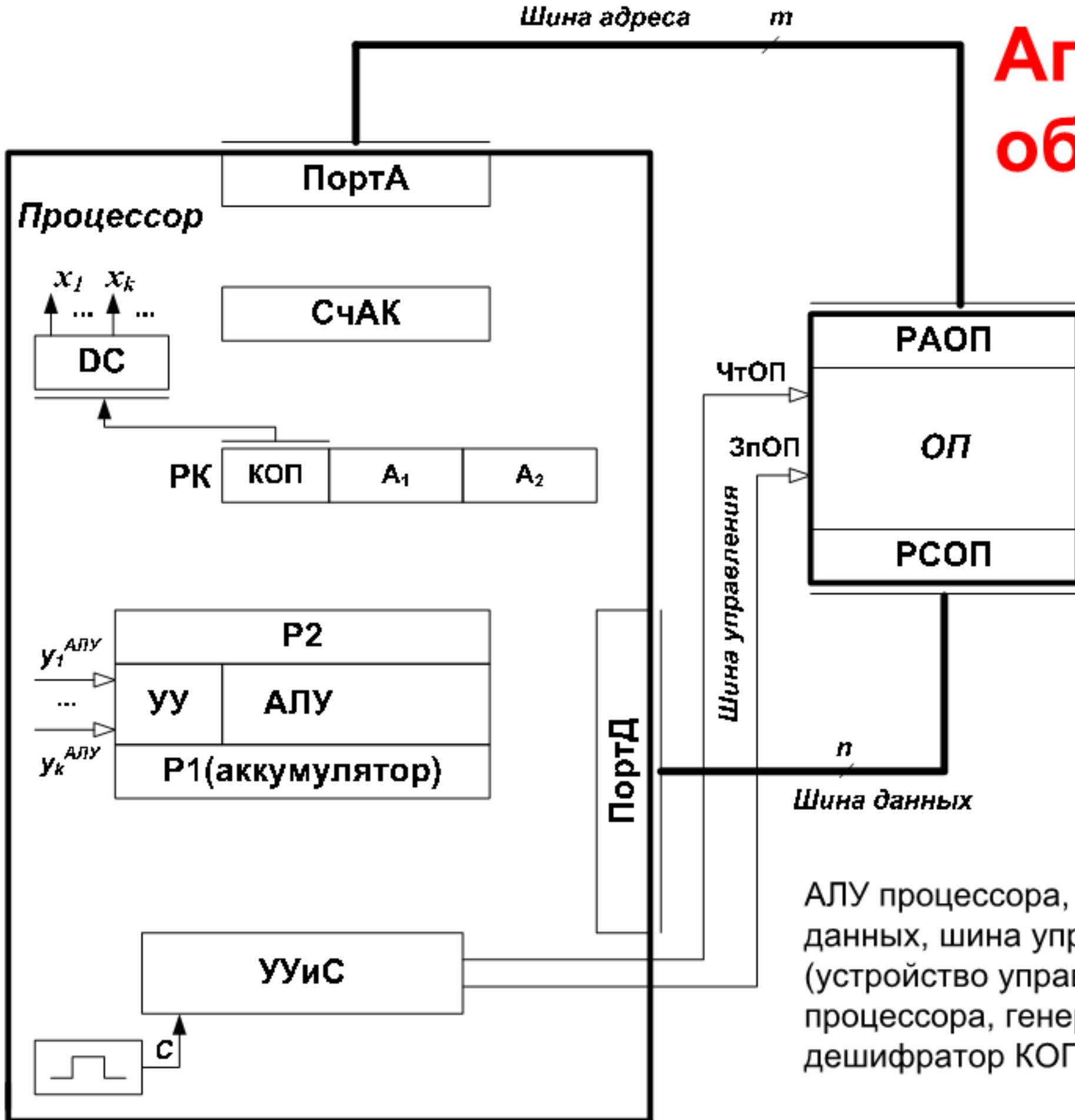
Данные о системе команд конкретного процессора можно получить из соответствующих справочников.

В качестве примера приведем фрагмент описания системы команд микропроцессора (микроконтроллера) КМ1816ВЕ51.

Таблица 3.13. Группа команд арифметических операций

Название команды	Мнемокод	КОП	Т	Б	Ц	Операции
Сложение аккумулятора с регистром ($l = 0 \div 7$)	ADD A, R _l	0010111	1	1	1	(A) \leftarrow (A) + (R _l)
Сложение аккумулятора с прямоадресуемым байтом	ADD A, ad	00100101	3	2	1	(A) \leftarrow (A) + (ad)
Сложение аккумулятора с байтом из РПД ($l = 0..1$)	ADD A, @R _l	00100111	1	1	1	(A) \leftarrow (A) + ((R _l))
Сложение аккумулятора с константой	ADD A, #d	00100100	2	2	1	(A) \leftarrow (A) + # d
Сложение аккумулятора с регистром и переносом	ADDC A, R _l	0011111	1	1	1	(A) \leftarrow (A) + (R _l) + (C)
Сложение аккумулятора с прямоадресуемым байтом и переносом	ADDC A, ad	00110101	3	2	1	(A) \leftarrow (A) + (ad) + (C)
Сложение аккумулятора с байтом из РПД и переносом	ADDC A, @R _l	00110111	1	1	1	(A) \leftarrow (A) + ((R _l)) + (C)
Сложение аккумулятора с константой и переносом	ADDC A, #d	00110100	2	2	1	(A) \leftarrow (A) + # d + (C)
Десятичная коррекция аккумулятора	DA A	11010100	1	1	1	Если (A _{0..3}) \geq 9 \vee ((AC) = -1), то (A _{0..3}) \leftarrow (A _{0..3}) + 6, затем если (A _{4..7}) \geq 9 \vee ((C) = -1), то (A _{4..7}) \leftarrow (A _{4..7}) + 6
Вычитание из аккумулятора регистра и заема	SUBB A, R _l	1001111	1	1	1	(A) \leftarrow (A) - (C) - (R _l)
Вычитание из аккумулятора прямоадресуемого байта и заема	SUBB A, ad	10010101	3	2	1	(A) \leftarrow (A) - (C) - (ad)
Вычитание из аккумулятора байта РПД и заема	SUBB A, @R _l	10010111	1	1	1	(A) \leftarrow (A) - (C) - ((R _l))
Вычитание из аккумулятора константы и заема	SUBB A, d	10010100	2	2	1	(A) \leftarrow (A) - (C) - # d
Инкремент аккумулятора	INC A	00000100	1	1	1	(A) \leftarrow (A) + 1
Инкремент регистра	INC R _l	0000111	1	1	1	(R _l) \leftarrow (R _l) + 1
Инкремент прямоадресуемого байта	INC ad	00000101	3	2	1	(ad) \leftarrow (ad) + 1
Инкремент байта в РПД	INC @R _l	00000111	1	1	1	((R _l)) \leftarrow ((R _l)) + 1
Инкремент указателя данных	INC DPTR	10100011	1	1	2	(DPTR) \leftarrow (DPTR) + 1
Декремент аккумулятора	DEC A	00010100	1	1	1	(A) \leftarrow (A) - 1
Декремент регистра	DEC R _l	0001111	1	1	1	(R _l) \leftarrow (R _l) - 1
Декремент прямоадресуемого байта	DEC ad	00010101	3	2	1	(ad) \leftarrow (ad) - 1
Декремент байта в РПД	DEC @R _l	00010111	1	1	1	((R _l)) \leftarrow ((R _l)) - 1
Умножение аккумулятора на регистр В	MUL AB	10100100	1	1	4	(B)(A) \leftarrow (A) \times (B)
Деление аккумулятора на регистр В	DIV AB	10000100	1	1	4	(A).(B) \leftarrow (A)/(B)

Аппаратное обеспечение



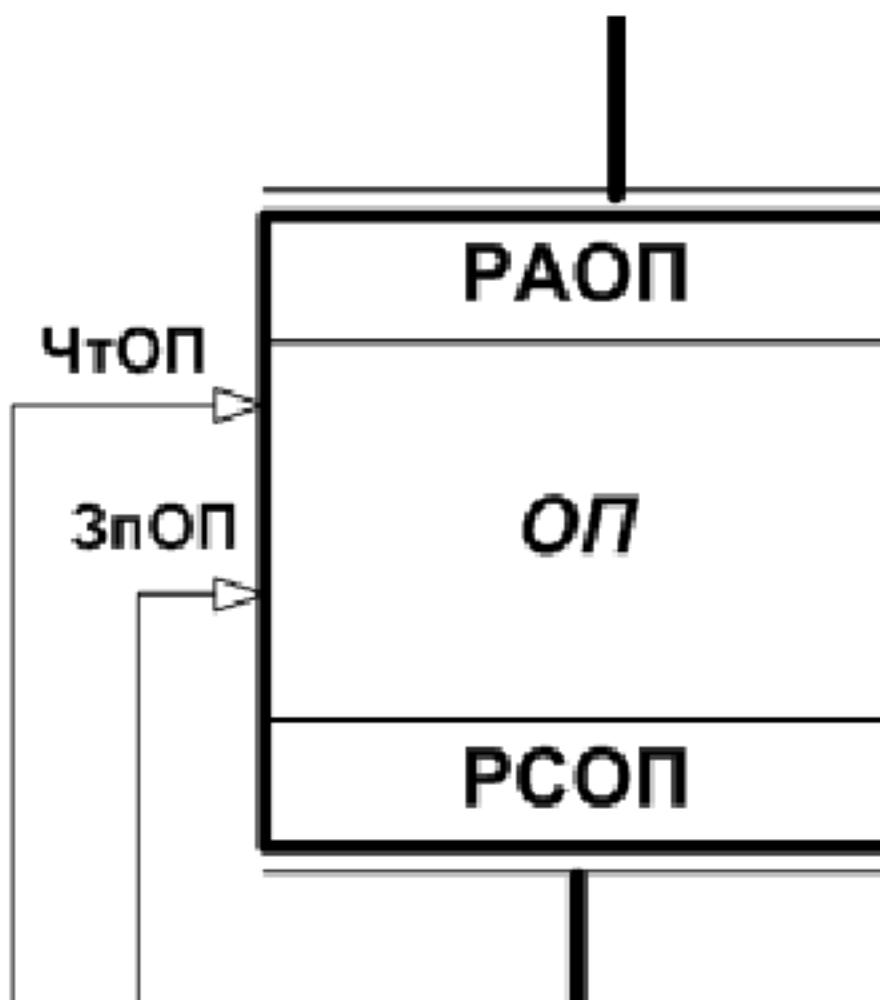
Регистры процессора:
 РК – регистр команд;
 ПортA – порт адреса;
 ПортD – порт данных;
 Р1 – аккумулятор АЛУ;
 Р2 – второй регистр АЛУ;
 СчАК – счетчик адреса команды.

Регистры оперативной памяти (ОП):
 РАОП – регистр адреса ОП;
 РСОП – регистр слова ОП.

АЛУ процессора, ОП, шина адреса, шина данных, шина управления, УУиС (устройство управления и синхронизации) процессора, генератор синхросерии С, дешифратор КОП (DC).

- 1)УУиС – устройство управления и синхронизации (управляет работой всех других устройств);
- 2)СЧАК – счетчик адреса команды (длина определяется количеством битов адреса в системе, зависящим от объема оперативной памяти в байтах) хранит адрес очередной команды, которая должна быть считана из памяти и выполнена ;
- 3)РК – регистр команд (длина определяется максимальной длиной команды) хранит выбранную из памяти команду на всем протяжении ее выполнения процессором;
- 4)АЛУ, имеющее два регистра для операндов, один из которых является аккумулятором, т.е. регистром, в который загружается операнд, а потом в нем появляется результат выполнения операции в АЛУ (длина регистров определяется максимальной длиной операнда);
- 5) ПортА подключен к шине адреса, которая, в свою очередь подключена к регистру адреса оперативной памяти РАОП (длина равна количеству битов (линий) шины адреса).
- 6)ПортД подключен к шине данных, которая, в свою очередь, подключена к регистру слова оперативной памяти РСОП (длина равна количеству битов (линий) шины данных).

... (продолжим список позже).



Чтобы осуществить считывание данных из ОП процессор должен выставить на шину адреса (в РАОП) адрес слова ОП, в котором хранятся данные, и выработать управляющий сигнал ЧтОП. Считанные данные появляются в РСОП и поступают по шине данных в порт данных.

Чтобы осуществить запись данных в ОП, процессор должен выставить на шину адреса (в РАОП) адрес слова ОП, в которое нужно записать данные, на шину данных (в РСОП) – сами данные, и выработать управляющий сигнал (ЗпОП).

Рабочий цикл процессора на примере выполнения двухадресной арифметической (или логической) команды.

Адрес
первого
операнда
в ОП

Рассматриваем
упрощенный вариант,
но достаточный для
понимания принципа
взаимодействия
процессора и ОП в
процессе выполнения
машинной команды.

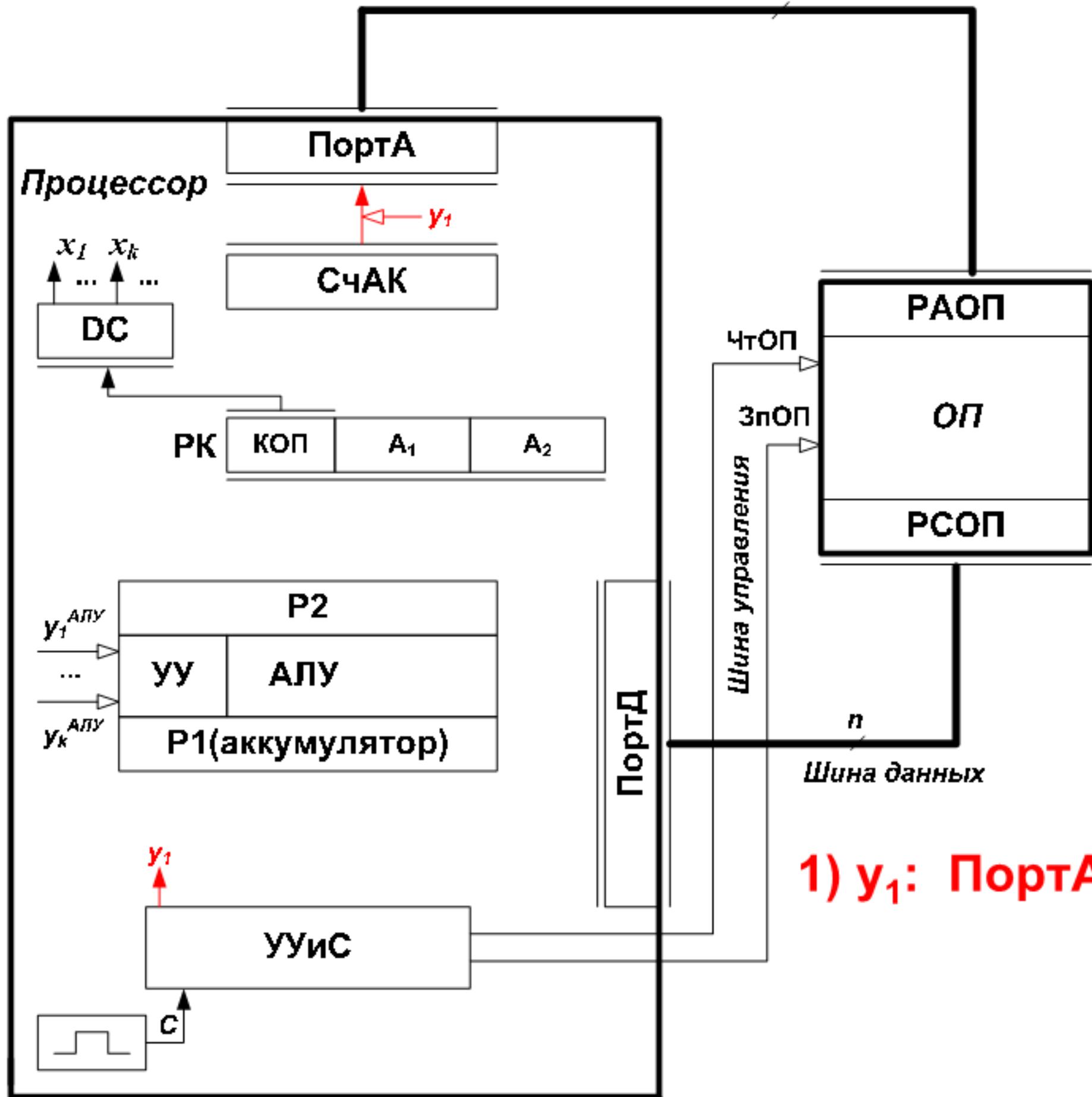
111...011	010...001	001...011
-----------	-----------	-----------

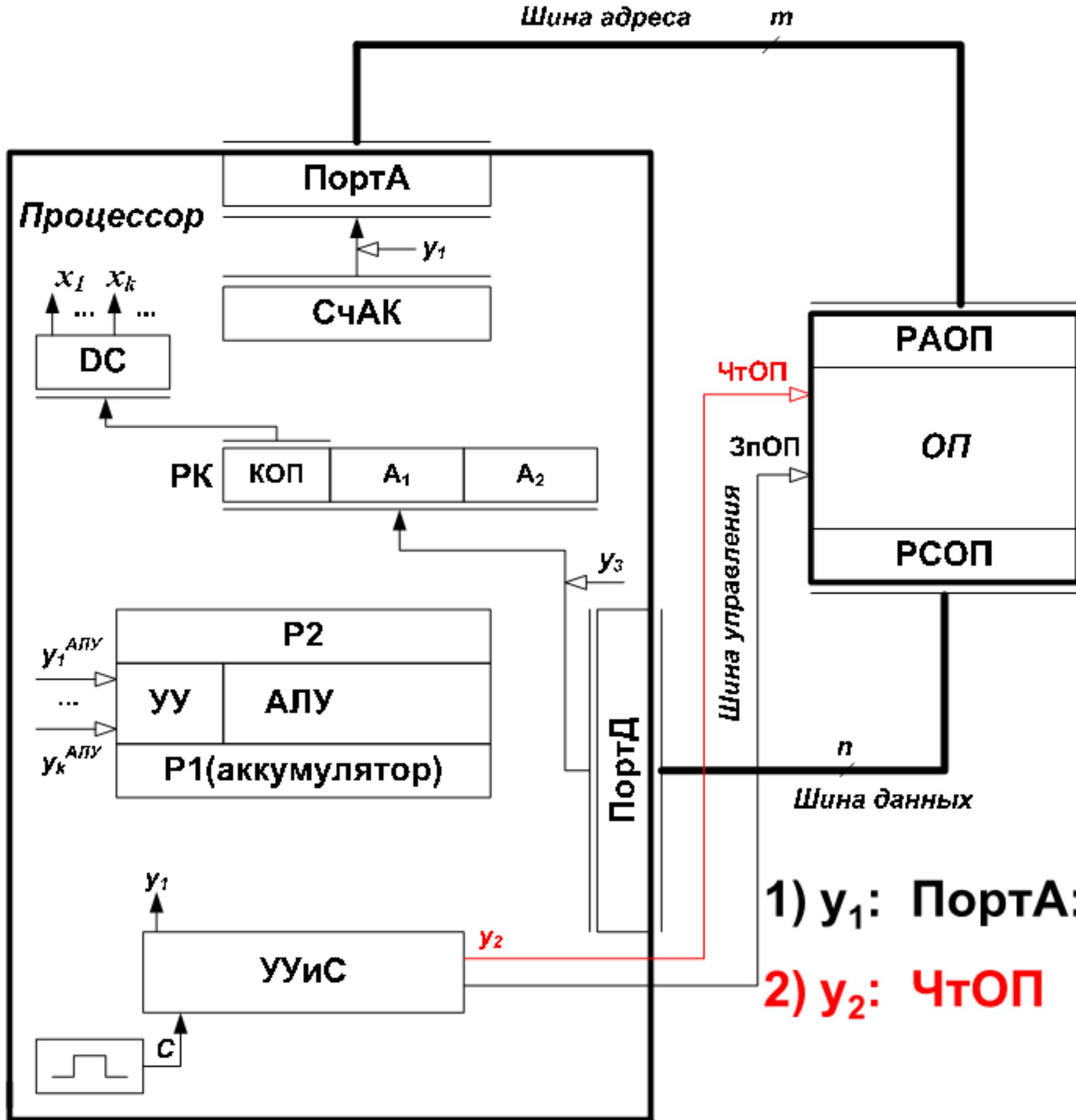
Код
операции

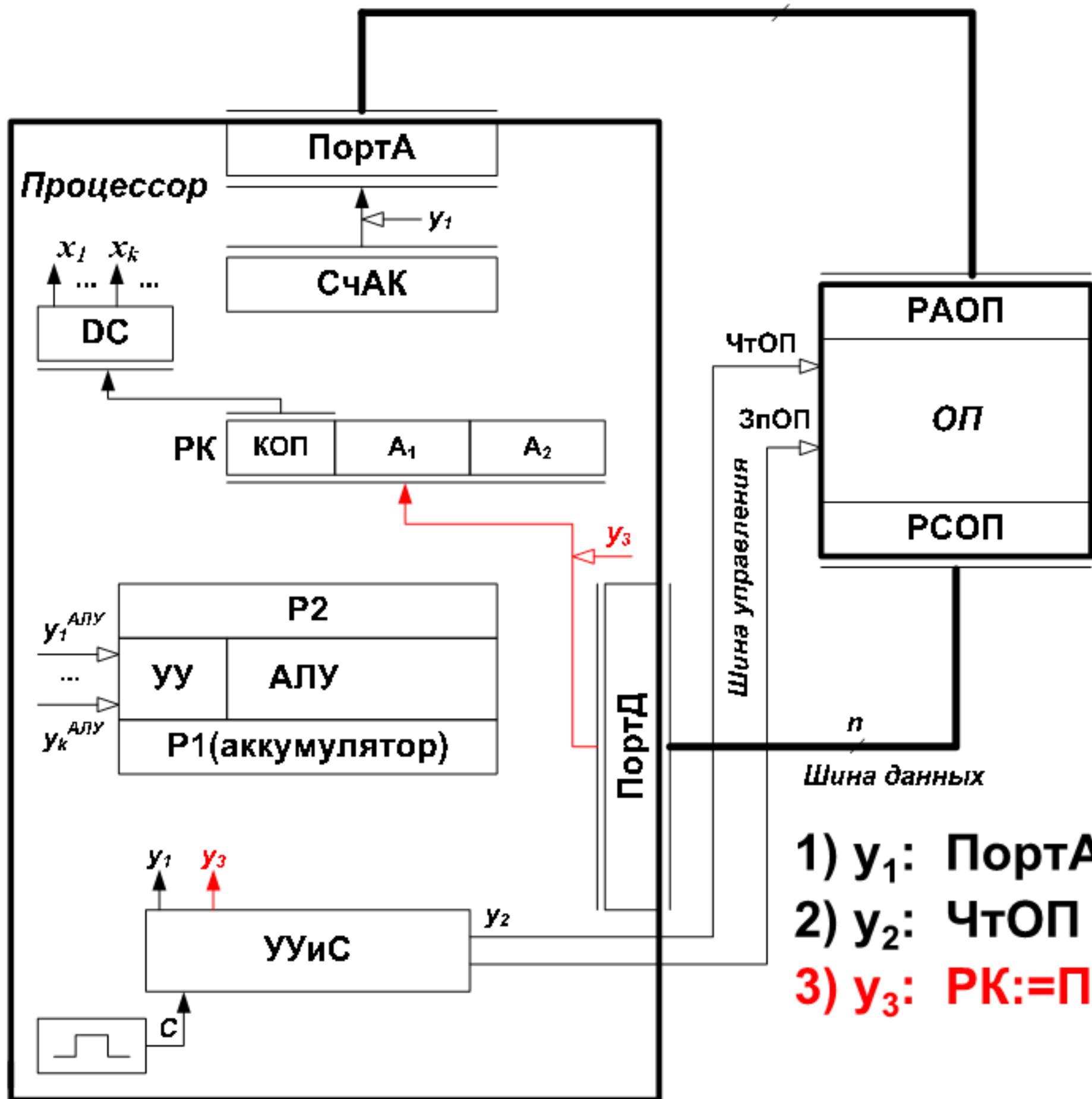
Адрес
второго
операнда
в ОП

1 этап.

Выборка команды из ОП





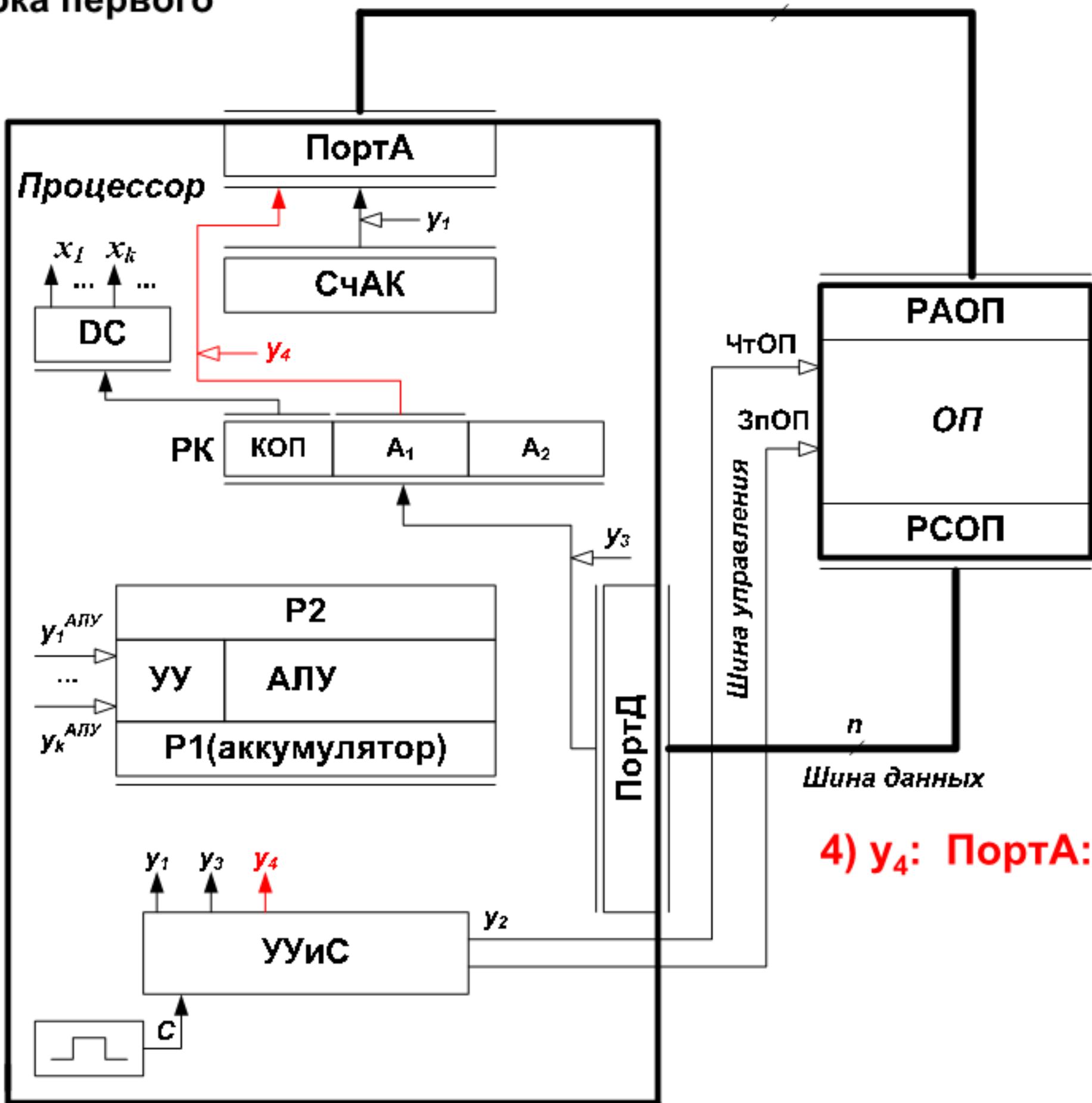


Рабочий цикл процессора

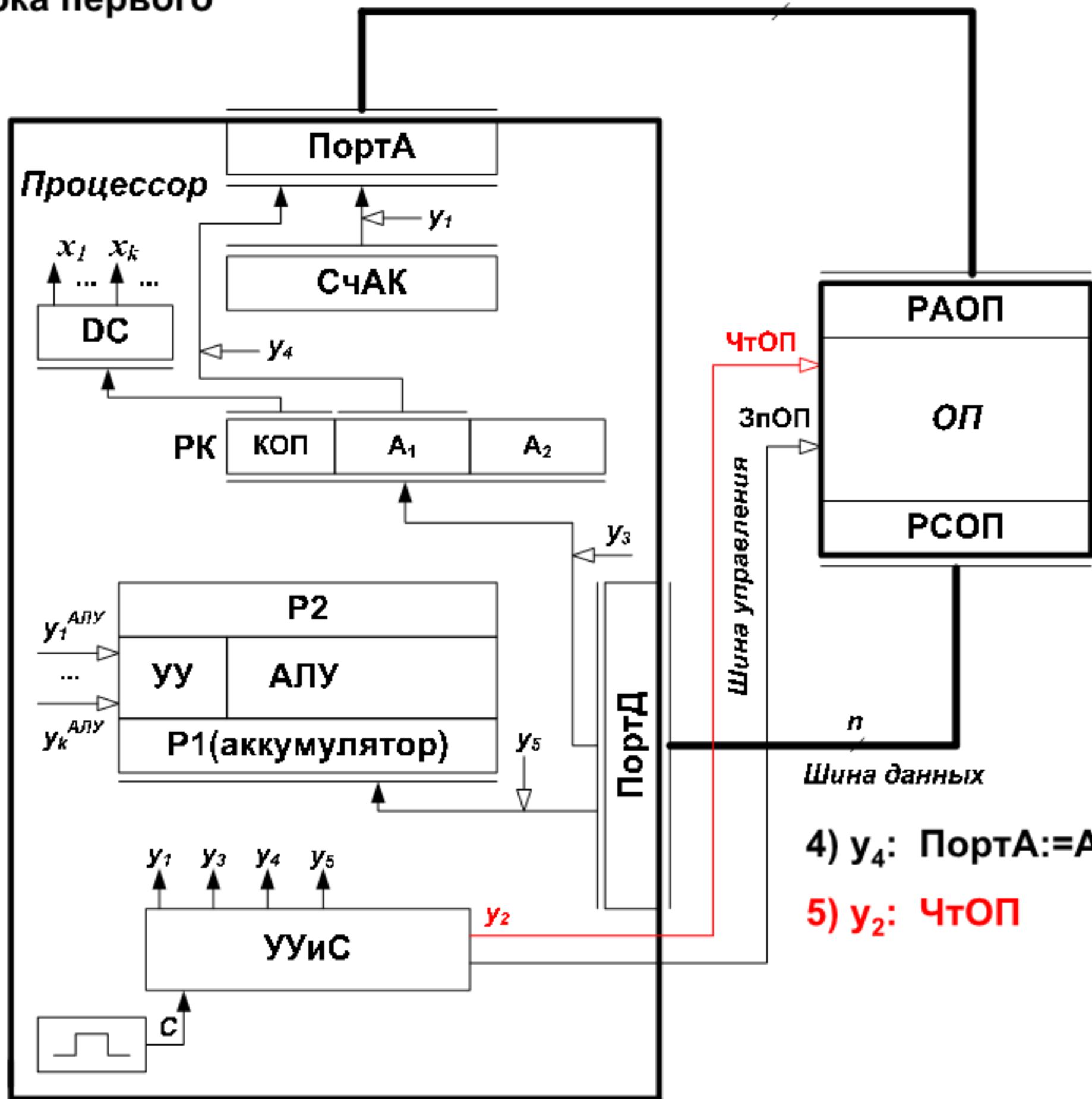
2 этап.

Выборка операндов из ОП

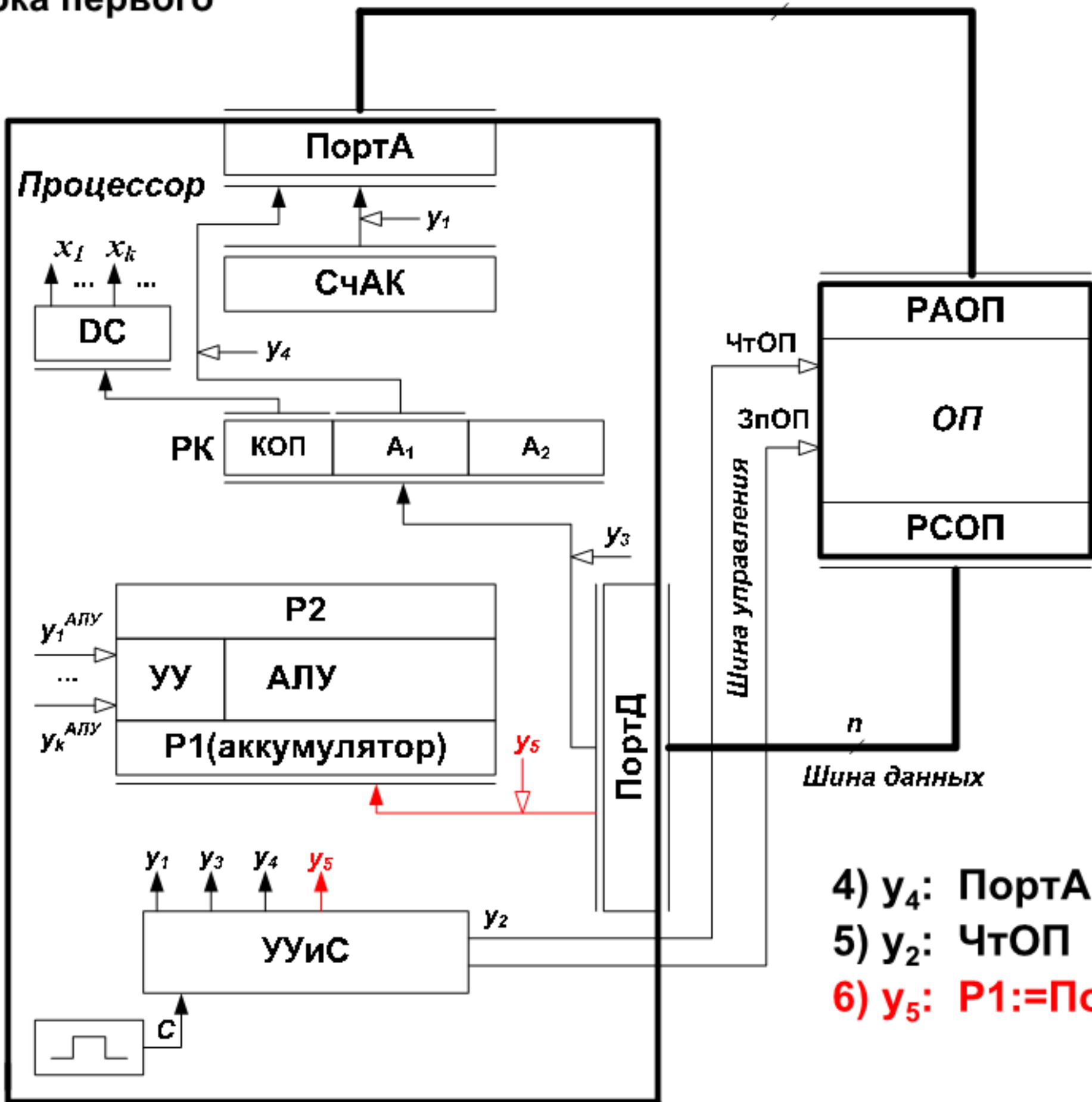
2.1. Выборка первого операнда



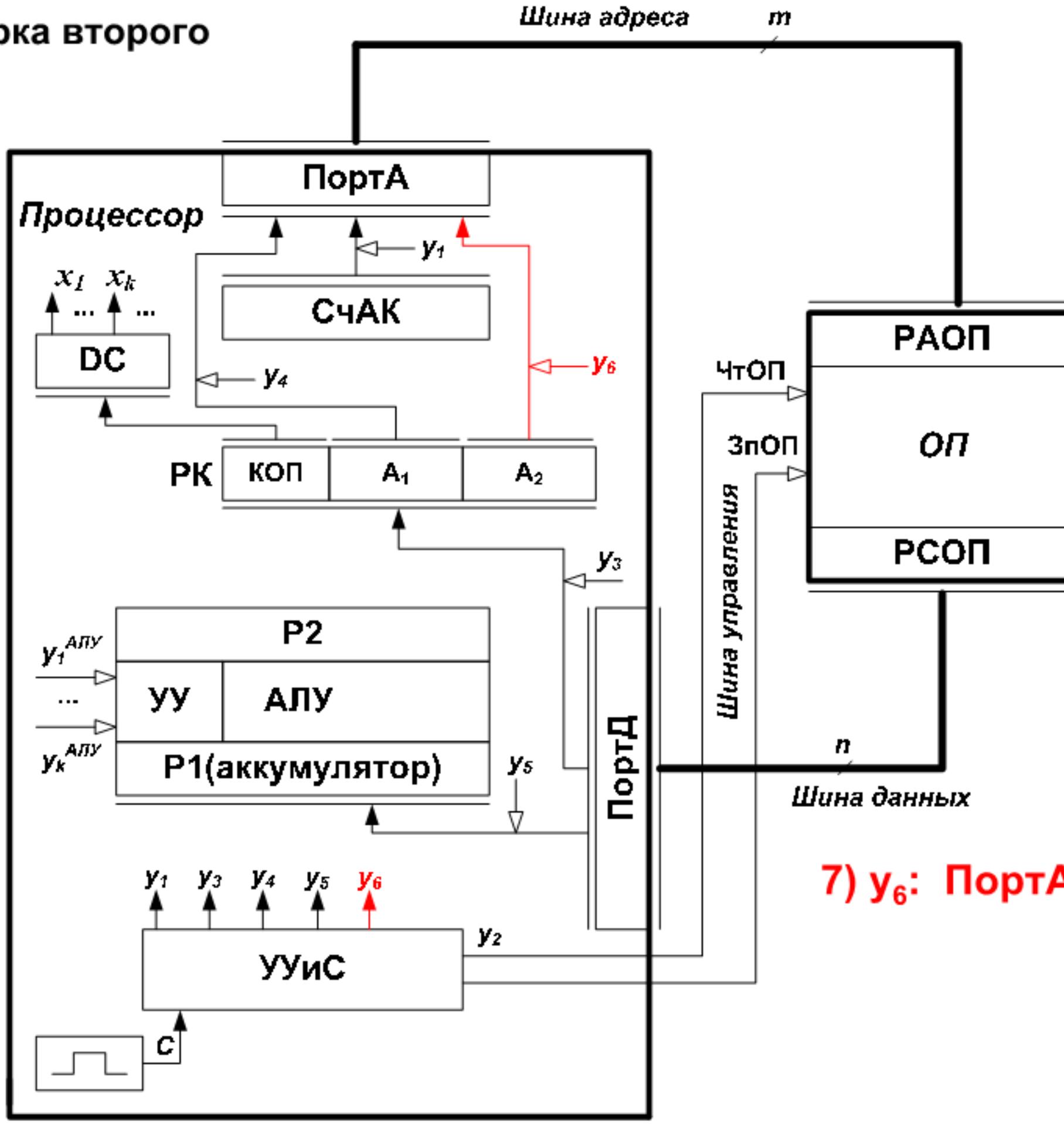
2.1. Выборка первого операнда



2.1. Выборка первого операнда

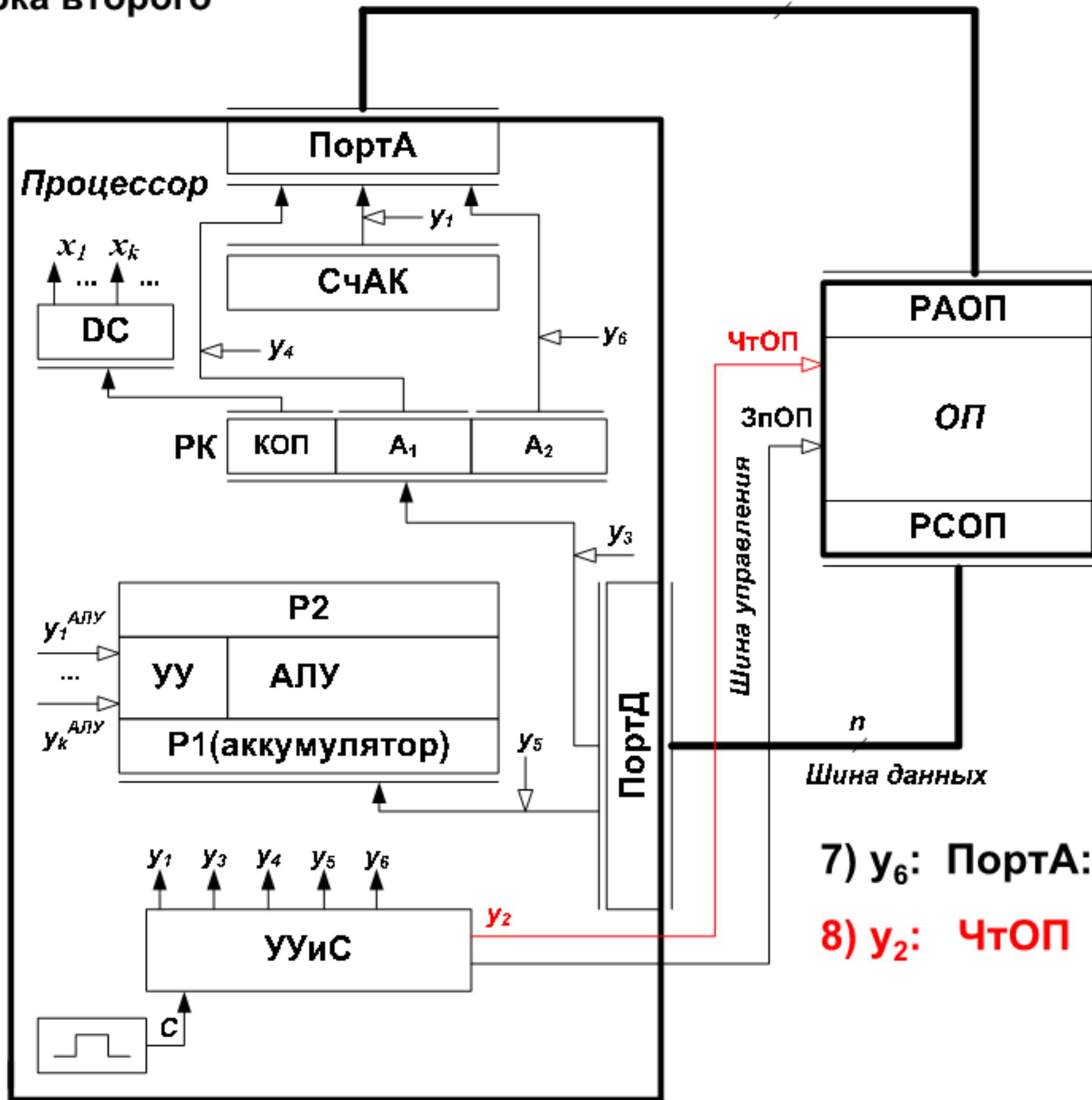


2.2. Выборка второго операнда



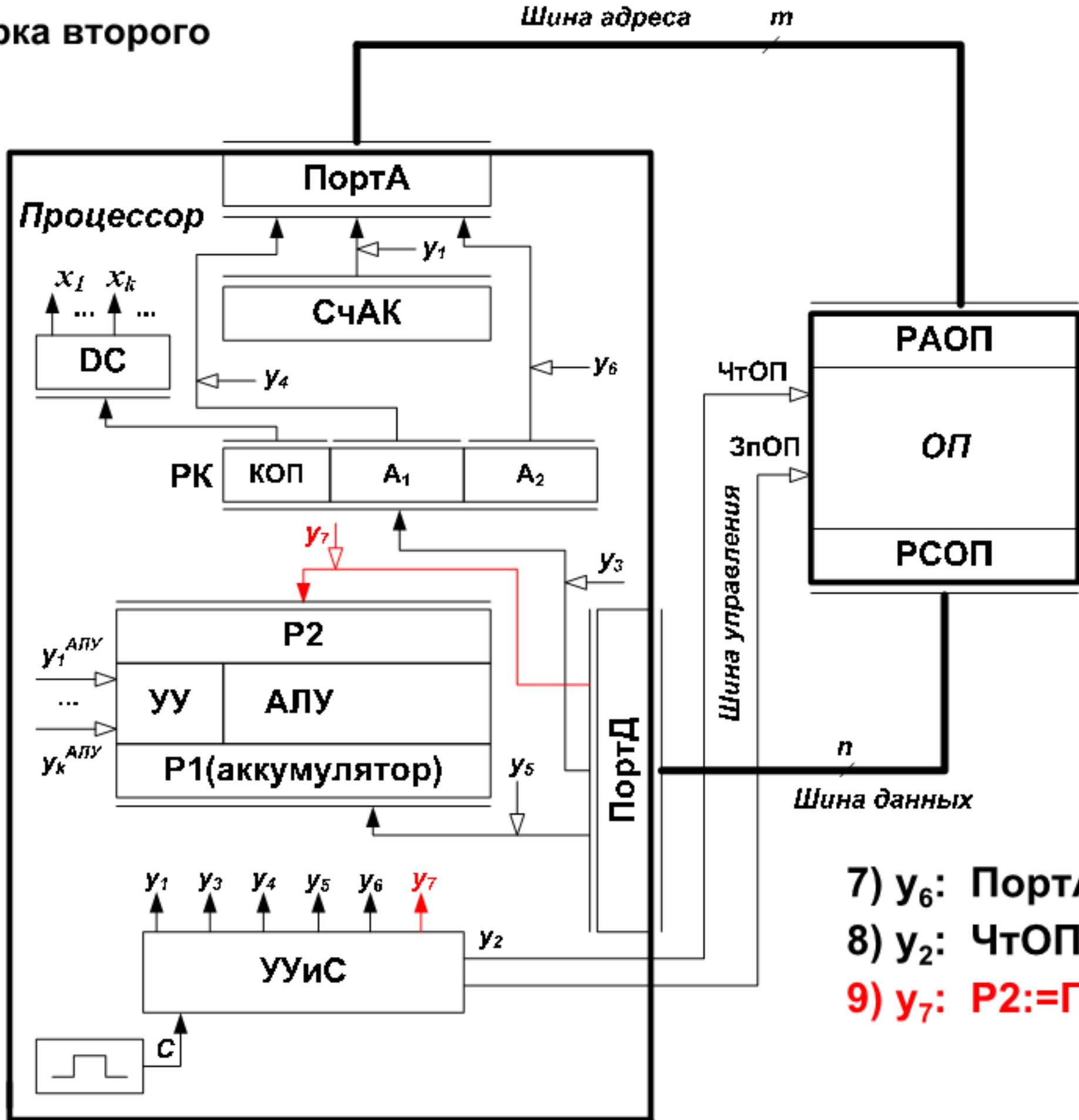
2.2. Выборка второго операнда

45



2.2. Выборка второго операнда

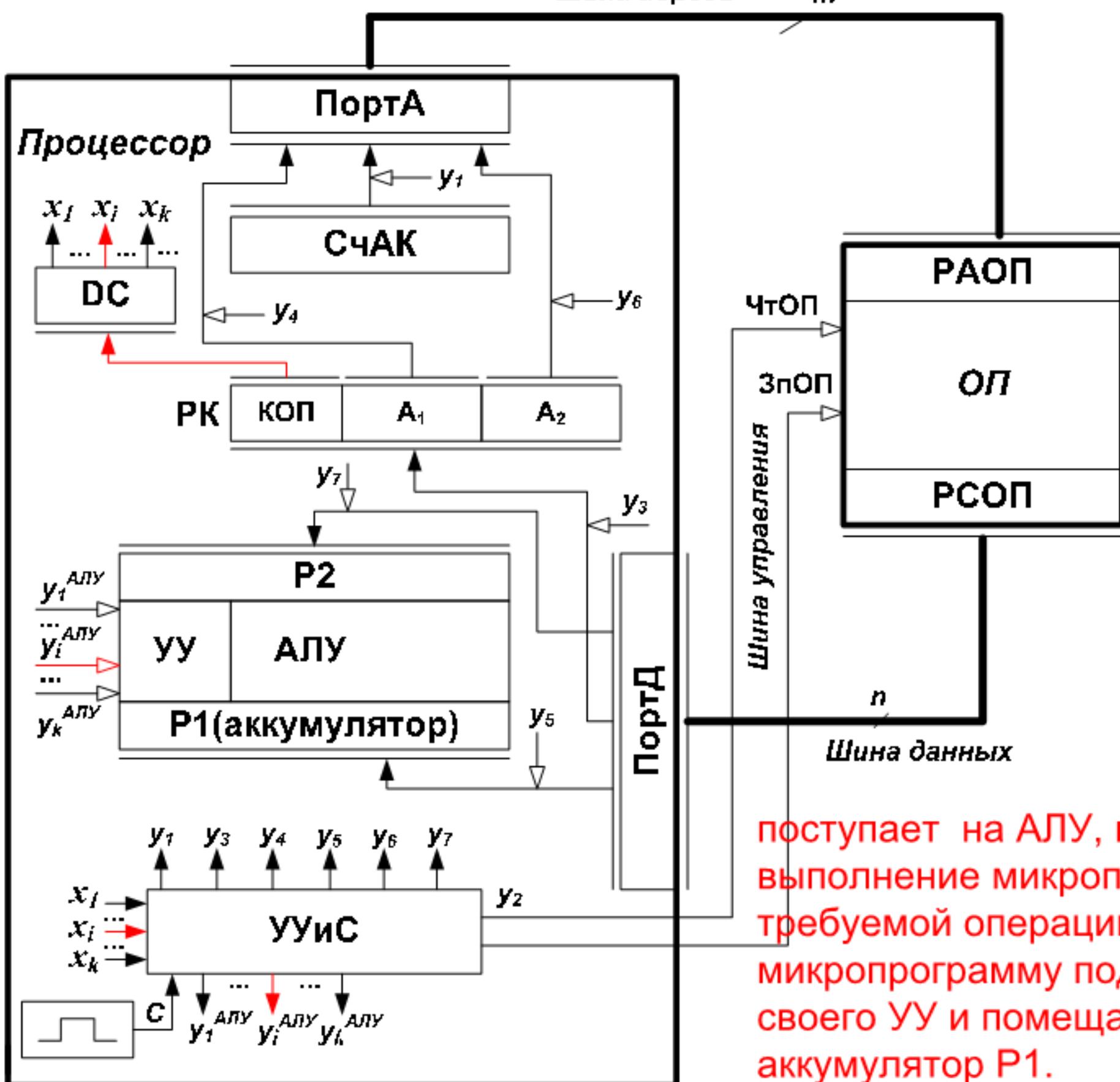
46



Рабочий цикл процессора

3 этап.

Выполнение операции в
АЛУ

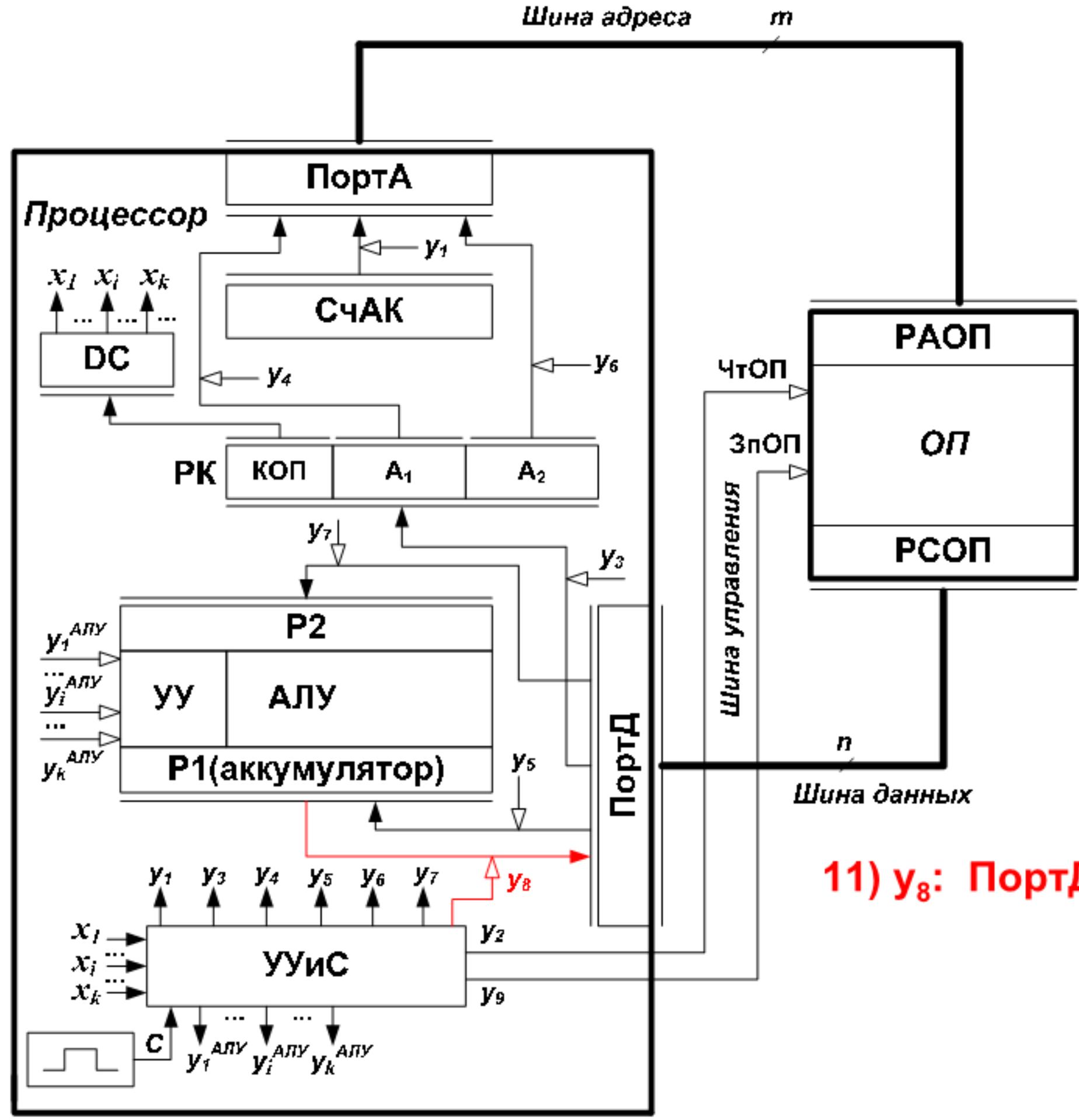


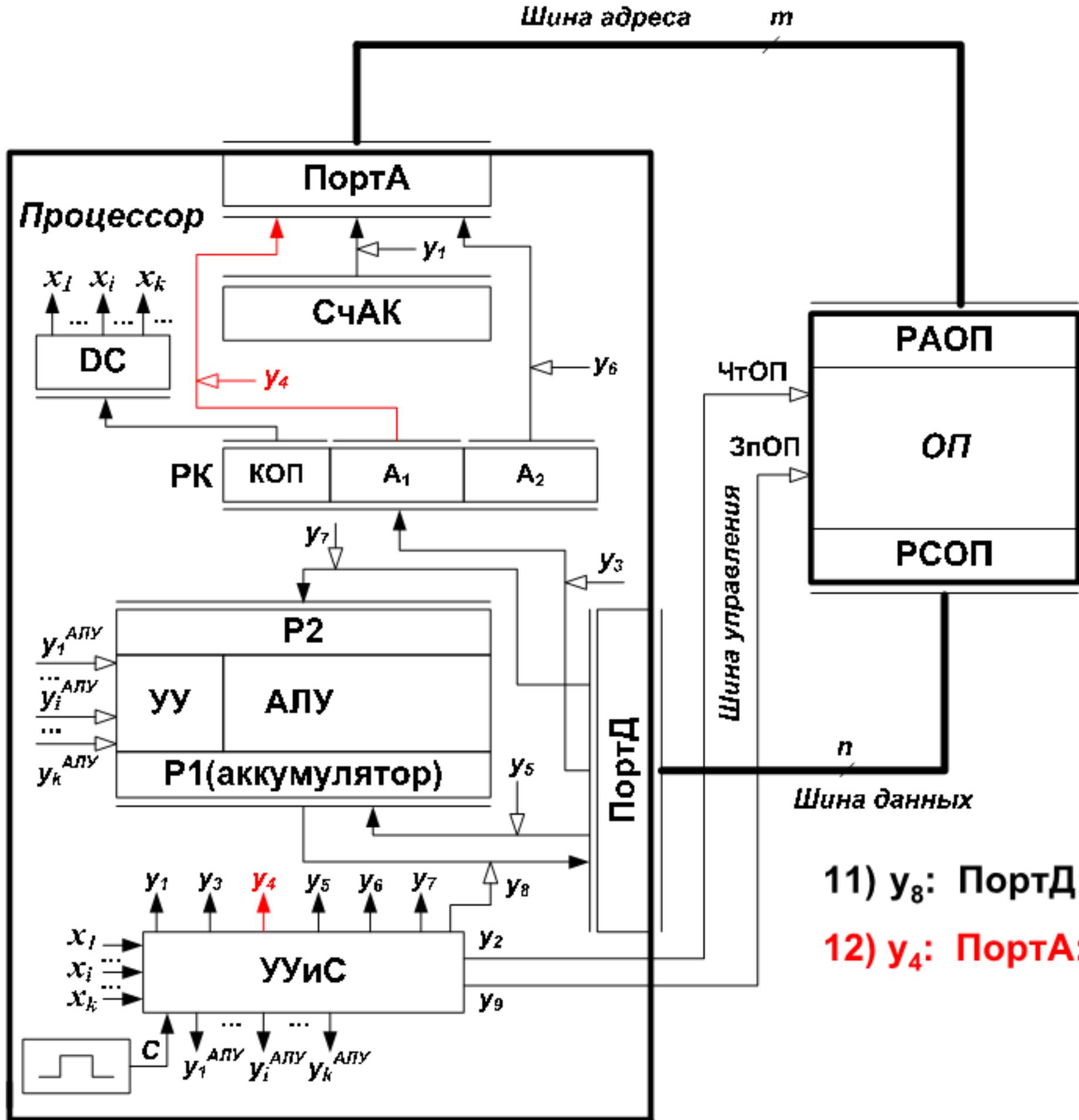
Значение в части РК, соответствующей КОП, расшифровывается на DC. На соответствующем i -ом выходе DC формируется осведомительный сигнал $x_i=1$. УУиС, проанализировав сигналы на своих входах x_i-x_k , в ответ на входной сигнал $x_i=1$ выдает выходной сигнал $y_i^{\text{ALU}}=1$. Этот управляющий сигнал поступает на АЛУ, в котором инициирует выполнение микропрограммы для требуемой операции. АЛУ выполняет микропрограмму под управлением своего УУ и помещает результат в аккумулятор Р1.

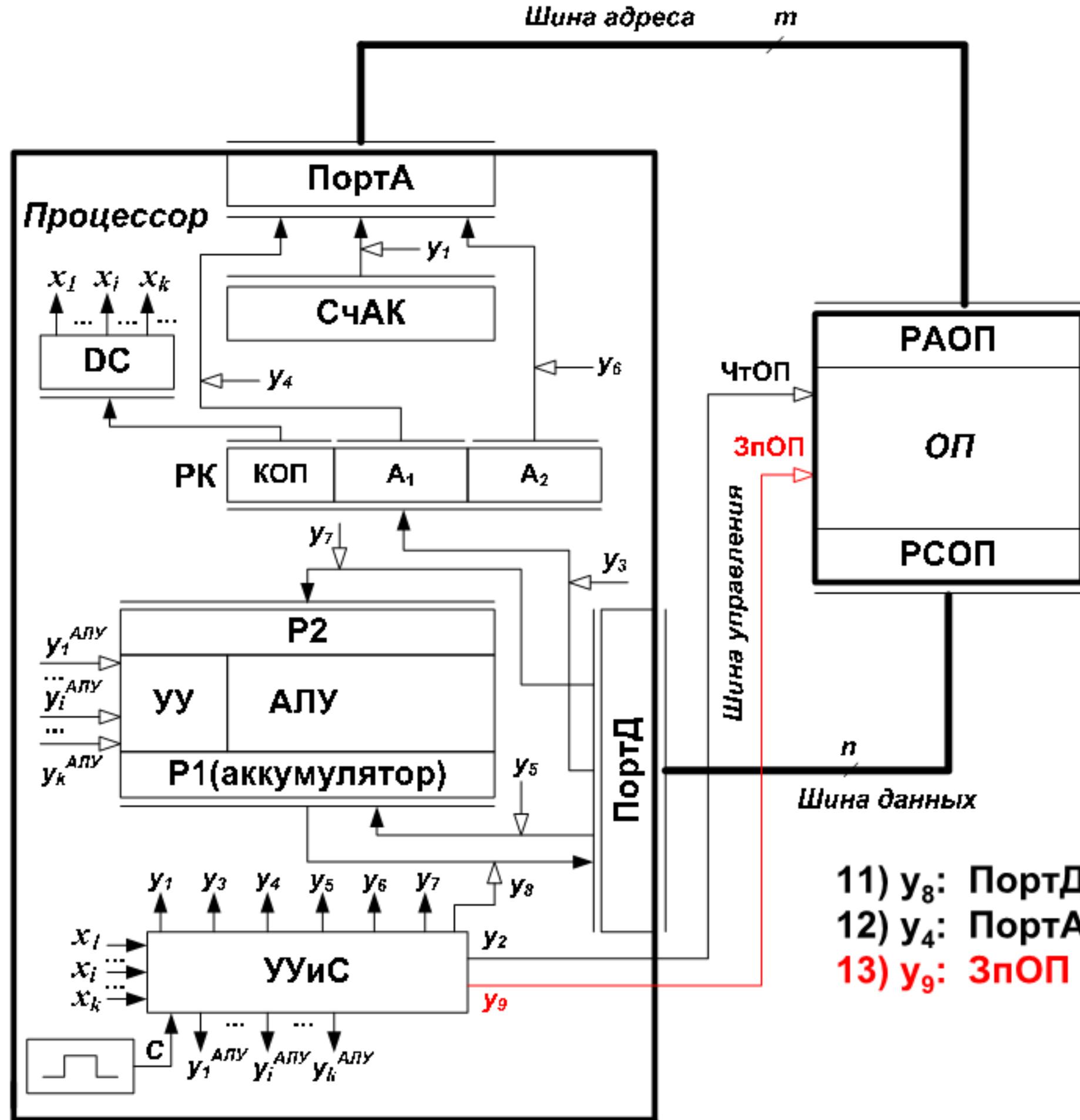
Рабочий цикл процессора

4 этап.

Запись результата в ОП
(на место первого операнда)







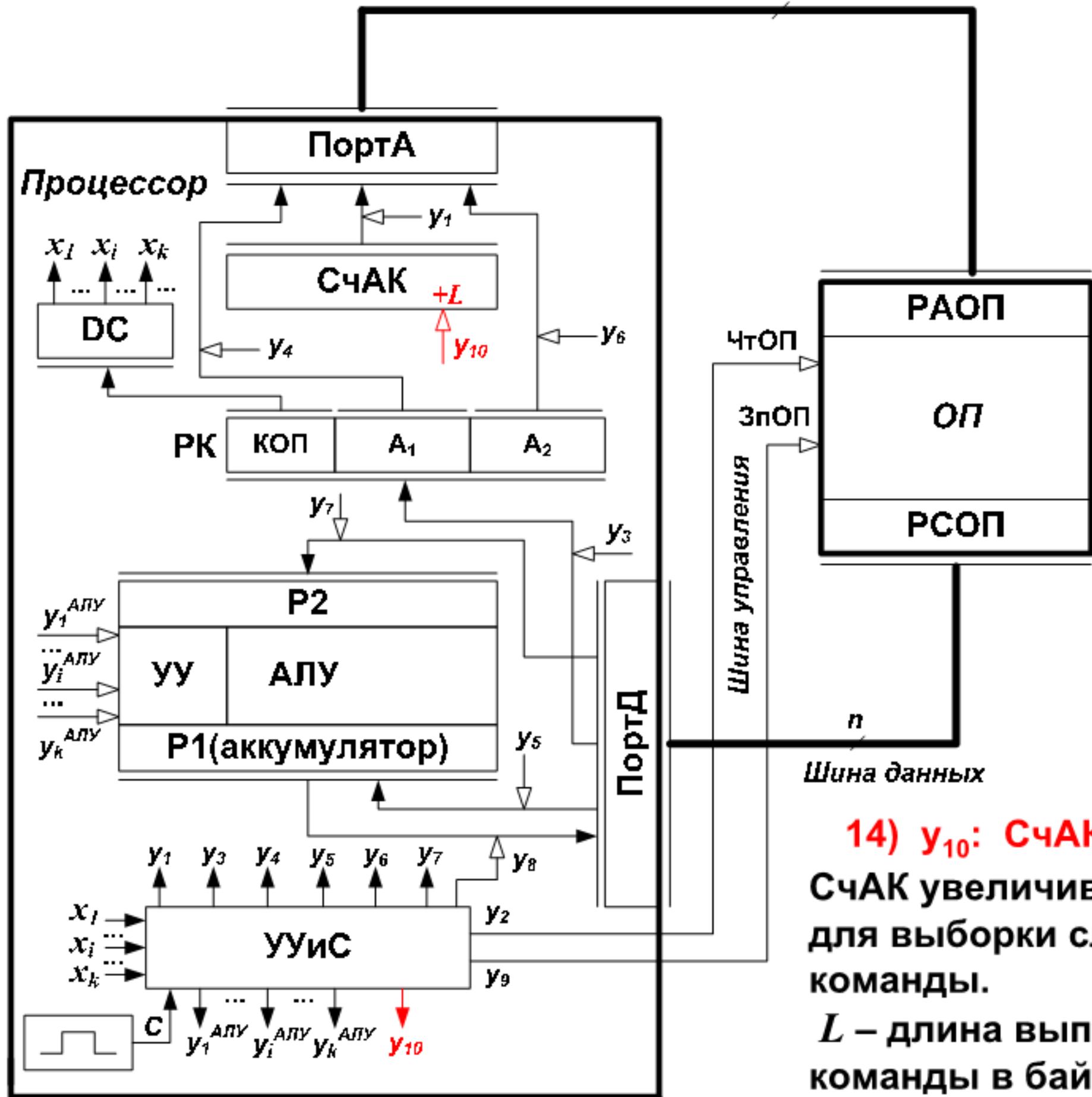
- 11) y_8 : ПортД:=P1
 12) y_4 : ПортA:=A1
 13) y_9 : ЗпОП

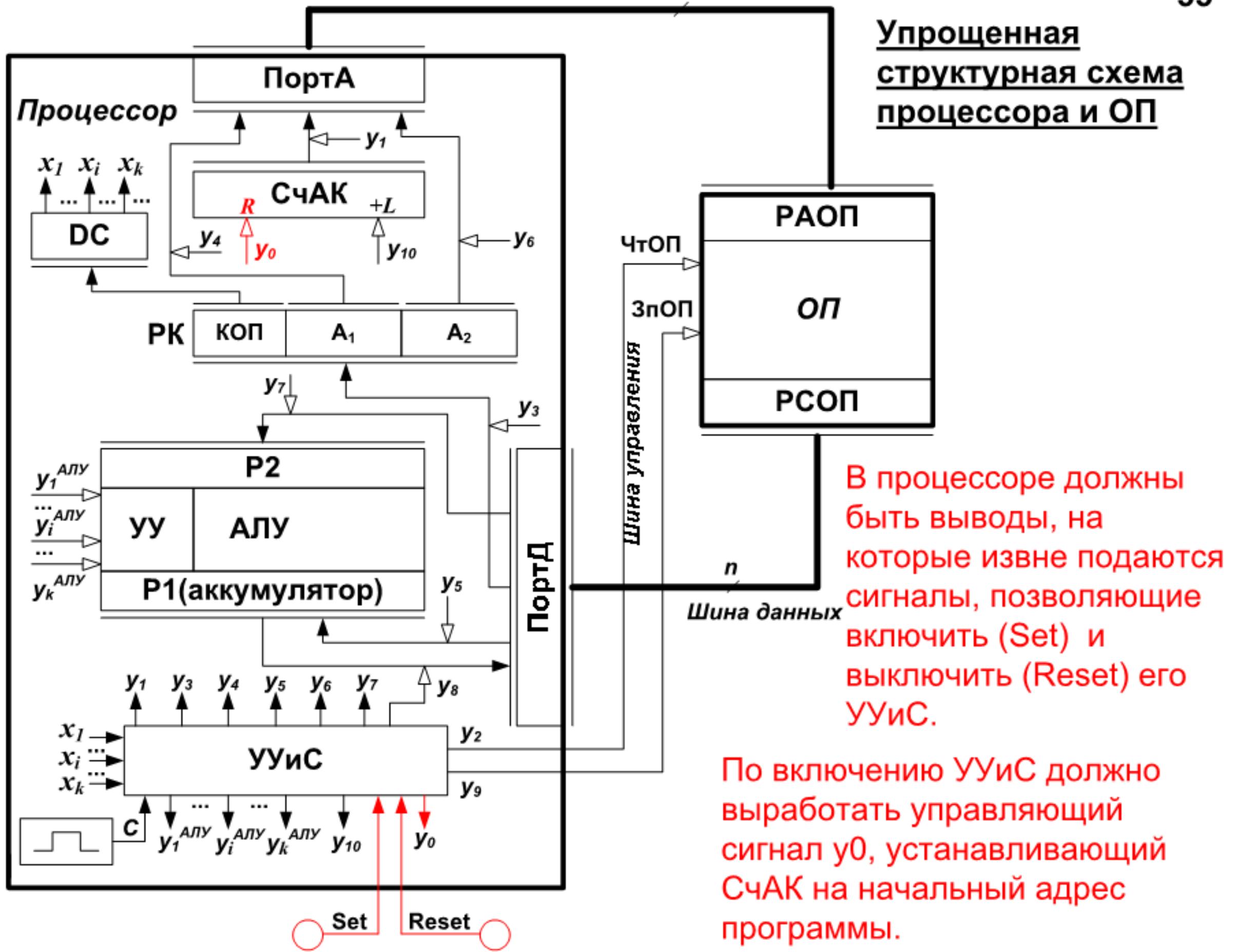
Рабочий цикл процессора

5 этап.

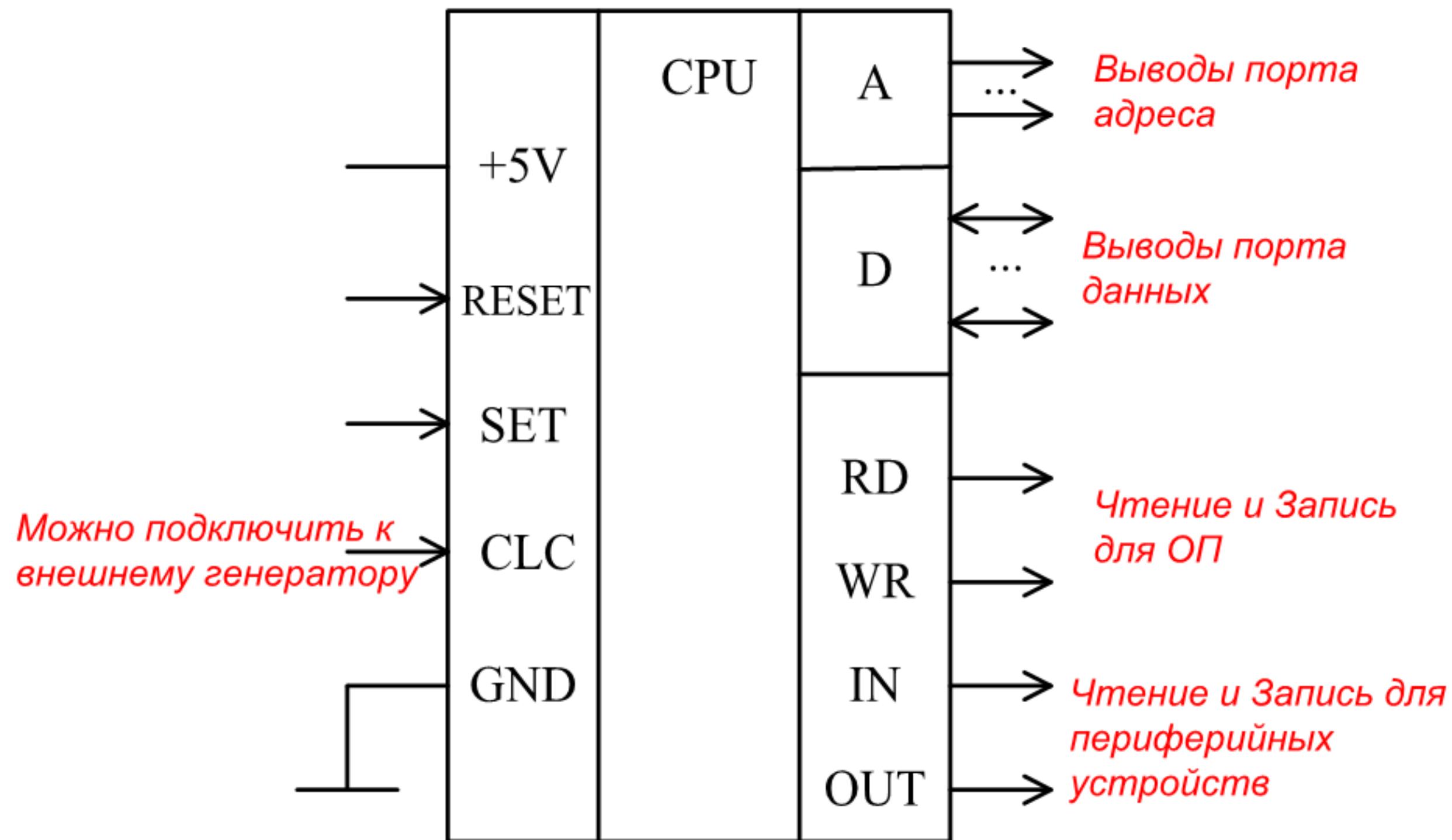
Продвижение СЧАК

**(этот этап можно было выполнить сразу после этапа
выборки команды)**





Обобщенное условно-графическое обозначение (УГО) процессора



УУиС после включения реализует микропрограмму 57 работы процессора, состоящую в повторении этапов его рабочего цикла:

Исполнение
предыдущих команд

$y_0, \dots, y_1, y_2, y_3, y_4, y_2, y_5, y_6,$

$y_2, y_7, y_i^{ALU}, y_8, y_4, y_9, y_{10}, \dots,$

y_1, y_2, y_3, \dots

Исполнение следующих
команд

Микропрограмма
обработки прерываний,
если они возникли в
процессе выполнения
текущей команды
(рассмотрим позже).

Пример.

Проиллюстрируем последовательность изменения содержимого регистров процессора при выполнении арифметической команды (сложение чисел с фиксированной точкой).

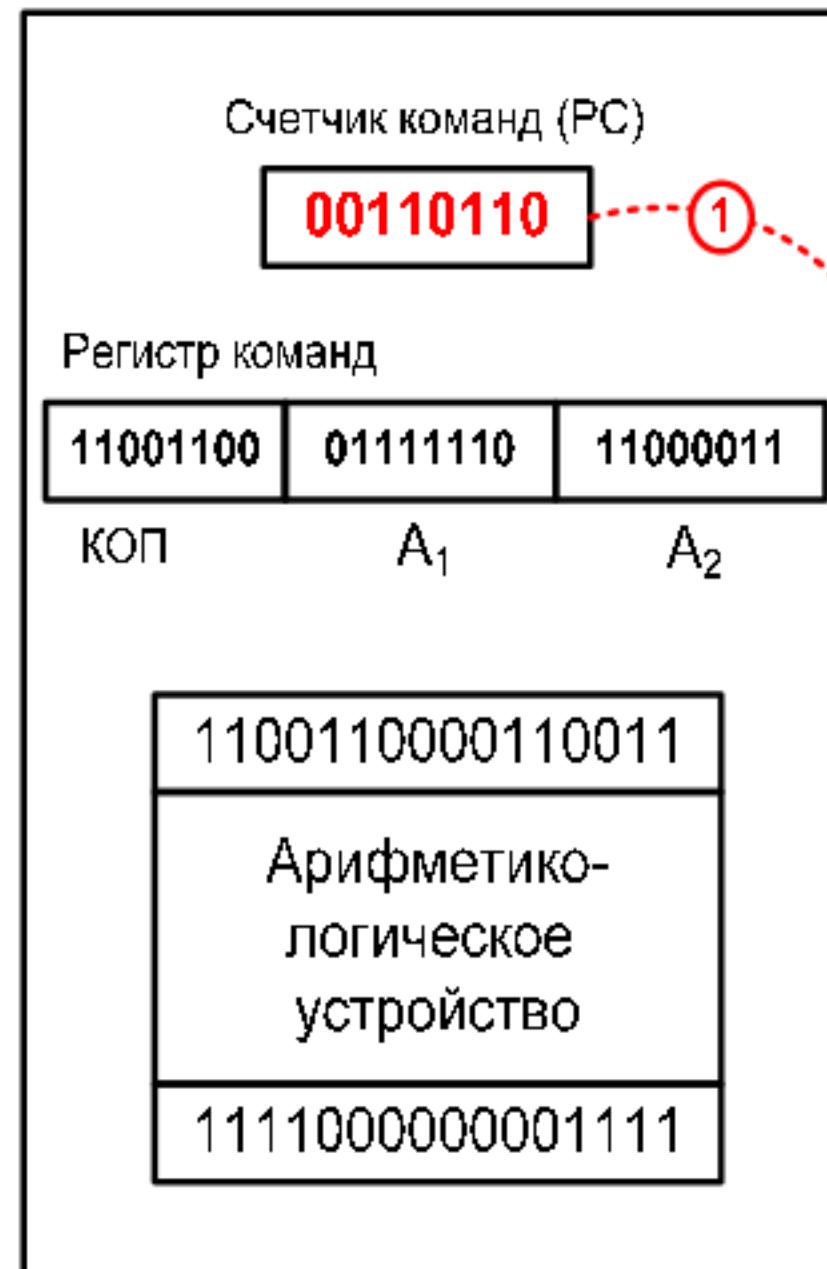
Дано:

- 1) Система команд процессора включает в себя 256 команд (следовательно поле КОП должно содержать 8 бит).
- 2) В арифметической операции участвуют два операнда, находящиеся в ОП (в команде должно быть два поля для представления адресов операндов). Если длина операнда превышает 1 байт, то адресом операнда считается адрес его первого байта. Поле КОП однозначно определяет выполняемую операцию, а также тип (длину) операндов.
- 3) ОП организована как последовательность байтов, каждый байт имеет адрес (номер).
- 4) Ширина шины данных 32 бита (значит, из ОП памяти за одну операцию чтения (записи) может быть считано (записано) минимум 1 байт, максимум 4 байта).
- 5) Ширина шины адреса 8 бит (значит объем оперативной памяти не может превышать $2^8=256$ байтов).
- 6) ОП хранит и данные и команды (прин斯顿ская архитектура ОП, предложенная Фон-Нейманом).

СЧАК часто обозначают СК (счетчик команд) или PC (program counter).

В рассматриваемый момент времени в СК – адрес текущей команды, в РК – предыдущая команда, в регистрах АЛУ operand и результат предыдущей команды.

Центральный процессор (CPU)

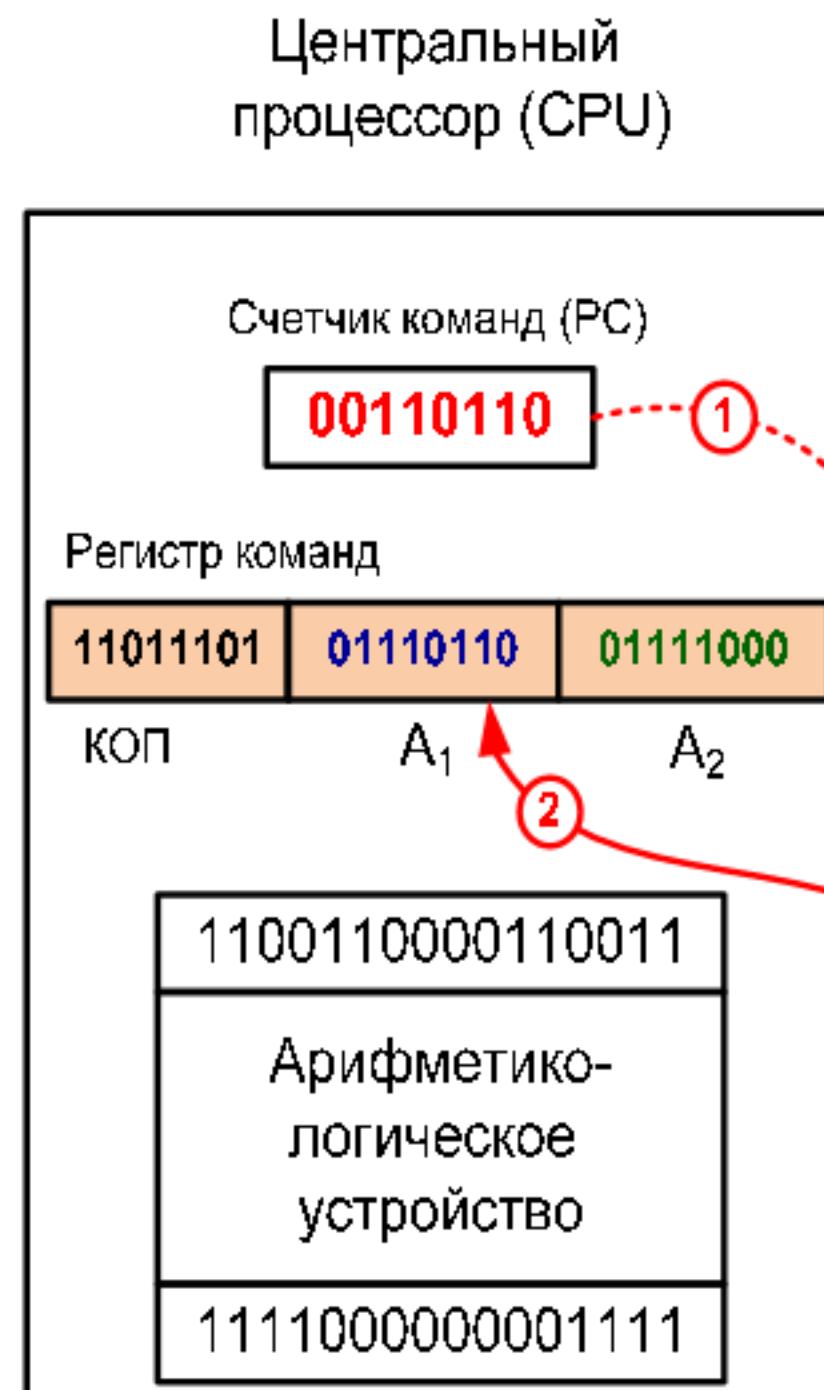


Оперативная память (Memory)

Адреса байтов
11111111 (255)
...
11101110 01111001 (121)
00110111 01111000 (120)
11010001 01110111 (119)
00010010 01110110 (118)
...
11111000 00111001 (57)
01111000 00111000 (56)
01110110 00110111 (55)
11011101 00110110 (54)
...
01011100 00000100 (4)
11110000 00000011 (3)
11000110 00000010 (2)
01001100 00000001 (1)
01001100 00000000 (0)

1) Процессор выставляет в порт адреса (на шину адреса) значение, находящееся в счетчике команд (СК).

Активные уровни управляющих сигналов чтения из ОП или записи в ОП выставляются процессором в нужные моменты времени на шине управления.



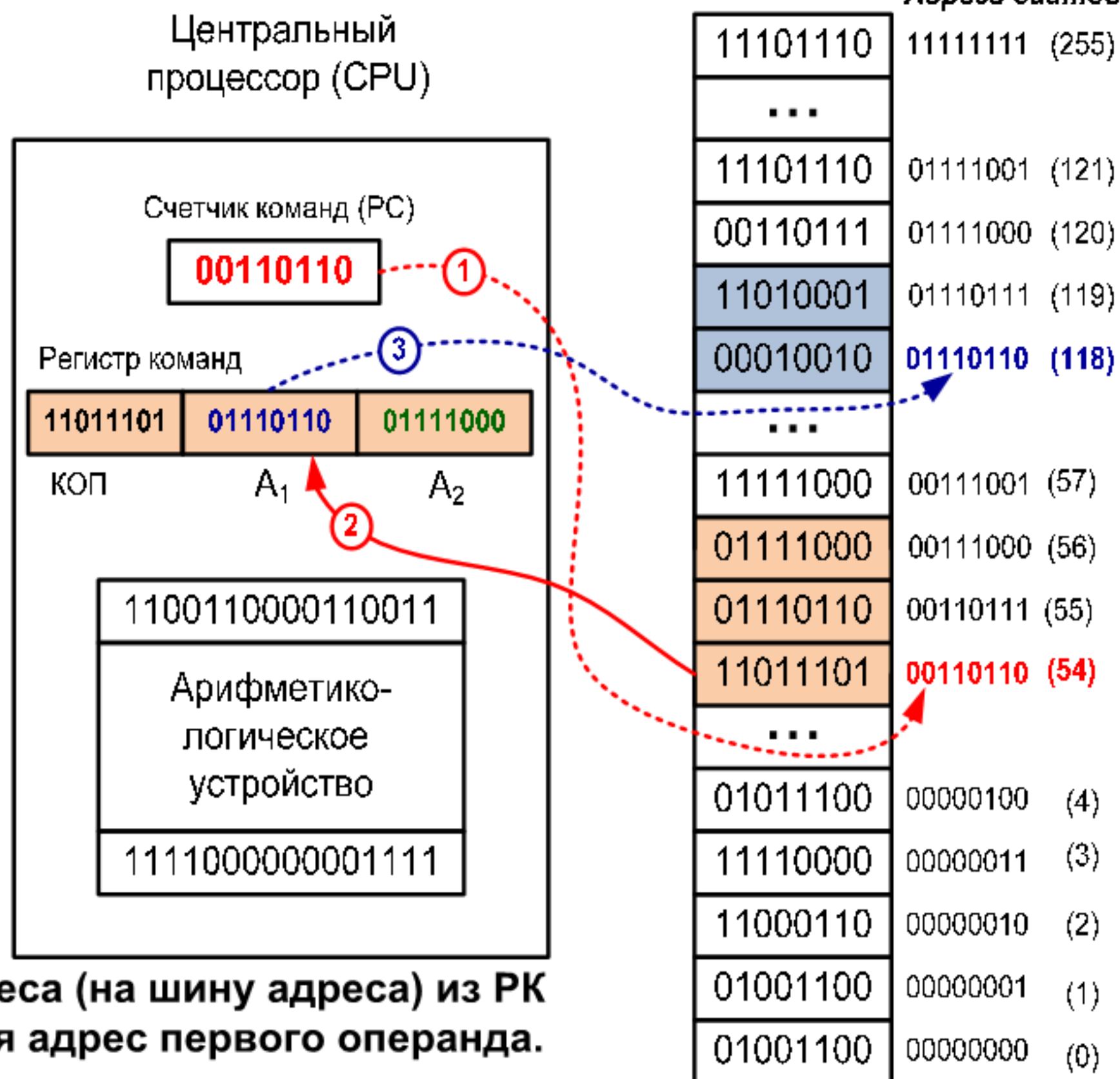
Оперативная память (Memory)

Адреса байтов

11101110	11111111 (255)
...	
11101110	01111001 (121)
00110111	01111000 (120)
11010001	01110111 (119)
00010010	01110110 (118)
...	
11111000	00111001 (57)
01111000	00111000 (56)
01110110	00110111 (55)
11011101	00110110 (54)
...	
01011100	00000100 (4)
11110000	00000011 (3)
11000110	00000010 (2)
01001100	00000001 (1)
01001100	00000000 (0)

2) На шину данных (в порт данных) из ОП считывается очередная команда (адрес области ОП – на шине адреса). Информация с шины данных попадает в регистр команд (РК) процессора.

Оперативная
память (Memory)



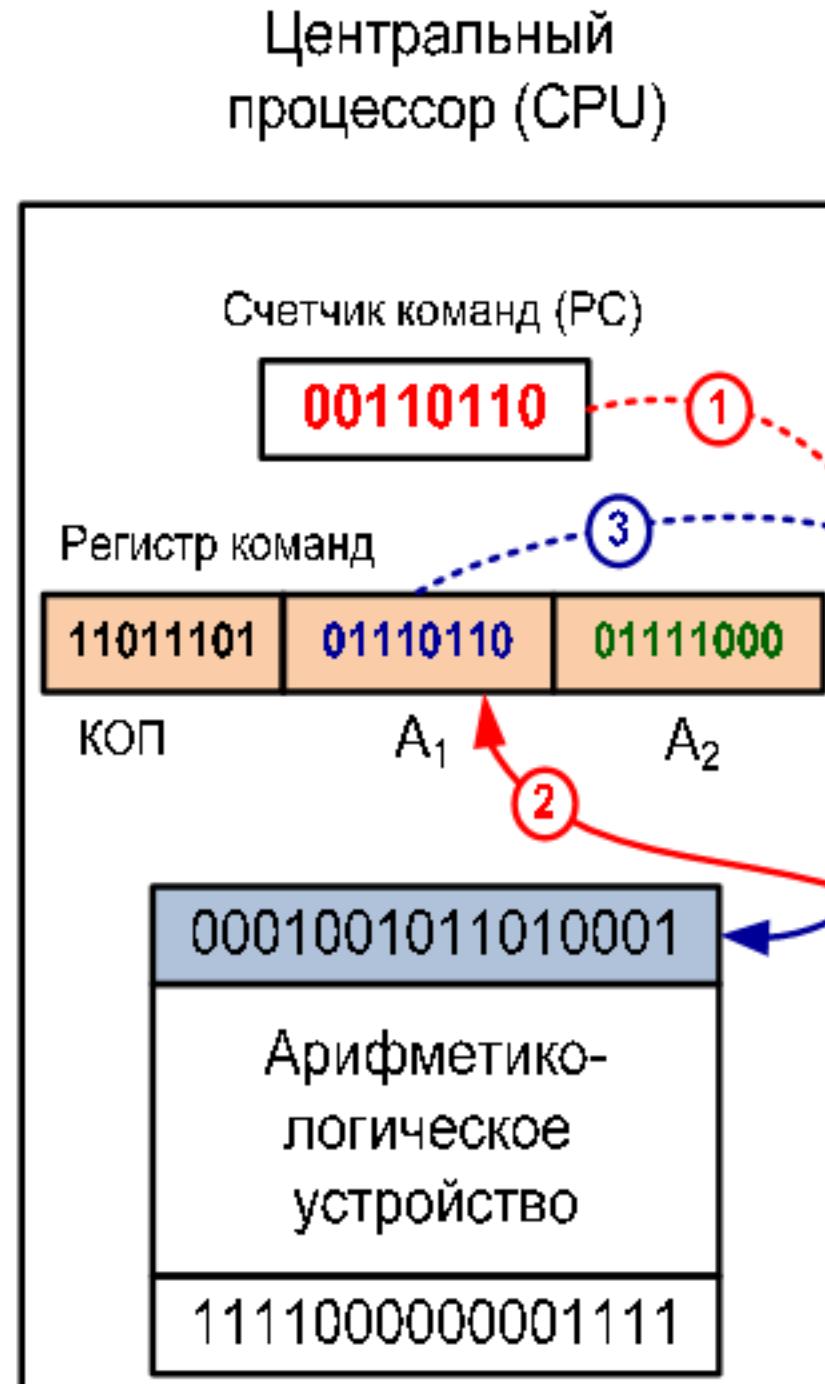
3) В порт адреса (на шину адреса) из РК выставляется адрес первого операнда.

Оперативная память (Memory)

Адреса байтов

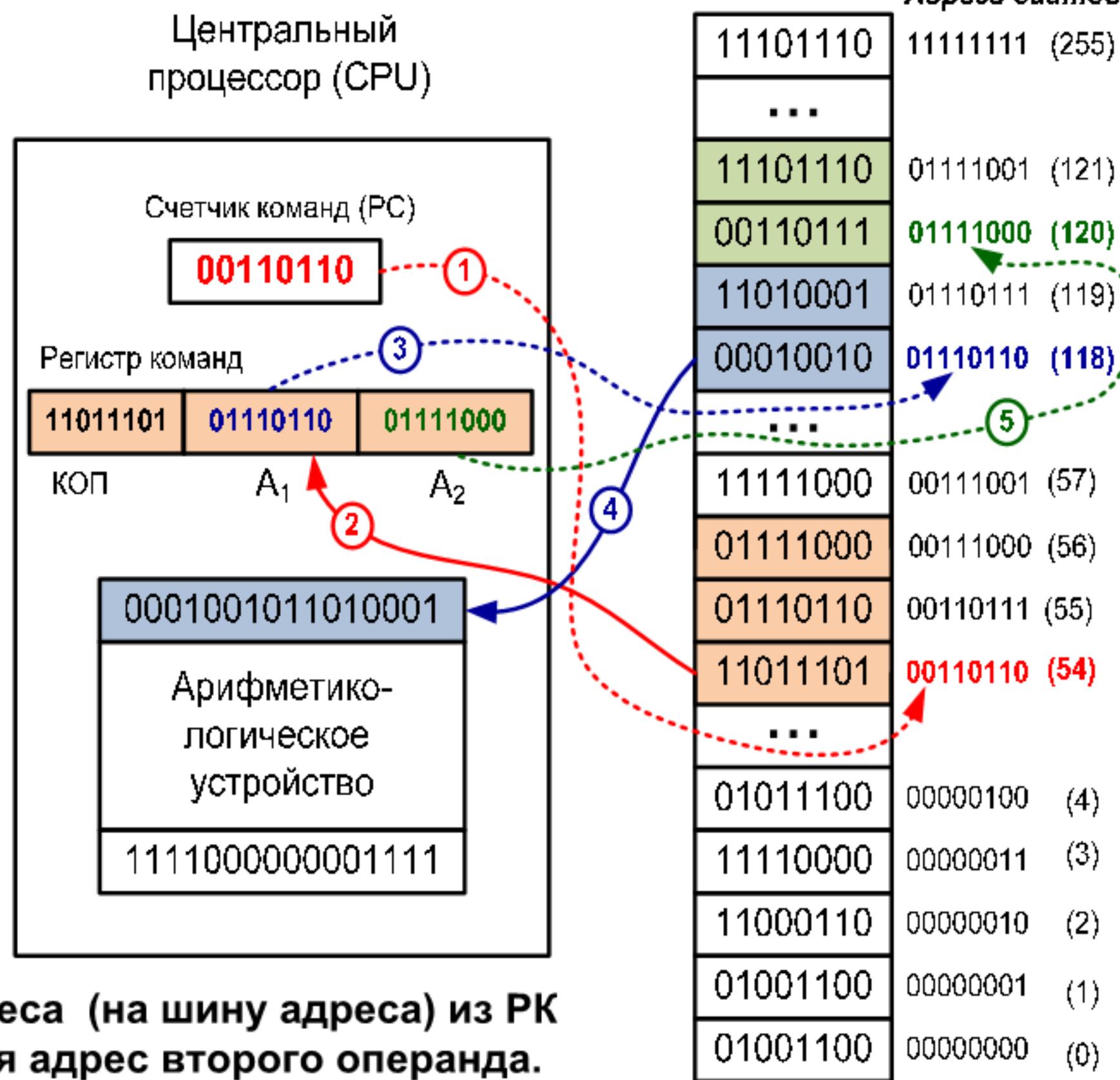
11101110	11111111 (255)
...	
11101110	01111001 (121)
00110111	01111000 (120)
11010001	01110111 (119)
00010010	01110110 (118)
...	
11111000	00110101 (57)
01111000	00111000 (56)
01110110	00110111 (55)
11011101	00110110 (54)
...	
01011100	00000100 (4)
11110000	00000011 (3)
11000110	00000010 (2)
01001100	00000001 (1)
01001100	00000000 (0)

Допустим, значение поля КОП подразумевает целочисленные операнды длиной 2 байта



- 4) На шину данных (в порт данных) из ОП считывается первый операнд (адрес области ОП – на шине адреса). Информация с шины данных попадает в первый регистр АЛУ (аккумулятор).

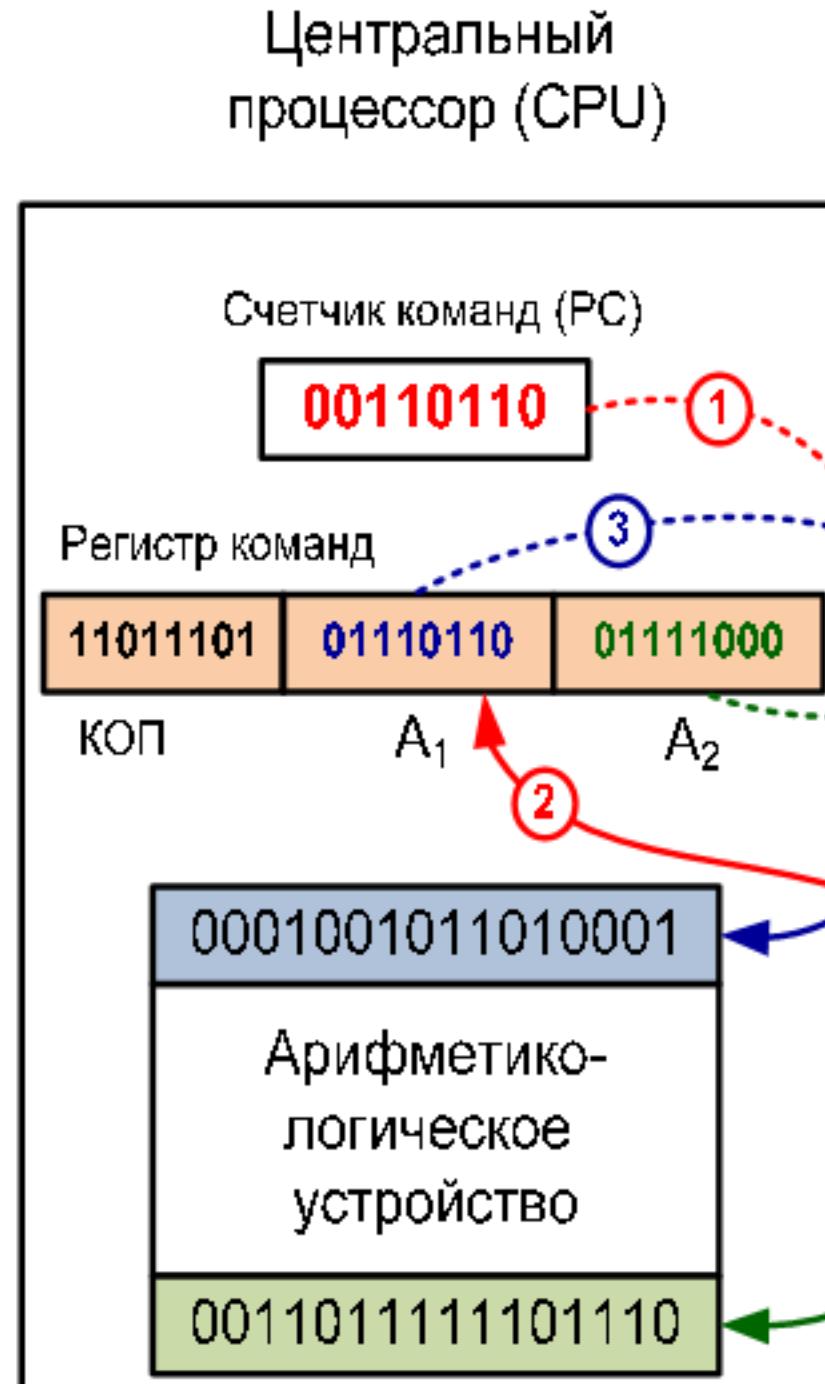
Оперативная
память (Memory)



Оперативная
память (Memory)

Адреса байтов

11101110	11111111 (255)
...	
11101110	01111001 (121)
00110111	01111000 (120)
11010001	01110111 (119)
00010010	01110110 (118)
...	
11111000	00111001 (57)
01111000	00111000 (56)
01110110	00110111 (55)
11011101	00110110 (54)
...	
01011100	00000100 (4)
11110000	00000011 (3)
11000110	00000010 (2)
01001100	00000001 (1)
01001100	00000000 (0)

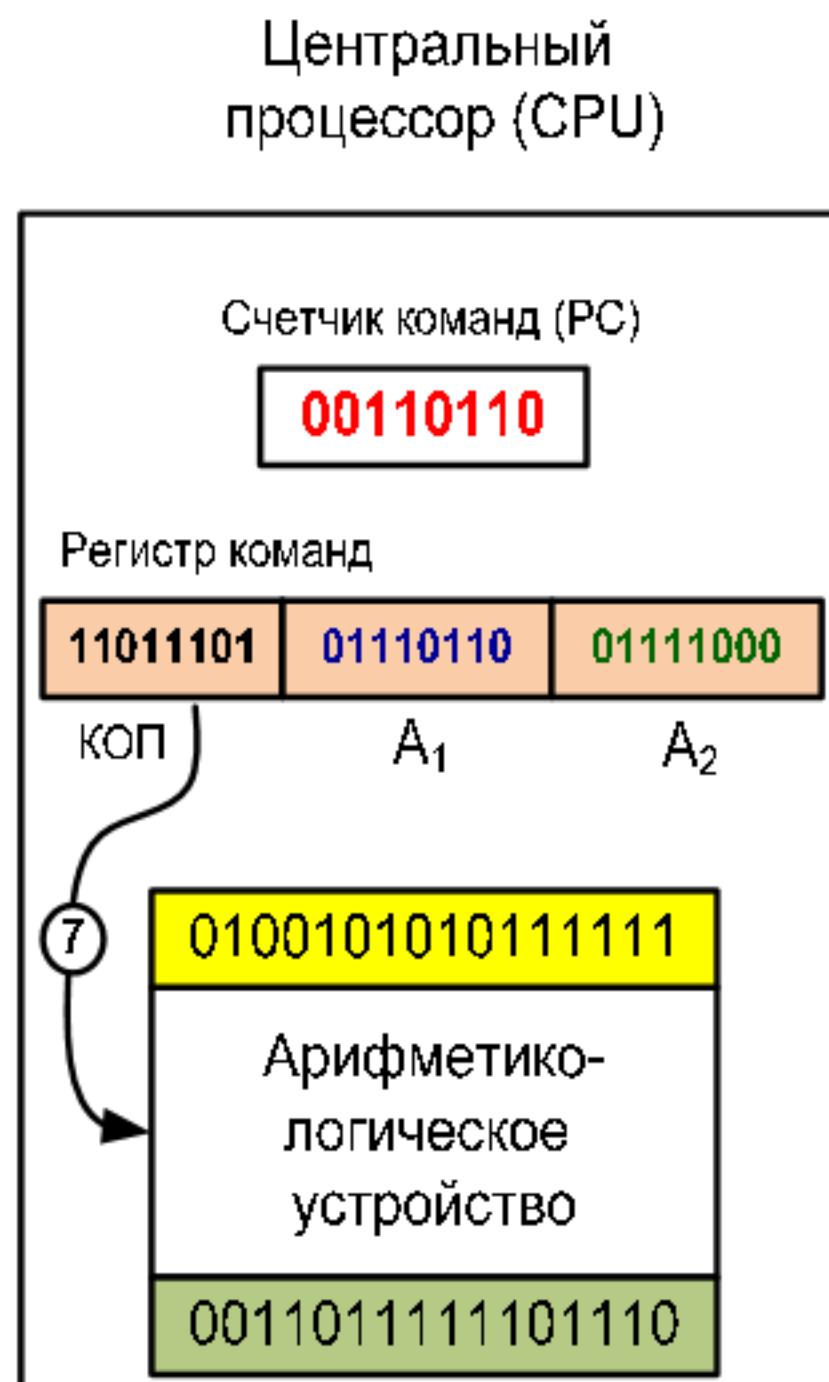


6) На шину данных (в порт данных) из ОП считывается второй операнд (адрес области ОП – на шине адреса). Информация с шины данных попадает во второй регистр АЛУ.

Оперативная
память (Memory)

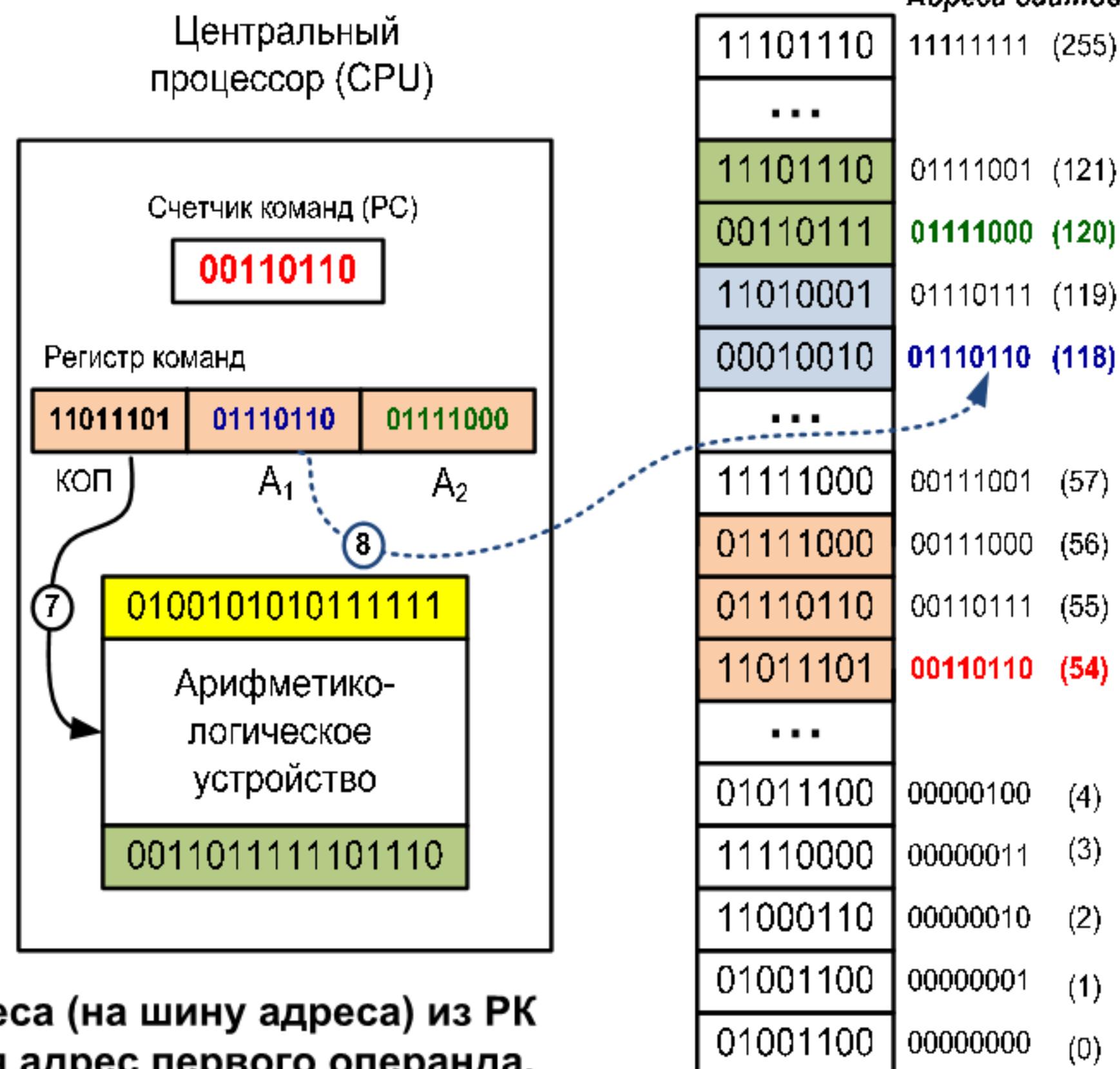
Адреса байтов

11101110	11111111 (255)
...	
11101110	01111001 (121)
00110111	01111000 (120)
11010001	01110111 (119)
00010010	01110110 (118)
...	
11111000	00111001 (57)
01111000	00111000 (56)
01110110	00110111 (55)
11011101	00110110 (54)
...	
01011100	00000100 (4)
11110000	00000011 (3)
11000110	00000010 (2)
01001100	00000001 (1)
01001100	00000000 (0)

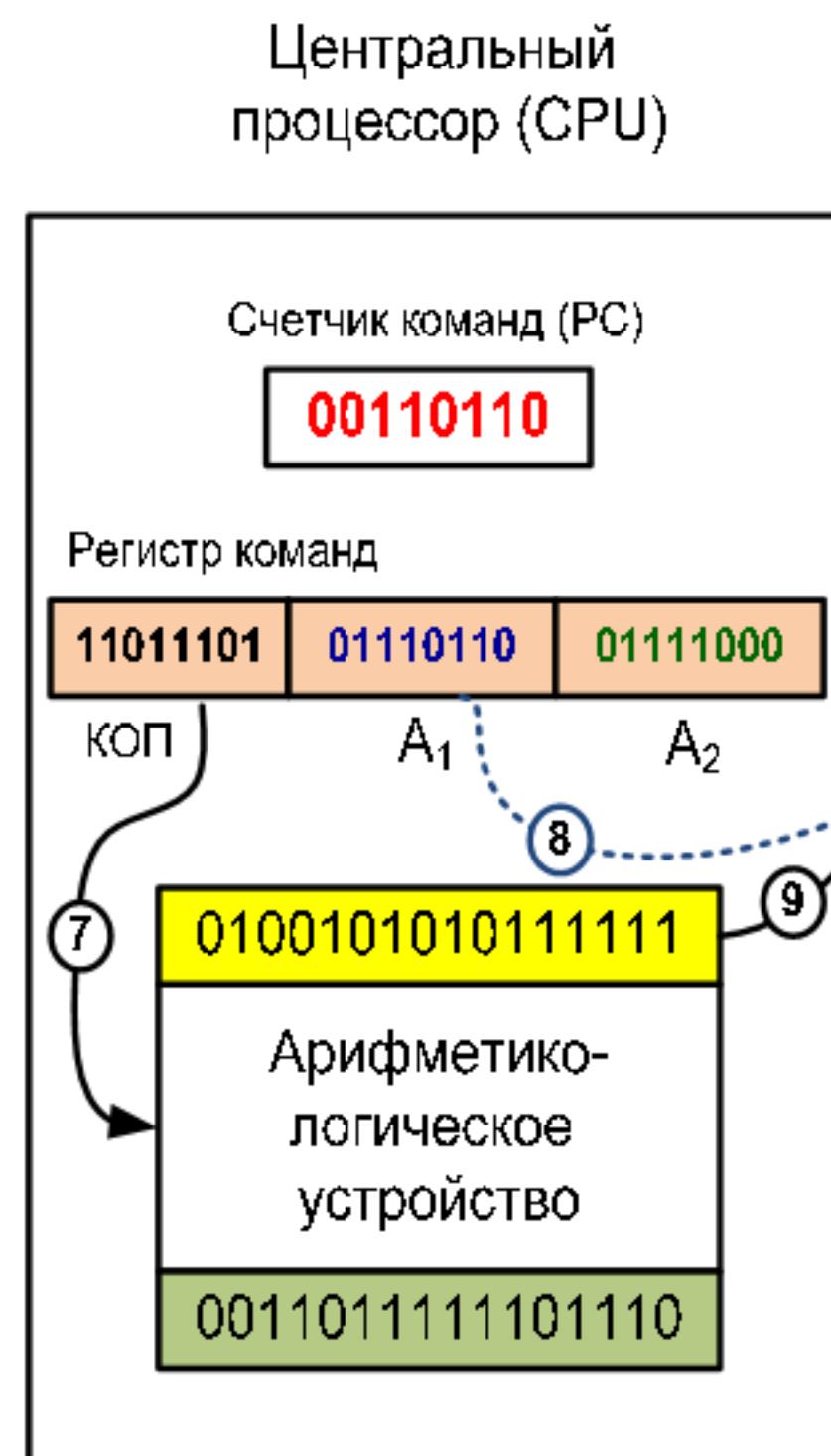


7) В АЛУ выполняется операция, соответствующая значению в поле КОП РК (здесь – сложение с фиксированной точкой). В аккумуляторе (выделен желтым цветом) появляется результат.

Оперативная
память (Memory)



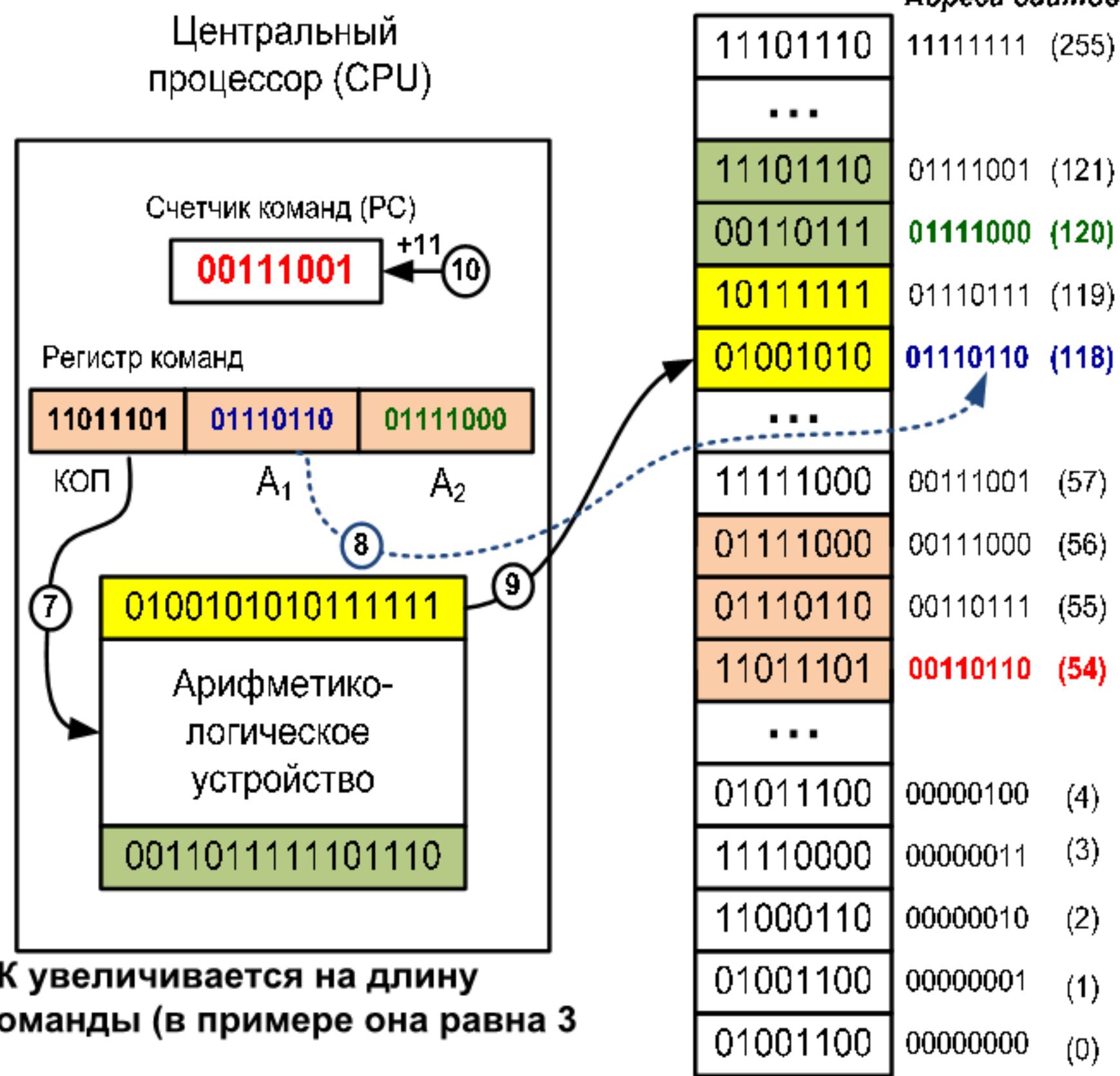
Оперативная
память (Memory)



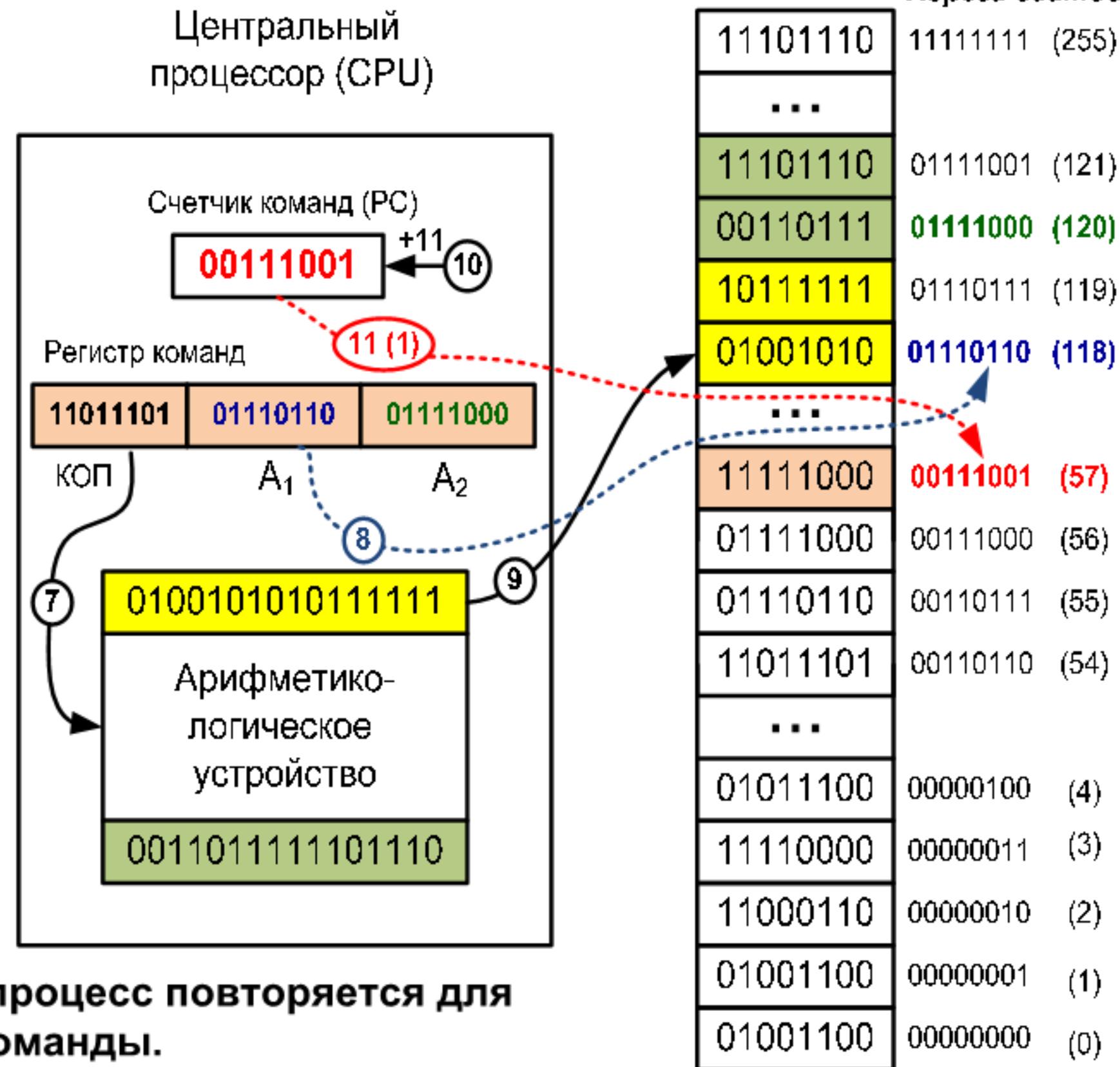
Адреса байтов	
11101110	11111111 (255)
...	
11101110	01111001 (121)
00110111	01111000 (120)
10111111	01110111 (119)
01001010	01110110 (118)
...	
11111000	00111001 (57)
01111000	00111000 (56)
01110110	00110111 (55)
11011101	00110110 (54)
...	
01011100	00000100 (4)
11110000	00000011 (3)
11000110	00000010 (2)
01001100	00000001 (1)
01001100	00000000 (0)

9) Результат из АЛУ по шине данных (через порт данных) записывается в ОП на место первого операнда (адрес области ОП – на шине адреса).

Оперативная
память (Memory)



Оперативная
память (Memory)



Оперативная
память (Memory)



Адреса байтов	
11101110	11111111 (255)
...	
11101110	01111001 (121)
00110111	01111000 (120)
10111111	01110111 (119)
01001010	01110110 (118)
...	
11111000	00111001 (57)
01111000	00111000 (56)
01110110	00110111 (55)
11011101	00110110 (54)
...	
01011100	00000100 (4)
11110000	00000011 (3)
11000110	00000010 (2)
01001100	00000001 (1)
01001100	00000000 (0)

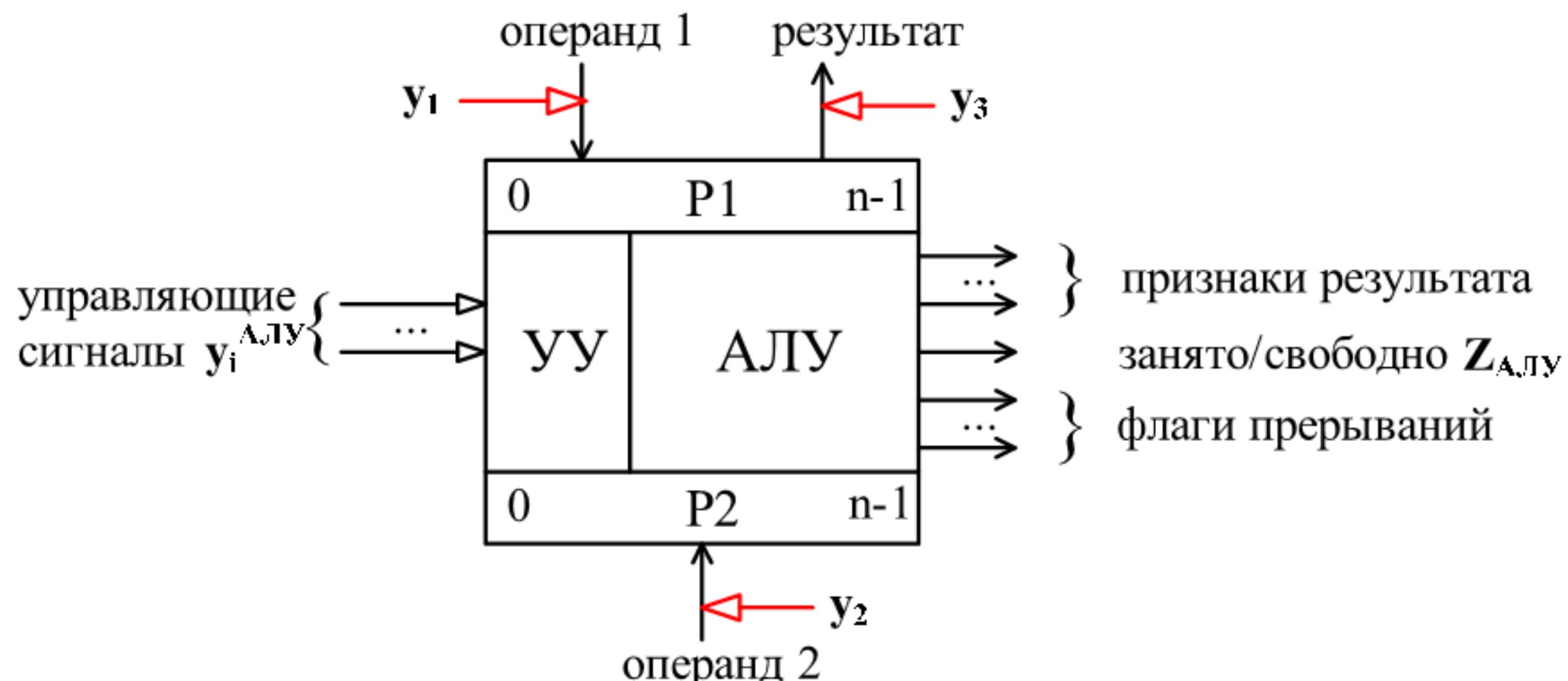
1) Процессор выставляет на шину адреса значение, находящееся в счетчике команд (СК)...

Вопросы, которые могут возникнуть:

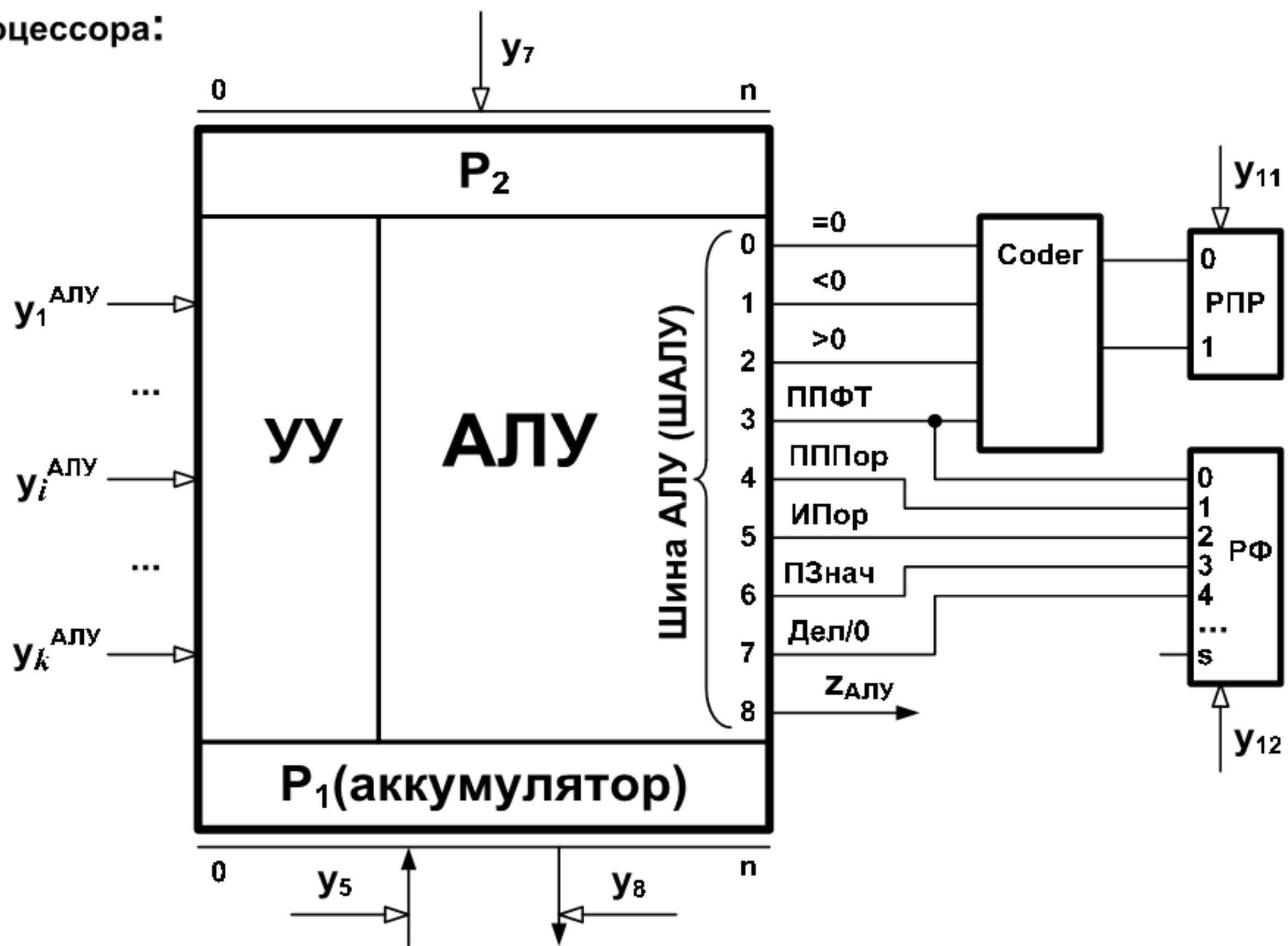
- 1) Как процессор узнает, что АЛУ закончило выполнение микропрограммы инициированной арифметической (или логической) операции?
- 2) Где и как сохраняется признак результата, который изменяют операции сложения, вычитания, инкрементирования, декрементирования и поразрядные логические операции ?

На рассмотренной выше структурной схеме процессора был изображен «облегченный» вариант автономного АЛУ.

В лекции, посвященной АЛУ, нами рассматривалась более полная схема:



процессора:



После того, как процессор загрузит в регистры АЛУ операнды (2 этап рабочего цикла) и инициирует выполнение нужной микропрограммы в АЛУ (3 этап рабочего цикла), АЛУ устанавливает на своем выходе Z_{ALU} (**ШАЛУ(8)** – сигнал «занято/свободно») единицу (занято!). Процессор ожидает, пока АЛУ не установит ноль на выходе. Это будет означать, что микропрограмма исполнения операции выполнилась, в Р1 находится результат, и теперь АЛУ готово для исполнения микропрограммы следующей операции (свободно!).

Помимо результата выполнения операции (в Р1) АЛУ формирует на выходнойшине (**ШАЛУ**) унитарный код признака результата (только для операций сложения, вычитания, инкрементирования, декрементирования, и поразрядных логических операций: И, ИЛИ, Исключающее ИЛИ):

ШАЛУ(0)=1, если $P1=0$;

ШАЛУ(1)=1, если $P1<0$;

ШАЛУ(2)=1, если $P1>0$;

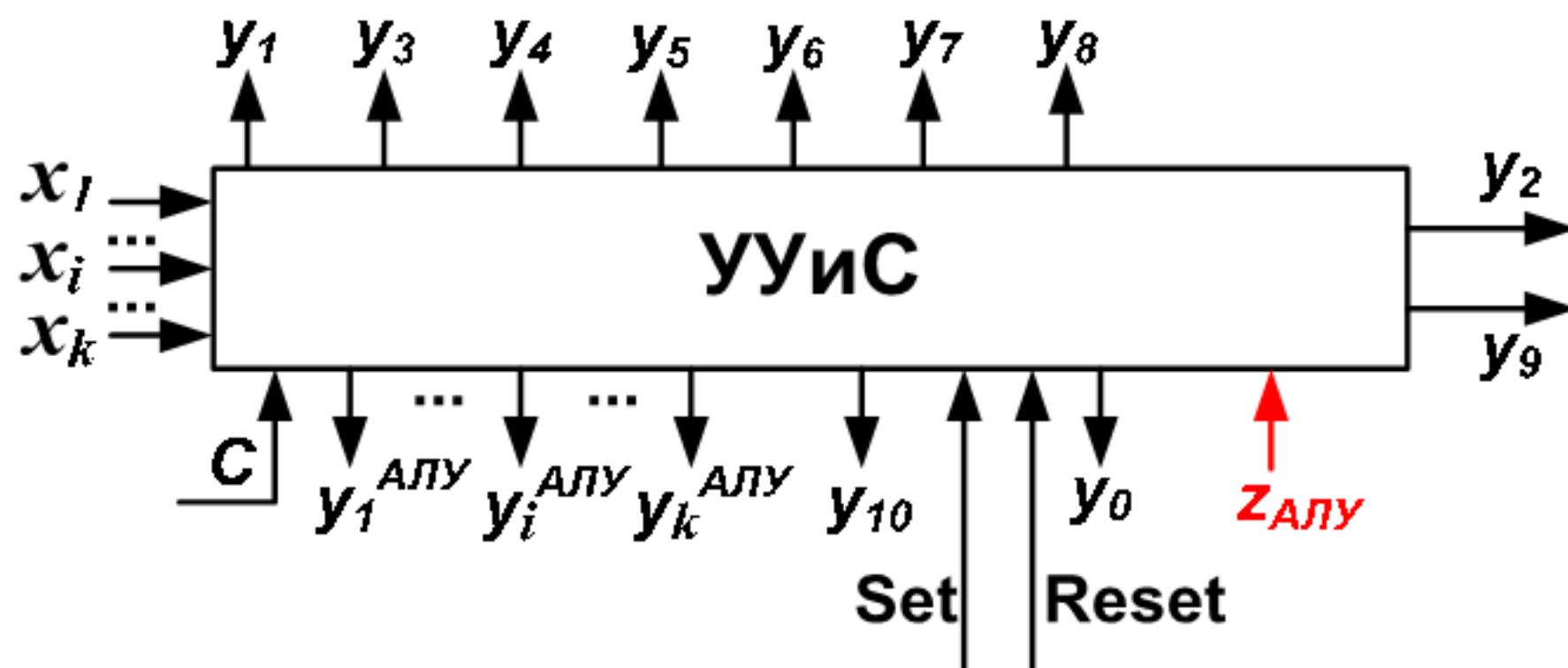
ШАЛУ(3)=1, если произошла исключительная ситуация «Переполнение с фиксированной точкой».

Четырехразрядный унитарный код признака результата преобразуется с помощью шифратора (**Coder**) в двухразрядный позиционный код признака результата: (00 – результат равен нулю, 01 – результат меньше нуля, 10 – результат больше нуля, 11 – переполнение с фиксированной точкой).

Помимо признака результата АЛУ имеет на ШАЛУ линии, на которых формируются флаги возникших исключительных ситуаций при выполнении операций с плавающей точкой и операций деления:

- ШАЛУ(4)=1 – переполнение порядка;
- ШАЛУ(5)=1 – исчезновение порядка;
- ШАЛУ(6)=1 – потеря значимости;
- ШАЛУ(8)=1 – деление на ноль.

После того как УУиС процессора получит от АЛУ осведомительный сигнал ШАЛУ(8)=0 ($Z_{\text{АЛУ}}=0$), процессор переходит к следующему (четвертому) этапу своего рабочего цикла – «Запись результата в ОП».



11.1) y_8 : ПортД:=Р1

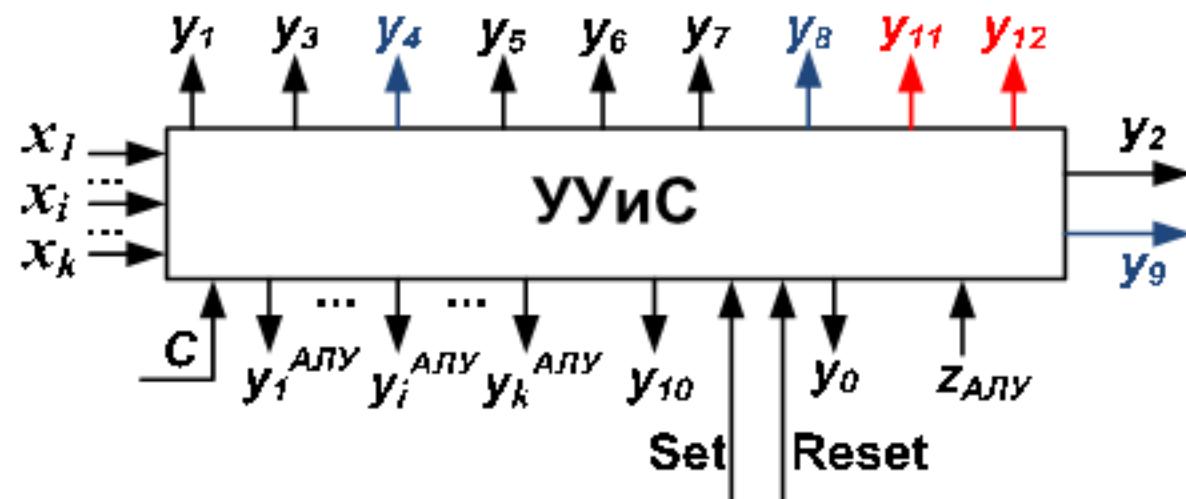
11.2) y_{11} : РПР(0:1):=Coder(ШАЛУ(0:3))

(только для add, sub, inc, dec, and, or, xor)

11.3) y_{12} : РФ(0:4):=ШАЛУ(3:7)

12) y_4 : ПортA:=A1

13) y_9 : ЗпОП



Помимо результата выполнения операции, на этом этапе (если выполнялась команда сложения, вычитания, инкрементирования, декрементирования, поразрядного И, ИЛИ, исключающего ИЛИ) сохраняется в регистре признака результата (РПР) позиционный код признака результата, сформированный шифратором (Coder). **Другие команды не должны изменять содержимое РПР!**

Флаги исключительных ситуаций, которые могут возникать при выполнении арифметических операций, запоминаются на регистре флагов прерываний (РФ), для их последующего анализа и обработки системой прерываний.

Если ШАЛУ(3:7)=00000, значит исключительной ситуации при выполнении операции в АЛУ не возникло, т.е. операция была выполнена успешно.

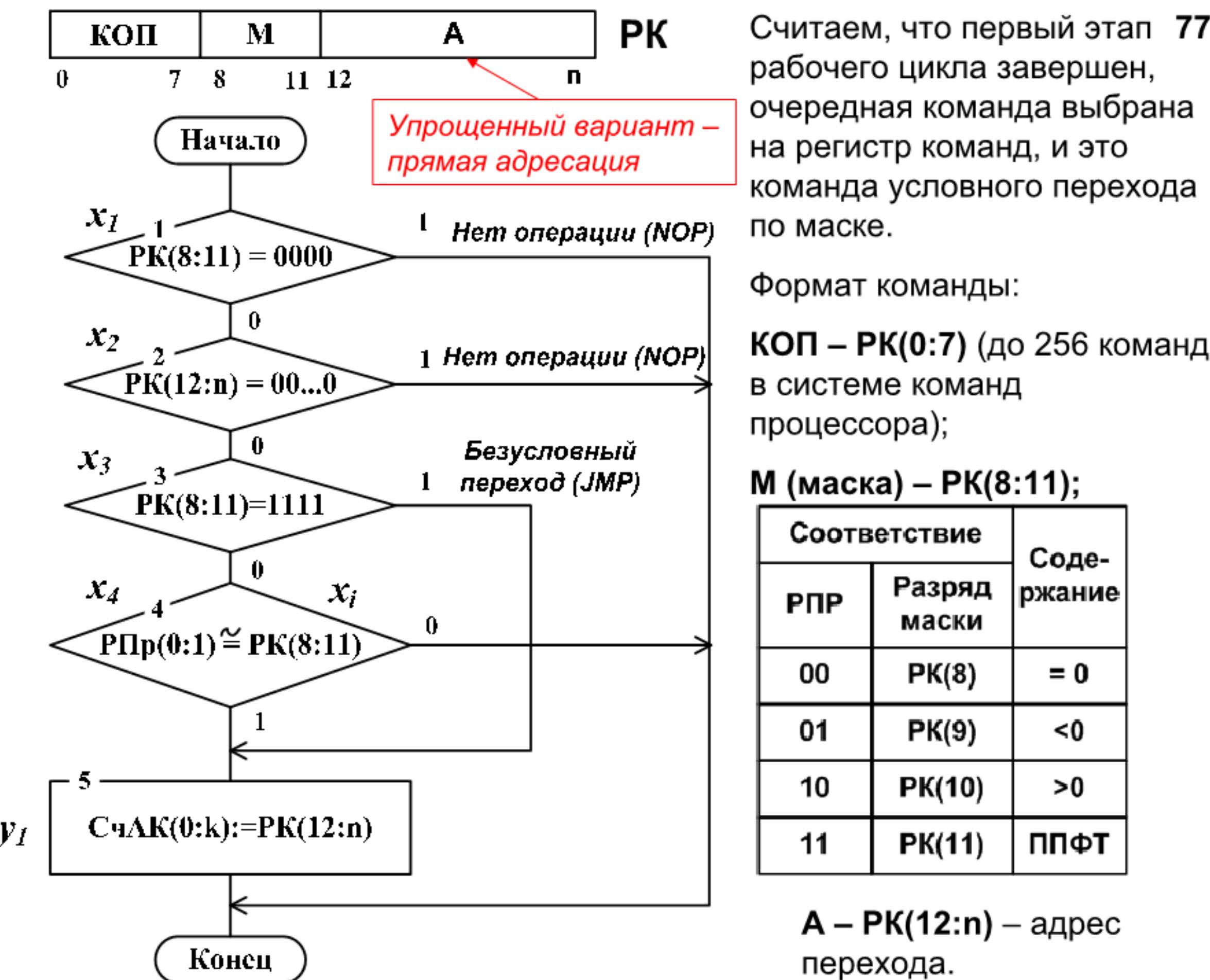
Внимание! В структуру процессора ввели еще два специальных регистра: двухразрядный РПР и $(s+1)$ -разрядный РФ. Разрядность РФ больше числа флагов исключительных ситуаций на ШАЛУ, т.к. возможны прерывания не только от АЛУ, но и от таймеров, внешних устройств и т.п. (рассмотрим позже).

После выполнения арифметической (логической) команды счетчик команд всегда продвигается на длину выполненной команды. Таким образом, осуществляется переход к следующей по порядку команде.

А как осуществить переход к произвольной команде, т.е. как организовать разветвляющийся или циклический вычислительный процесс?

Для этого служат команды перехода (передачи управления).

В качестве примера команды перехода рассмотрим команду условного перехода по маске.



Если в поле маски перехода M или в поле адреса перехода A нули, то переход не выполняется (рассматриваемая команда соответствует команде «нет операции»).

Если в поле M все единицы, переход выполняется обязательно (рассматриваемая команда соответствует команде безусловного перехода).

Если значение признака результата соответствует коду, указанному в поле M, продвинутый (СЧАК продвигается после выборки команды перехода перед ее выполнением) адрес команды в СЧАК замещается адресом перехода A. В противном случае продолжается выполнение обычной последовательности команд с использованием продвинутого адреса. Признак результата остается без изменения.

Прерывания программы отсутствуют.

Вопрос: Какова разрядность поля A (чему равно n) и какова разрядность СЧАК (чему равно k)?

Разрядность адреса в вычислительной системе (N_A) определяется в зависимости от объема ОП в байтах $E_{оп}$:

$$N_A = \log_2 E_{оп}$$

В частности, такую разрядность будут иметь СЧАК и поле А регистра команд (РК).

Фрагмент структурной схемы процессора, исполняющий команду условного перехода по маске.

79

Нумерация управляющих и осведомительных сигналов взята условно (не связана с нумерацией в рассмотренном ранее фрагменте, исполняющем арифметическую операцию).

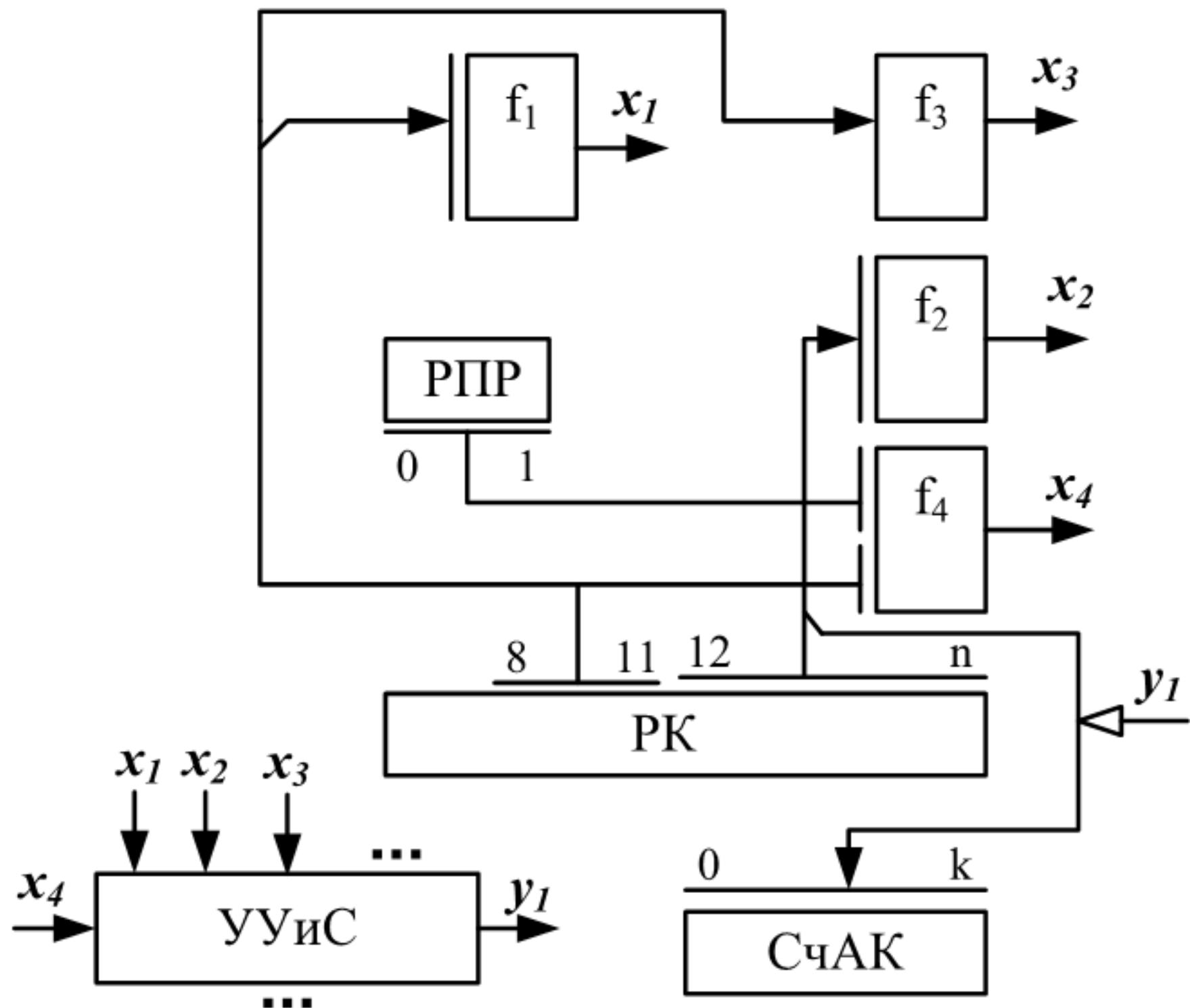
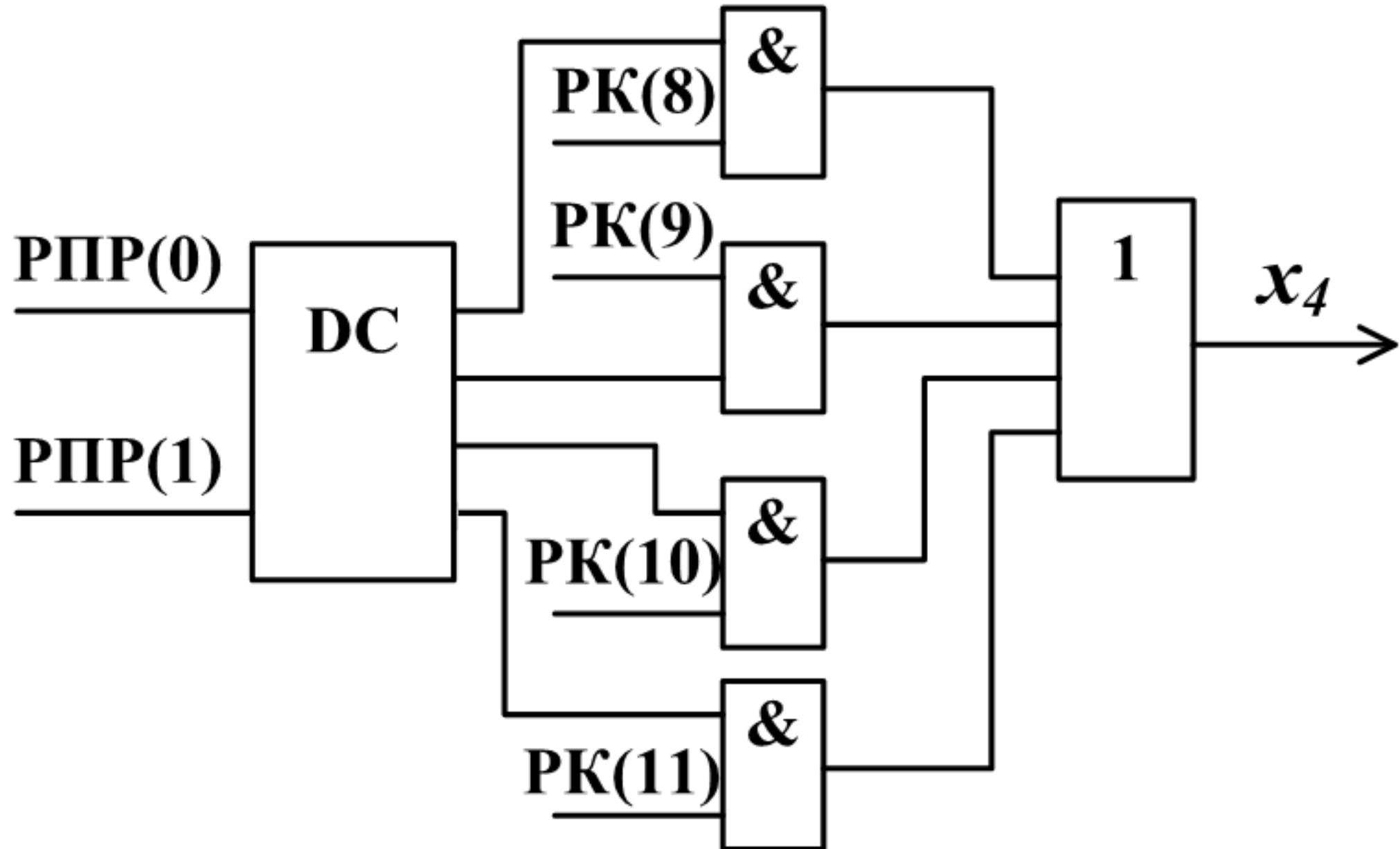


Схема вычисления осведомительного сигнала x_4 (булевая функция f_4)



Для x_1, x_2, x_3 – самостоятельно !