3.4. Система команд КР1816ВЕ48

3.4.1. Формат команд

Всі команди МК мають формат 1 чи 2 байта і виконуються за один чи два машинні цикли. Кожний цикл виконується за 5 тактів. Частота синхронізації тактів складає $F/3$, а циклів – $F/15$. Наприклад, при заданій частоті F=6 МГц тривалість тактів і циклів складає 0,5 і 2,5 мкс відповідно. За два машинні цикли виконуються всі команди з безпосереднім операндом, команди введення-виведення, команди передачі управління і роботи з підпрограмами, а також команди пересилок `MOVX`, `MOVP`, `MOVP3`. Решта команд виконуються за один машинний цикл. В МК передбачена можливість суміщення виконання однієї команди і вибірки наступної, що може зменшити час виконання команди. Мікроконтролер оперує з командами чотирьох типів (рис. 3.33).

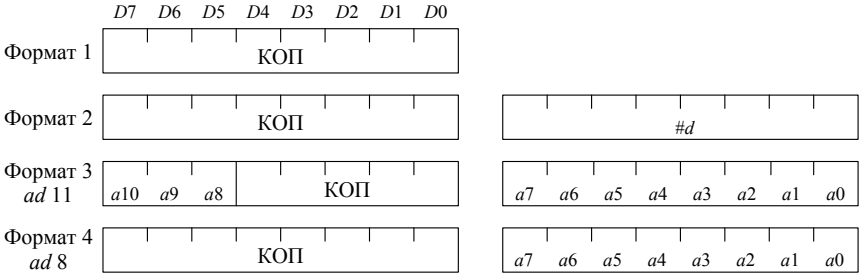

Рис. 3.33. Типи команд МК48

3.4.2. Система команд

У МК48 використовуються чотири способи адресації:

– пряма,
– безпосередня,
– непряма,
– неявна.

До переваг системи команд МК48 можна віднести: ефективний ввід/вивід, включаючи маскування і можливість управління окремими бітами портів; реалізацію розгалуження за значенням окремих бітів; обробку як двійкових, так і десяткових двійково-кодуючих чисел.

Під час виконання команд використовуються ознаки, що формуються в регістрі слова стану програми *RSW*, і ознаки користувача. Команди, в результаті виконання яких модифікуються ознаки, наведені в табл. 3.4. Функціональне призначення та способи формування ознак описані в розділі 3.2.2.

Таблиця 3.4. Формування ознак результату

Мнемоніка	**Ознаки**	**Мнемоніка**	**Ознаки**
ADD	*C AC*	JTF	*TF* = 0
CLR C	*C* = 0	MOV PSW,A	*C AC F*0 *BS*
CPL C	*C*	RETR	*C AC F*0 *BS*
CLR F0	*F*0 = 0	RLC A	*C*
CLR F1	*F*1 = 0	RRC A	*C*
CPL F0	*F*0	SEL MB0; SEL MB1	*DBF*
CPL F1	*F*1	SEL RB0; SEL RB1	*BS*
DA A	*C AC*		

Для опису команд використовуються мнемокоди мови асемблера МК48. Під час запису символічного коду команд застосовуються наступні позначення:

A — акумулятор;
T
r — номер регістру,
Rr — регістр з номером *r*;
b — номер біту,
p — номер порту вводу/виводу,
Pp — порт з номером *p*
a — адреса,
d — безпосередній операнд.
#d — безпосередній операнд (восьмирозрядне двійкове число);
@Rr — операнд, що адресується непрямо через *Rr*.
@A
#Rr

В стовбці «Коментарі» наведений опис операцій, що виконуються під час виконання команди. Для запису операцій використовуються мова мікрооперацій із застосуванням, символічних імен і скорочень. Номери розрядів регістрів подаються у квадратних дужках [ ]. Операнд за непрямої адресації подається у дужках ( ). Наприклад, запис $A := (Rr)$ означає, що в акумуляторі *A*

фіксується число, що зчитане з внутрішньої пам'яті даних за адресою, записаною в регістрі *Rr*. Для запису операндів часто використовуються складені слова, що записані у вигляді двох слів, розділених крапкою. Наприклад, запис *PC*[11..8].*A* подає дванадцятирозрядне двійкове слово, вісім молодших розрядів якого є вмістом акумулятора *A*, а чотири старші розряди – вмістом бітів (11..8) лічильника команд *PC*.

Всю множину команд асемблеру МК48 можна розбити на чотири основні групи:

- команди пересилки даних,
- команди основної групи:
  - виконання арифметичних операцій;
  - виконання логічних операцій;
- команди передачі управління;
- команди управління режимами роботи.

Система команд наведена в табл. 3.5. Команди згруповані за функціональними ознаками.

Команди пересилки даних

На рис. 3.34 представлений граф, що ілюструє можливі операції пересилки даних в МК48 (табл. 3.5). Операнди розрізняються за місцем розташування і за способом адресації. В операціях пересилки приймають участь такі операнди: акумулятор, РЗП, RSW, таймер, порти вводе/виводу, безпосередній операнд, ЗПД, РПД, ПП.

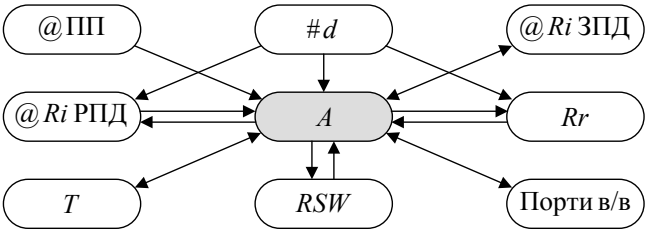

Рис. 3.34. Граф обміну даними у МК48

Всі операції пересилки даних в МК48 виконуються з застосуванням акумулятора *A*.

Під час пересилки даних між *A* та регістрами загального призначення РЗП банків регістрів, використовується пряма адресація, коли адреса операнда вміщується в тілі команди. Номер регістра, вміст якого пересилається в акумулятор вказується в трьох молодших бітах коду операції (рис. 3.33).

Таблиця 3.5. Система команд мікроконтролера КМ1816ВЕ48

Мнемоніка	Код команди	Коментарі
1	2	3
Основна група команд та команди пересилки даних		
Команди звернення до акумулятора		
CLR A	00100111	Встановлення вмісту акумулятора в нуль $A := 0$
CPL A	00110011	Інвертування вмісту A; $A := NOT\ A$
INC A	00010111	Інкремент вмісту A; $A := A + 1$
DEC A	00000111	Декремент вмісту A; $A := A - 1$
RR A	01110111	Циклічний зсув вмісту A вправо; $A[7] := A[0]$; $A[i] := A[i+1]$; $i = \overline{6, 0}$
RL A	11100111	Циклічний зсув вмісту A вліво; $A[0] := A[7]$; $A[i] := A[i+1] := A[i]$; $i = \overline{6, 0}$
RRC A	01100111	Циклічний зсув вмісту A з бітом переносу вправо; $A[7] := C$; $C := A[0]$; $A[i] := A[i+1]$; $i = \overline{6, 0}$
RLC A	11110111	Циклічний зсув вмісту A з бітом переносу вліво; $A[0] := C$; $C := A[1]$; $A[i+1] := A[i]$; $i = \overline{6, 0}$
SWAP A	01000111	Обмін тетрадами A; $A[7..0] \leftrightarrow A[3..0]$
DA A	01010111	Десяткова корекція вмісту A

Продовження табл. 3.5

1	2	3
`MOV A, Rr` ; $r=(7-0)$	`11111rrr`	Пересилка вмісту регістру в A; $A := Rr$
`MOV Rr, A` ; $r=(7-0)$	`10101rrr`	Пересилка вмісту A в регістр; $Rr := A$
`XCH A, Rr` ; $r=(7-0)$	`00101rrr`	Обмін вмісту A і регістру; $A \leftrightarrow Rr$
`ANL A, Rr` ; $r=(7-0)$	`01011rrr`	Логічне І вмісту A і регістру; $A := A\ AND\ Rr$
`ORL A, Rr` ; $r=(7-0)$	`01001rrr`	Логічне АБО вмісту A і регістру; $A := A\ OR\ Rr$
`XRL A, Rr` ; $r=(7-0)$	`11011rrr`	Виключне АБО вмісту A і регістру; $A := A\ XOR\ Rr$
`ADD A, Rr` ; $r=(7-0)$	`01101rrr`	Сума вмісту A і регістру; $A := A + Rr$
`ADDC A, Rr` ; $r=(7-0)$	`01111rrr`	Сума вмісту A, регістру і переносу C; $A := A + Rr + C$
`DEC Rr` ; $r=(7-0)$	`11001rrr`	Декремент вмісту регістру; $Rr := Rr - 1$
`INC Rr` ; $r=(7-0)$	`00011rrr`	Інкремент вмісту регістру; $Rr := Rr + 1$
Команди звернення до внутрішньої пам'яті даних		
`MOV A,@Rr` ; $r=0,1$	`1111000r`	Пересилка із внутрішньої пам'яті даних в A; $A := (Rr)$
`MOV @Rr,A` ; $r=0,1$	`1010000r`	Пересилка вмісту A до внутрішньої пам'яті даних; $(Rr) := A$
`XCH A,@Rr` ; $r=0,1$	`0010000r`	Обмін вмістом A і комірки внутрішньої пам'яті даних; $A \leftrightarrow Rr$
`XCHD A,@Rr` ; $r=0,1$	`0011000r`	Обмін молодшими тетрадами A і комірки внутрішньої пам'яті даних; $A[3..0] \leftrightarrow (Rr[3..0])$
`ANL A,@Rr` ; $r=0,1$	`0101000r`	Логічне І вмісту A і комірки внутрішньої пам'яті даних; $A := A\ AND\ (Rr)$

Продовження табл. 3.5

1	2	3
`ORL A,@Rr` ; $r = 0,1$	`0100000r`	Логічне АБО вмісту A і комірки резидентної пам'яті даних; $A := A\ OR\ (Rr)$
`XRL A,@Rr` ; $r = 0,1$	`1101000r`	Виключення АБО вмісту A і комірки резидентної пам'яті даних; $A := A\ XOR\ (Rr)$
`ADD A,@Rr` ; $r = 0,1$	`0110000r`	Сума вмісту A і комірки резидентної пам'яті даних; $A := A + (Rr)$
`ADDC A,@Rr` ; $r = 0,1$	`0111000r`	Сума вмісту A, комірки резидентної пам'яті даних і переносу C; $A := A + (Rr) + C$
`INC @Rr` ; $r = 0,1$	`0001000r`	Інкремент комірки резидентної пам'яті даних; $(Rr) := (Rr) + 1$
Команди роботи з зовнішньою пам'яттю даних		
`MOVX A,@Rr` ; $r = 0,1$	`1000000r`	Пересилка із ЗПД в A; $A := (Rr)$
`MOVX @Rr,A` ; $r = 0,1$	`1001000r`	Пересилка вмісту A до ЗПД; $(Rr) := A$
Команди звернення до пам'яті програми		
`MOV Rr,#d` ; $r = (7-0)$	`10111rrr` `dddddddd`	Пересилка безпосереднього операнда до регістру; $(Rr) := d$
`MOV A,#d`	`00100011` `dddddddd`	Пересилка безпосередньої адреси до A $A := d$
`MOV @Rr,#d` ; $r = 0,1$	`1011000r` `dddddddd`	Пересилка безпосереднього операнда до внутрішньої пам'яті даних $(Rr) := d$

Продовження табл. 3.5

1	2	3
ANL A,#d	01010011 dddddddd	Логічне І вмісту A з безпосереднім операндом; $A := A\ AND\ d$
ORL A,#d	01000011 dddddddd	Логічне АБО вмісту A з безпосереднім операндом; $A := A\ OR\ d$
XRL A,#d	11010011 dddddddd	Виключне АБО вмісту A з безпосереднім операндом; $A := A\ XOR\ d$
ADD A,#d	00000011 dddddddd	Сума вмісту A та безпосереднього операнду; $A := A + d$
ADDC A,#d	00010011 dddddddd	Сума вмісту A, безпосереднього операнду та переносу C; $A := A + d + C$
MOVP A,@A	10100011	Пересилка даних із поточної сторінки пам'яті програм до A; $A := (PC[11..8].A)$
MOVP3 A,@A	11100011	Пересилка даних із сторінки 3 пам'яті програм до A; $A := (0011.A)$
Команди звернення до регістру PSW		
MOV PSW,A	11010111	Пересилка вмісту A до регістру PSW; $PSW := A$
MOV A,PSW	11000111	Пересилка вмісту регістру PSW до A; $A := PSW$
MOV A,T	01000010	Пересилка вмісту $TCNT$ в A; $A := TCNT$
MOV T,A	01100010	Пересилка вмісту A в $TCNT$; $TCNT := A$
Команди встановлення ознак		
CLR C	10010111	Встановлення в нуль ознаки C; $C := 0$

Продовження табл. 3.5

1	2	3
`CPL C`	`10100111`	Інвертування ознаки C; $C := NOT\ C$
`CLR F0`	`10000101`	Встановлення в нуль ознаки $F0$; $F0 := 0$
`CLR F1`	`10100101`	Встановлення в нуль ознаки $F1$; $F1 := 0$
`CPL F0`	`10010101`	Інвертування ознаки $F0$; $F0 := NOT\ F0$
`CPL F1`	`10110101`	Інвертування ознаки $F1$; $F1 := NOT\ F1$
Команди звернення до портів P1 і P2		
`ANL Pp,#d` ; $p = 1, 2$	`100110pp` `dddddddd`	Логічне І порту $P1(P2)$ з безпосереднім операндом; $Pp := Pp\ AND\ d$
`ORL Pp,#d` ; $p = 1, 2$	`100010pp` `dddddddd`	Логічне АБО порту $P1(P2)$ з безпосереднім операндом; $Pp := Pp\ OR\ d$
`IN A,Pp`	`000010pp`	Введення даних із порту $P1(P2)$ в A; $A := Pp$
`OUTL Pp,A`	`001110pp`	Виведення вмісту A в порт $P1(P2)$ $Pp := A$
Команди звернення до портів P4, P5, P6, P7		
`ANLD Pp,A` ; $p = (7-4)$	`100111pp`	Логічне І порту $P4(P5, P6, P7)$ з A; $Pp := Pp\ AND\ A[3..0]$
`ORLD Pp,A` ; $p = (7-4)$	`100011pp`	Логічне АБО порту $P4(P5, P6, P7)$ з A; $Pp := Pp\ OR\ A[3..0]$
`MOVD A,Pp` ; $p = (7-4)$	`000011pp`	Ввід із порту $P4(P5, P6, P7)$ в A; $A[7..4] := 0$; $A[3..0] := Pp$
`MOVD Pp,A` ; $p = (7-4)$	`001111pp`	Вивід молодшої тетради із A в порт $P4(P5, P6, P7)$; $Pp := A[3..0]$

Продовження табл. 3.5

1	2	3
	Команди звернення до порту BUS	
ANL BUS,#d	10011000 dddddddd	Логічне І порту *BUS* з безпосереднім операндом; $BUS := BUS\ AND\ d$
ORL BUS,#d	10001000 dddddddd	Логічне АБО порту *BUS* з безпосереднім операндом; $BUS := BUS\ \ OR\ d$
INS A,BUS	00001000	Введення даних із порту *BUS* в *A*; $A := BUS$
OUTL BUS,A	00000010	Виведення вмісту *A* в порт *BUS*; $BUS := A$
	Команди передачі управління	
JMP a	aaa00100 aaaaaaaa	Безумовний перехід $PC[10..0] := a[10..0]\ ;\ PC[11] := MB$
JMPP @A	10110011	Безумовний перехід в межах поточній сторінці; $PC[7..0] := (A)$
JC a	11110110 aaaaaaaa	Перехід, якщо $C = 1\ \ PC[7-0] := a$ інакше $PC := PC + 2$
JNC a	11100110 aaaaaaaa	Перехід, якщо $C = 0$
DJNZ Rr,a	11101rrr aaaaaaaa	Декремент вмісту регістру і перехід, якщо вміст регістру не дорівнює нулю
JZ a	11000110 aaaaaaaa	Перехід, якщо вміст *A* дорівнює нулю
JNZ a	10010110 aaaaaaaa	Перехід, якщо вміст *A* не дорівнює нулю

Продовження табл. 3.5

1	2	3
JF0 a	10110110 aaaaaaaa	Перехід, якщо $F0 = 1$
JF1 a	01110110 aaaaaaaa	Перехід, якщо $F1 = 1$
JT0 a	00110110 aaaaaaaa	Перехід, якщо $T0 = 1$
JNT0 a	00100110 aaaaaaaa	Перехід, якщо $T0 = 0$
JT1 a	01010110 aaaaaaaa	Перехід, якщо $T1 = 1$
JNT1 a	01000110 aaaaaaaa	Перехід, якщо $T1 = 0$
JTF a	00010110 aaaaaaaa	Перехід, якщо $TF = 1$
JNI a	10000110 aaaaaaaa	Перехід, якщо $INT = 0$
JBb a	bbb10010 aaaaaaaa	Перехід, якщо розряд *Bb* акумулятора встановлений в одиницю
CALL a	aaa10100 aaaaaaaa	Виклик підпрограми; $SP := SP + 1$; $(SP) := PSW[7..4]$; $PC[11] := MB$; $PC[10..0] := a[10..0]$
RET	10000011	Повернення із підпрограми; $SP := SP - 1$; $PC := (SP[11..0])$

Продовження табл. 3.5

1	2	3
RETR	10010011	Повернення із підпрограми з встановленням стану; $SP := SP - 1$; $PC := SP[11..0]$; $PSW[7..4] := (SP[15..12])$
Команди управління режимами роботи		
ENTO CLK	01110101	Дозвіл видачі імпульсів синхрон. на *TO*
SEL MBO	11100101	Вибір нульового банку пам'яті програм; $MB := 0$
SEL MB1	11110101	Вибір першого банку пам'яті програм; $MB := 1$
SEL RBO	11000101	Вибір нульового банку регістрів пам'яті даних; $RB := 0$
SEL RB1	11010101	Вибір першого банку регістрів пам'яті даних; $RB := 1$
NOP	00000000	Немає операції
EN I	00000101	Дозвіл зовнішніх переривань
DIS I	00010101	Заборона зовнішніх переривань
EN TCNTI	00100101	Дозвіл переривань від таймера/лічильника
DIS TCNTI	00110101	Заборона переривань від таймера/лічильника
STRT T	01010101	Запускання таймера/лічильника в режимі таймера
STRT CNT	01000101	Запускання таймера/лічильника в режимі лічильника
STOP TCNT	01100101	Зупинка таймера/лічильника

Обмін даними між *A* та комірками РПД або ЗПД здійснюється з використанням непрямої адресації. При цьому, покажчики адреси розміщуються в регістрах *R*0 або *R*1 вибраного банку регістрів.

За неявної адресації у коді команди неявно (за замовчуванням) указується один з операндів. Найчастіше таким операндом є акумулятор. За безпосередньої адресації у тілі команди в якості другого байту вказується безпосередній операнд (константа), який пересилається за місцем призначення, визначеним першим операндом.

До пам'яті програм здійснюється доступ лише у напряму читання даних.

Всі команди, окрім `MOV PSW, A`, не впливають на встановлення ознак.

Більшість команд виконує пересилку восьмибітних даних. Декілька команд оперують з чотирибітними операндами (тетрадами) і застосовуються для звернення до чотирибітних портів вводу/виводу *P*4, *P*5, *P*6, *P*7.

В МК48 передача даннях відбувається двох режимах: пересилки (завантаження) и обміну. Під час пересилки дані передаються від джерела до приймача, при цьому джерело даних не змінює свого вмісту. Обмін припускає одночасну передачу даних в обох напрямах при цьому змінюються значення обох операндів, що приймають участь в обміні.

Команди пересилки даних всередині МК48 виконуються за один машинний цикл, а обмін даними з ЗПД потребує двох машинних циклів.

Приклади команд:

```
MOV   A,Rr    ; (A) := Rr , r = (7 – 0); пряма адресація.
MOV   A,#d    ; A := d ; безпосередня адресація.
MOV   A,#05   ; A := 05; безпосередня адресація.
MOV   Rr,#d   ; (Rr) := d , r = (7 – 0).
MOV   A,PSW   ; (A) := PSW.
MOV   A,T     ; (A) := T .
MOV   A,@R0   ; (A) := ((Rr)), r = 0,1; непряма адресація.
XCH   A,Rr    ; (A) ↔ (Rr), r = (7 – 0).
XCH   A,#Rr   ; (A) ↔ (Rr), r = (7 – 0).
```

3.5.4. Розробка підпрограм виконання складних арифметичних операцій

Приклад 3.15: Розробити програму множення цілих восьмирозрядних чисел $Z = X \times Y$, де $(X, Y) < 1$. Множення реалізувати першим способом. У вихідному стані операнди подані в прямому коді.

Множення першим способом відбувається молодшими розрядами множника зі зсувом суми часткових добутків в сторону молодших розрядів за нерухомим множеним.

Операційна схема множення першим способом та алгоритм зображені на рис. 3.36 та рис.3.37. Цифрова діаграма стану регістрів зображена на рис. 3.38.

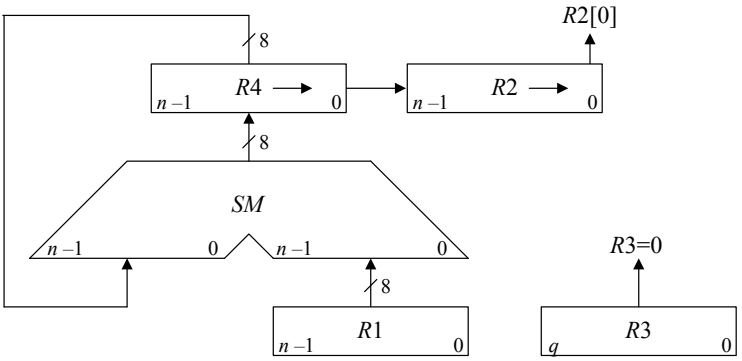

Рис. 3.36. Операційна схема множення чисел першим способом

Рис. 3.37. Алгоритм виконання операції множення

	$\rightarrow R4$ [0 .. n]		$\rightarrow R2(X)$ [0 .. n]		$R1(Y)$ [0 .. n]		$R3$(Лічильник)
	0	0000	0	1011	0	1111	101
1	0	0000			0	1111	
	0	1111					
	0	1111					
	0	0111	1	0101			100
2	0	0111			0	1111	
	0	1111					
	1	0110					
	0	1011	0	1010			011
3					0	1111	
	0	0101	1	0101			010
4	0	0101			0	1111	
	0	1111					
	1	0100					
	0	1010	0	1010			001
5					0	1111	
	0	0101	0	0101			000

Рис.3.38. Цифрова діаграма виконання операції множення

```
; У вихідному стані операнд знаходиться в регістрі R1 та R2.
; Після виконання операції множення старші розряди добутку
; збережені в регістрі R4, молодші – в регістрі R2.
; R3 – лічильник циклів.
; Визначення знаку результату і подання операндів в ПК
      CLR   C          ; обнуління ознаки C
      CLR   F1         ; обнуління ознаки F1
      XCH   A,R1       ; обмін A та R1
      XRL   A,R2       ; ВИКЛЮЧНЕ АБО вмісту A та R2
      JB7   ll1        ; визначення знаку результату
      JMP   ll2
Ll1:  CPL   F1         ; встановлення ознаки F1
Ll2:  MOV   A,R2       ; A:=R2
      ANL   A,#7Fh     ; встановлення знакового розряду R2 в
                       ; нуль
      MOV   R2,A       ; R2:=A
      MOV   A,R1       ; A:=R1
```

```
      ANL   A,#7Fh     ; встановлення знакового розряду R1 в
                       ;  нуль
      MOV   R1,A       ; R1:=A
; зсув множника вправо
Ll4:  MOV   A,R2       ; A:=R2
      CLRC
      RRC   A          ; зсув
      MOV   R2,A       ; R2:=A
      JNC   Ll3        ; аналіз цифри множника, перехід якщо
                       ; A[0]=0
; додавання множника до суми часткових добутків
      MOV   A,R4       ;
      ADD   A,R1       ; A := R4 + R1
      MOV   R4,A       ; R4 := A
; зсув суми часткових добутків
Ll3:  CLR   C          ; C = 0
      MOV   A,R4       ; A := R4
      RRC   A          ; зсув вправо, C := A[0]
      MOV   R4,A       ; R4:=A
; перевірка вмісту лічильника циклів, та
; повернення на початок циклу, якщо R3 ≠ 0
      DJNZ  R3,ll4     ; аналіз лічильника циклів
; перевірка знаку результату, якщо F1 = 1 встановлення
; в одиницю старшого розряду регістру результату
      JF1   Ll5                    ; аналіз знаку результату
      JMP   Ll6
Ll5:  MOV   A,R4                   ;A:=R4
      ORL   A,#80h                 ;R4[7]:=1
      MOV   R4,A                   ;R4:=A
Ll6:  NOP
      END.
```

Приклад 3.21: Розробити для МК48 програму обчислення квадратного кореня $A = \sqrt{B}$, де $0 \le B < 1$.

Найбільш простий алгоритм обчислення квадратного кореня з *n-розрядної* мантиси числа зводиться до підбору цифр результату розряд за розрядом, починаючи із старшого 2^{-1} розряду. При цьому

обчислення *i*-ї цифри результату *X* відбувається таким чином. Після отримання чергової $(i-1)$-ї цифри в *i-й* розряд *A* розміщується одиниця. Обчислюється різниця $(B-A_i^2)=R_i$. Якщо, $R_i \geq 0$ то і A_i є число, де цифри всіх розрядів співпадають з цифрами результату *A*. Якщо, $R_i < 0$ то в і-му розряді необхідно поставити нуль і переходити до обчислення $(i+1)$ -го розряду. Оскільки в цьому випадку обчислення знову починається з підстановки пробної одиниці, то замість заміни одиниці на нуль в *i* -му розряді віднімається одиниця з $(i+1)$ -го.

Виконання обчислення з точністю до шостого розряду приведене на діаграмі (рис. 3.42, *а*). Для досягнення регулярності обчислень у разі отримання додатної різниці операцію віднімання доцільно замінити підсумуванням зворотного коду наступного результату з дописаними цифрами 11, при цьому отримаємо підсумовування в доповнювально коді, наприклад, діаграма (рис. 3.42, *б*).

Для виконання обчислень в МК48 застосовуються восьмирозрядні регістри, тому корінь з восьмирозрядної мантиси можливо обчислити тільки з точністю до чотирьох розрядів після коми.

Операційна схема пристрою для обчислення кавадраткого кореня зображена на рис. 3.43, цифрова діаграма та алгоритм обчислення на рис. 3.44, та 3.45 відповідно.

```
0     00 10010001
    – 00 01
 1    00 01010001
      00 10100010 зсув
    – 00 101
 1    00 00000010
      00 00000100 зсув
    – 00 1101
 0    11 00110100
      10 01101000 зсув
    + 00 11011
 0    11 01000000
      10 10000000 зсув
    + 00 110011
```

```
0     00 10010001
    + 11 11
 1    00 01010001
      00 10100010 зсув
    + 11 011
 1    00 00000010
      00 00000100 зсув
    + 11 0011
 0    11 00110110
      10 01101100 зсув
    + 00 11011
 0    11 01000100
      10 10001000 зсув
    + 00 110011
```

```
0    11|01001100          0    11|01010100
     10|10011000 зсув          10|10101000 зсув
   + 00|1100011           + 00|1100011
0    11|01011110          0    11|01011110
```

$A = \sqrt{B} = 0{,}110000$ $A = \sqrt{B} = 0{,}110000$

а *б*

Рис. 3.42. Діаграма обчислення квадратного кореня: *а* – у прямому коді; *б* – у доповнювальному коді.

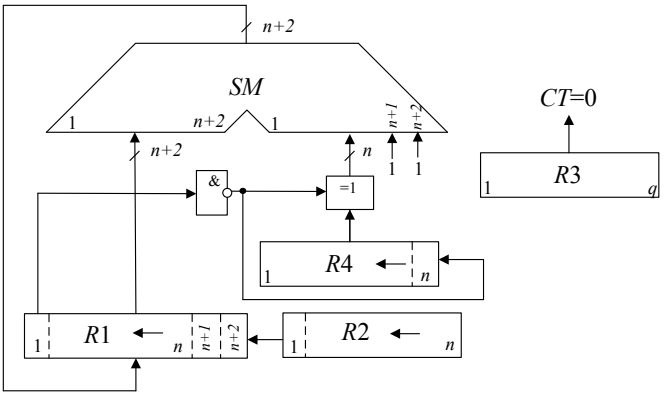

Рис. 3.43. Операційна схема обчислення квадратного кореня

№	← *RA*	← *RB*		← *RC*	*CT*	Мікрооперації
	[1 .. *n*]	[1 .. *n*]	*n*+1 *n*+2	[1 .. *n*]		
ПС	000000	000000	00	10010001	100	
1	000000	000000	10	01000100		2(*RGC* := *l*[*RGC*].0, *RGB* := *l*[*RGB*].*RGC*(0))
		+111111	11			*RGB* := *RGB* + $\overline{RGA}$.11
		000000	01			
2	00000,1	000000	10	10001000		*RGC* := *l*[*RGC*].0, *RGB* := *l*[*RGB*].*RGC*(0), *RGA* := *l*[*RGA*].*RGB*(0)
		000001	01	00010000		*RGC* := *l*[*RGC*].0, *RGB* := *l*[*RGB*].*RGC*(0)
		+111110	11			*RGB*(0) = 0 ⇒ *RGB* := *RGB* + $\overline{RGA}$.11

		000000	00		011	$CT := CT - 1; CT \neq 0$
3	0000,11	000000	00	00100000		$RGC := l[RGC].0,$ $RGB := l[RGB].\underline{RGC}(0),$ $RGA := l[RGA].RGB(0)$
		000000	00	01000000		$RGC := l[RGC].0,$ $RGB := l[RGB].RGC(0)$
		+111100	11			$RGB(0) = 0 \Rightarrow \_\_\_\_$ $RGB := RGB + RGA.11$
		111100	11		010	$CT := CT - 1; CT \neq 0$
4	000,110	111001	10	10000000		$RGC := l[RGC].0,$ $RGB := l[RGB].\underline{RGC}(0),$ $RGA := l[RGA].RGB(0)$
		110011	01	00000000		$RGC := l[RGC].0,$ $RGB := l[RGB].RGC(0)$
		+000110	11			$RGB(0) = 1 \Rightarrow$ $RGB := RGB + RGA.11$
		111010	00		001	$CT := CT - 1; CT \neq 0$
5	00,1100	110100	00	00000000		$RGC := l[RGC].0,$ $RGB := l[RGB].\underline{RGC}(0),$ $RGA := l[RGA].RGB(0)$
		101000	00	00000000		$RGC := l[RGC].0,$ $RGB := l[RGB].RGC(0)$
		+001100	11			$RGB(0) = 1 \Rightarrow$ $RGB := RGB + RGA.11$
		110100	11		000	$CT := CT - 1; CT = 0$

Рис.3.44. Цифрова діаграма обчислення квадратного кореня

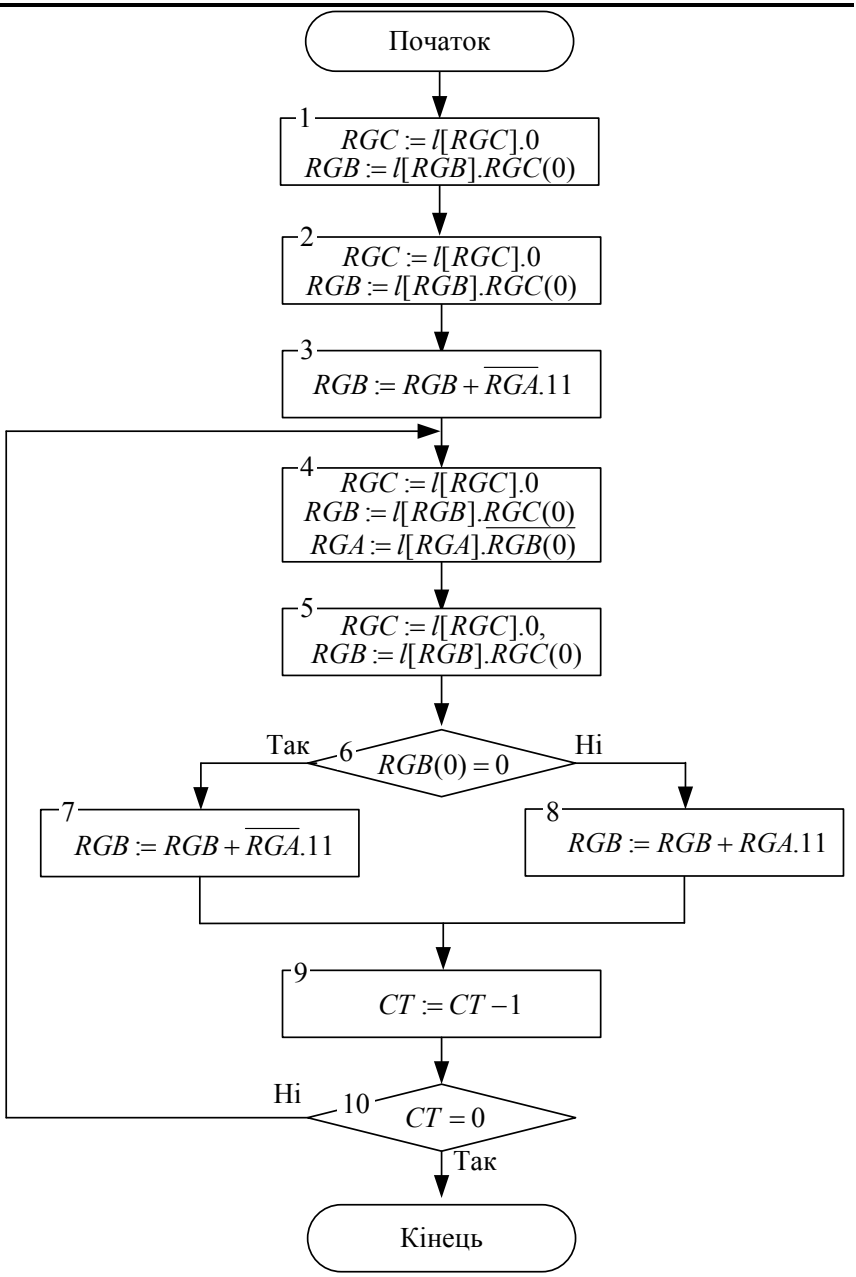

Рис. 3.45. Алгоритм обчислення квадратного кореня

```
; обчислення кореня з восьмирозрядної мантіси числа
; з точністю до чотирьох знаків після коми
      INS   A,BUS          ;
      MOV   R2,A           ; A – завантаження X
      MOV   R2,#10010001   ;
            B
      MOV   R1,#0h         ; B – для підсумовування
      MOV   R0,#0h         ; C – для
                           ; зберігання реультата
; завдання кількості повторень циклу CT:=4
      MOV   R3,#4h
LL3:  MOV   A,R0
      MOV   R4,A       ; R4:=A(R4:=A.11)
      MOV   R5,#0h     ; B[ZN]:=0
; два зсуви вліво A, B
      CLR   C
      MOV   A,R2
      RLC   A
      MOV   R2,A
      MOV   A,R1
      RLC   A
      MOV   R1,A
      CLR   C
      MOV   A,R2
      RLC   A
      MOV   R2,A
      MOV   A,R1
      RLC   A
      MOV   R1,A
      JB7   LL1        ; аналіз B[ZN]
      MOV   A,R4
      CPL   A          ; R4:=!A
      MOV   R4,A
LL1:  MOV   A,R4
      RLC   A
      RLC   A
      ORL   A,#3h
      MOV   R4,A       ; R4:=A.11
      ADD   A,R1
```

```
      MOV   R1,A
      JB7   LL2
      MOV   R5,#1h     ; !B[ZN]:=1
LL2:  CLR   C
      MOV   A,R0
      RLC   A
      ADD   A,R5       ; A[N]:=!B[ZN]
      MOV   R0, A
      DJNZ  R3,LL3     ; перевірка циклу
      MOV   A,R0
      OUTL  BUS,A
      END
```

3.5.5. Розробка програм управління

Приклад 3.14: Розробити програму реалізації алгоритму управління, що заданий логічною схемою алгоритм (ЛСА).

Вихідні дані:

- Алгоритм управління:

$$\text{Н}\ Y_5 X_1 \overset{1}{\uparrow}(Y_1 Y_2) X_2 \overset{2}{\uparrow}\overset{1}{\downarrow}(Y_3 Y_4)\overset{2}{\downarrow}\ \text{К}$$

- Тривалість управляючих сигналів:

$$T(y_1) = T(y_2) \geq 240 мкс,\ \text{T}(\text{y}_3) = T(y_4) \geq 30 мкс,\ \text{T}(\text{y}_5) = 450 мкс$$

Алгоритм управління зображений на рис. 3.46.

Для вводу і виводу сигналів будемо використовувати порт *P*1, причому, розряди порту *P*1[6] та *P*1[7] в початковому стані налаштовані на ввід ($P1[6,7]=1$), а *P*1[5..0] – на вивід інформації. Відповідність виходів порту і сигналів вказано в табл. 3.6.

Таблиця 3.6. Відповідність виходів порту і сигналів

Розряд порту	*P*17	*P*16	*P*15	*P*14	*P*13	*P*12	*P*11	*P*10
Сигнал	*X*1	Вхід	*Y*1	*Y*2	*Y*3	*Y*4	*Y*5	Вихід

Для формування потрібної тривалості сигналів *Y*1 і *Y*2 будемо використовувати *TCNT* в режимі таймера. При *F*=6 *МГц* для відліку проміжку часу, не меншого 240 мкс, необхідна зміна змістовного *TCNT* на 3. Для формування інтервалу часу, тривалістю 30 мкс

(керуючі сигнали *Y*3 та *Y*4), при вказаній частоті *F* потрібно виконати 12 командних циклів. Для формування затримки в 450 мкс (*Y*5), необхідно використовувати і таймер, і тривалість команди.

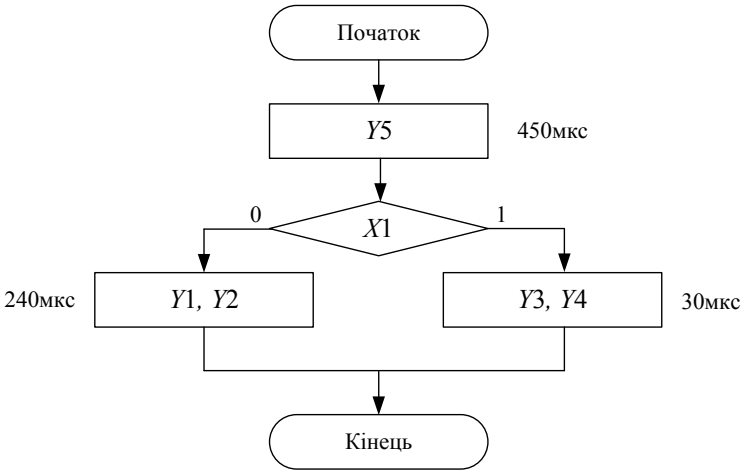

Рис. 3.46 Алгоритм управління

```
;Перед виконанням програми порт P1 знаходиться в стані 11111111.
            MOV   R5,#4H      ; завантаження константи 4 в R5
            MOV   A,#FDH      ; завантаження константи
                              ; (– 3)ДК в таймер
            MOV   T,A
            ORL   P1,#18H     ; встановлення сигналів
                              ; Y3 = Y4 = 1
Loop:       NOP               ; відлік часового інтервалу
            ANL   P1,#C0h     ;P1:=11000000
            MOV   R5,#10      ; завантаження константи 10
                              ; в R5
            MOV   A,#FBh      ;завантаження константи
                              ; (– 5)ДК в таймер
            MOV   T,A
            ORL   P1,#2       ; встановлення сигналу Y5 = 1
            STRT  T           ; запуск таймера
Label2:     JTF   Label1      ; відлік часового інтервалу
            JMP   Label2      ; з використанням таймеру
                              ; (400мкс)
Label1:     DJNZ  R5,label1   ; відлік часового інтервалу
```

```
                                ; з використанням R5(50мкс)
          ANL   P1,#C0h         ; зняття управляючого сигналу
          MOV   A,#FDh          ; завантаження константи
                                ; (– 3)дк в таймер
          MOV   T,A
          MOV   R5,#4           ; завантаження константи 4 в R5
          IN    A,P1            ;ввід та перевірка
          JB7   Label3
          ORL   P1,#30h         ; встановлення сигналів
                                ; Y1 = Y2 = 1
          STRT  T               ; запуск таймера
Label4:   JTF   Label5          ; відлік часових інтервалів
          JMP   Label4          ; з використанням таймеру
Label5:   ANL   P1,#C0h         ; зняття керуючих сигналів
                                ; Y3 = Y4 = 1
Label3:   NOP                   ; відлік часового інтервалу
          DJNZ  R5,label3       ; з використанням R5
          ANL   P1,#C0h         ; зняття управляючих сигналів
          END.
```

***Приклад 3.18*:** Розробити структурну схему підключення до МК48 програмованого периферійного адаптера ВВ55. Розробити програму пересилки даних із порту *PB* в порти *PA* та *PC*. Адреси портів ППА належать загальному адресному простору зовнішньої пам'яті даних.

Вихідні дані:

- Адреси портів:

  PA – ACh, PB – ADh, PC – AEh, Регістр УСРР – *AFh.*

Структурна схема підключення до МК48 однієї сторінки ЗПД та програмованого периферійного адаптера ВВ55 зображена на рис. 3.47. Для підключення ППА застосовується селектор адреси СА.

; Прийом байту із порту *PB* в порти *PA* та *PC*.

```
MOV       R0,#0AFh      ; адресу Регістру УСРР завантажуємо у
                        ; покажчик адреси
```

```
MOV     A,#082h         ; ініціалізація ВВ55 (порти PA і PC
                        ; налаштовуються на вивід, а порт PB –
                        ; на ввід даних)
MOV     @R0,A
MOV     R0,#0ADh        ; підготовка адреси порта PB
MOVX    A, @R0          ; читання даних із порту PB
MOV     R0,#0ACh        ; підготовка адреси порта PA
MOV     @R0,A           ; запис даних в порт PA
MOV     @R1,#0AEh       ; підготовка адреси порту PC
MOV     @R1,A           ; запис даних в порт PC
END
```

***Приклад 3.19*:** Розробити структурну схему підключення до МК48 ЗПД та програмованого зв'язувального адаптера ВВ51. Зовнішня пам'ять даних складається з чотирьох сторінок, для перемикання між якими застосовуються два молодших виводи порту *P*1. Адреси регістрівУСРР та УСК ПЗА належать першій сторінці ЗПД (адреса РД -00h; адреса РС(УС) -01h)

Розроблена структурна схема МПС зображена на рис. 3.48.

```
; P1 = 11111111 встановлюється під час ініціалізації
        ANL     P1,#0FDh ; вибір номера сторінки
        ORL     P1,#01h  ; "1" 11111101 для ВВ51
        SEL     RB1      ; установка першого банку
                         ; регістрів
        MOV     R0,#00h  ; адреса регістра даних ПЗА
                         ; завантажується в R0
        MOV     R1,#01h  ; адреса Регістру УСРР (УСРР,
                         ;УСІ) ПЗА завантажується в R1
        MOV     A,#07Eh  ; ініціалізація ПЗА
                         ; (асинхронний режим).
        MOVX    @R1,A    ; передача байта в послідовний ;
                         канал з регістра R2
        MOV     A;#31h   ; запис УСК (УСІ) в PG ПЗА
        MOVX    @R1,A    ;
WW1:    MOVX    A,@R1    ; читання слова стану ПЗА
                         ; і перевірка готовності передавача
        ANL     A,#01
        JZ      WW1      ;
```

```
        MOV     A,R2        ; запис даних в передавач ПЗА
                            ; з R2
        MOVX    @R0,A       ; прийом байта з послідовного ;
                            каналу в R2
        MOV     A,#34h      ; запис управляючого слова в ПЗА
        MOVX    @R1,A       ;
WW2:    MOV     X,a,@R1     ; читання слова стану з ПЗА
                            ; і перевірка готовності приймача
        ANL     a,#02h
        JZ      WW2
        MOVX    A,@R1       ; читання стану слова в ПЗА
                            ; і перехід, якщо знайдена помилка
        ANL     A,#38h
        JNZ     ERROR
        MOVX    A,@R0       ; читання байта даних та передача
                            ; вR2
        MOV     R2,A        ;
ERROR:  NOP                 ; обробка помилки
        END
```