

2. МІКРОПРОЦЕСОРИ НА ОСНОВІ СЕКЦІОНОВАНИХ ІНТЕГРАЛЬНИХ СХЕМ

2.1. Загальні відомості

Секціоновані мікропроцесорні серії інтегральних схем (ІС) призначені для розробки МПС з будь-якою системою команд і довільною розрядністю.

Найпоширенішим набором таких ІС є серія К1804, яка складається з наступних ІС:

К1804ГГ1	– генератор тактових сигналів
К1804ВА1, К1804ВА2, К1804ВА3	– чотирирозрядні приємопередавачі;
К1804ВЖ1	– шістнадцятирозрядна схема виправлення помилок в ЗП;
К1804ВН1	– блок пріоритетного переривання на 8 запитів;
К1804ВР1	– схема прискореного переносу;
К1804ВР2	– блок управління станами та зсувами;
К1804ВР3	– розширювач блока пріоритетного переривання;
К1804ВС1, К1804ВС2	– чотирирозрядні процесорні елементи;
К1804ВУ1, К1804ВУ2	– чотирирозрядні генератори адрес мікрокоманд;
К1804ВУ3	– схема управління генератором адрес мікрокоманд;
К1804ВУ4	– дванадцятирозрядний генератор адрес мікрокоманд (формував адреси наступної мікрокоманди);
К1804ВУ5	– чотирирозрядний генератор адрес мікрокоманд;
К1804ІР1	– чотирирозрядний регістр;
К1804ІР2, К1804ІР3	– восьмирозрядні регістри.

Розглянемо основні ІС серії К1804 таведемо особливості проектування мікропроцесорних систем на їх основі.

2.2. Процесорний елемент

ІС К1804ВС1 виконана за технологією ТТЛШ і є чотирирозрядним універсальним процесорним елементом (ПЕ), призначеним для

- АЛБ – арифметико-логічний блок;
НОЗП – надоперативний запам'ятовуючий пристрій, що складається з шістнадцяти регістрів $R15 - R0$;
 RA, RB – регістри тимчасового зберігання адрес операндів за каналом A і B ;
 $ЗCQ, ЗCB$ – зсувачі;
 MS, MR, MY – мультиплексори вибору операндів S і R та даних Y
 CU – схема управління.

Умовне позначення ПЕ зображене на рис 2.2, призначення виводів мікросхеми описане в табл. 2.1.

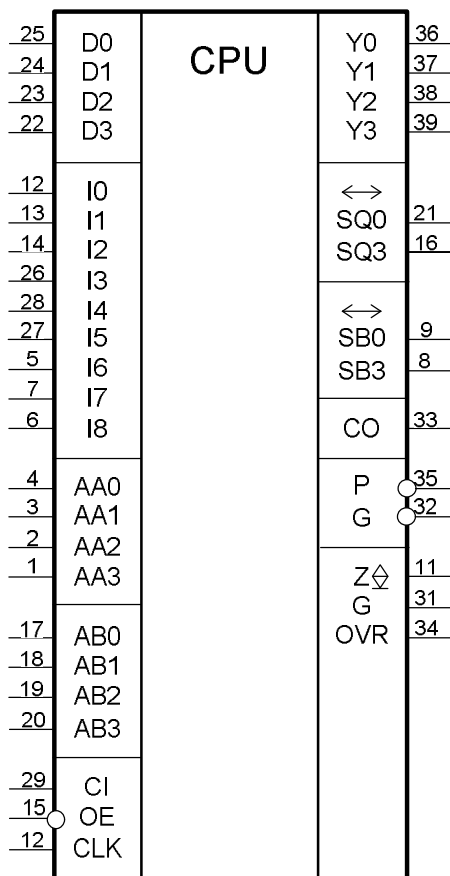


Рис 2.2. Умовне графічне позначення мікросхеми ПЕ K1804BC1

Молодший розряд ПЕ має номер 0, а старший – номер 3.

Виходи Y мають три стани (0, 1 і високоомний). Встановлення в високоомний стан цих виходів здійснюється подачею на вхід OE одиничного потенціалу $\overline{OE} = 1$.

Виводи $SQ3$, $SQ0$, $SB3$ і $SB0$ є двоспрямованими і мають також три стани. При виконанні на зсувачах $3CB$ і $3CQ$ зсуву вправо виводи $SB3$ і $SQ3$ є входами, а виводи $SB0$ і $SQ0$ – виходами. При зсуві інформації вліво функції виводів змінюються на протилежні. Якщо зсув на зсувачі не виконується, то відповідні йому виводи знаходяться в високоомному стані. Вихід Z виконаний за схемою з відкритим колектором. Решта виходів IC мають два стани (0 та 1).

Таблиця 2.1. Функціональне призначення виводів мікросхеми ПЕ К1804BC1

Номери виводів	Позначення	Функціональне призначення
1 – 4	$AA3 - AA0$	Входи адрес регістрів НОЗП за каналом A
12 – 14, 26, 28, 27, 5 – 7	$I8 - I0$	Входи мікрокоманди
8 (9)	$SB3 (SB0)$	Вхід (вихід) старшого (молодшого) розряду $3CB$
11	Z	Вихід ознаки нульового результату в АЛБ
15	CLK	Вхід синхросигналів
16 (21)	$SQ3 (SQ0)$	Вхід (вихід) старшого (молодшого) розряду $3CQ$.
17-20	$AB0 - AB3$	Входи адреси регістрів НОЗП за каналом B
22 – 25	$D3 - D0$	Входи даних
29	CI	Вхід переносу в АЛБ
31	$F3$	Вихід старшого розряду АЛБ
32,35	G,P	Виходи прискореного переносу
33	CO	Вихід переносу з АЛБ
34	OVR	Вихід ознаки переповнювання результату
36 – 39	$Y0 - Y3$	Виходи даних
40	OE	Вхід дозволу видачі даних
33	CO	Вихід переносу з АЛБ

В ПЕ використовується двоадресний НОЗП, який забезпечує видачу інформації за двома незалежними каналами A і B (рис. 2.1.). Інформація на виходах A і B визначається відповідно адресами AA і AB . Якщо на входах AA і AB встановлені однакові адреси регістрів, то на виходи A і B видається вміст одного й того самого регістру. Адреси регістрів дорівнюють двійковим еквівалентам їх номерів. Наприклад, регістр $R5$ має адресу 0101 і таке інше. Запис інформації в НОЗП відбувається за адресою B – тільки за одним каналом.

Для управління ПЕ в структурі мікрокоманди відведено 18 розрядів (рис. 2.3).



Рис. 2.3. Поля мікрокоманди, які використовуються для управління ПЕ

Характер перетворення інформації в кожному такті роботи ПЕ визначається інформаційним словом (мікроінструкцією МІ), що розміщується в полі АЛБ_МІ і надходить на входи $I8-I0$ мікросхеми K1804BC1. Формат інформаційного слова для управління ПЕ наведений на рис. 2.4.

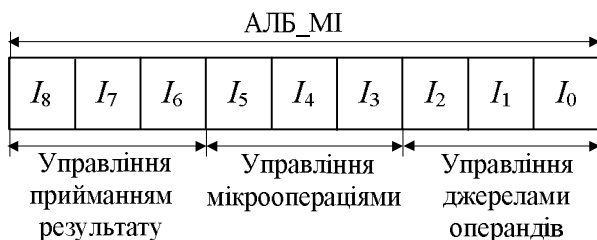


Рис. 2.4. Формат інформаційного слова АЛБ_МІ мікрокоманди

Поле $(I_8I_7I_6)$ управляє мультиплексором MY , зсувачами $ЗСQ$ і $ЗСВ$, а також вибором приймача результату (рис.2.1). Кодування розрядів даного поля наведене у табл. 2.2.

Результат F мікрооперації, що виконувалась в АЛБ, може бути записаний в НОЗП за каналом B ($F \rightarrow B$) без змін, а також у модифікованому вигляді – із зсувом на один розряд вліво ($2F \rightarrow B$) або

вправо ($F/2 \rightarrow B$). За встановлення значення $I_8 I_7 I_6 = 000$ результат записується в регістр RQ ($F \rightarrow Q$). Результат не фіксується в регістрах ПЕ за встановлення значення $I_8 I_7 I_6 = 001$.

Таблиця 2.2. Кодування поля управління мультіплексором

Розряди мікрокоманди			Мікрооперації		Y
I_8	I_7	I_6	НОЗП	RQ	
0	0	0	–	$F \rightarrow Q$	F
0	0	1	–	–	F
0	1	0	$F \rightarrow B$	–	A
0	1	1	$F \rightarrow B$	–	F
1	0	0	$F/2 \rightarrow B$	$Q/2 \rightarrow Q$	F
1	0	1	$F/2 \rightarrow B$	–	F
1	1	0	$2F \rightarrow B$	$2Q \rightarrow Q$	F
1	1	1	$2F \rightarrow B$	–	F

Під час встановлення на вході сигналу дозволу видачі даних \overline{OE} ($\overline{OE} = 1$) та за $I_8 I_7 I_6 = 010$ мультіплексор MY видає на вихідні шини Y інформацію з виходів каналу A НОЗП (з регістру RA), а в інших випадках – з виходів F АЛБ.

Одночасно із записом результату в НОЗП в регістрі RQ може бути виконаний зсув інформації вліво ($I_8 I_7 I_6 = 110$) або вправо ($I_8 I_7 I_6 = 100$).

Розряди $I_5 I_4 I_3$ поля АЛБ_МІ МК (рис.2.4) визначають мікрооперацію в АЛБ відповідно до табл. 2.3, де R і S виходи мультіплексорів MR і MS .

Мікрооперації підсумовування і віднімання виконуються у доповнювальному коді з урахуванням вхідного переносу CI . Логічні мікрооперації – диз'юнкція (АБО), кон'юнкція (І) і сума за модулем два (ВИКЛЮЧНЕ АБО) є порозрядними. Вибір джерел операндів здійснюється за допомогою мультіплексорів MS і MR та управляється розрядами $I_2 I_1 I_0$ поля АЛБ_МІ мікрокоманди (табл. 2.4).

Таблиця 2.3. Кодування поля $I_5 I_4 I_3$

Розряди мікрокоманди			Мікрооперація в АЛБ
I_5	I_4	I_3	
0	0	0	$R + S + CI$
0	0	1	$S - R - 1 + CI$
0	1	0	$S - R - 1 + CI$
0	1	1	$R \vee S$
1	0	0	$R \& S$
1	0	1	$\overline{R} \& S$
1	1	0	$R \oplus S$
1	1	1	$\overline{R \oplus S}$

Таблиця 2.4. Кодування поля $I_2 I_1 I_0$

Розряди мікрокоманди			Джерела операндів	
I_2	I_1	I_0	R	S
0	0	0	A	Q
0	0	1	A	B
0	1	0	0	Q
0	1	1	0	B
1	0	0	0	A
1	0	1	D	A
1	1	0	D	Q
1	1	1	D	0

Такт роботи ПЕ полягає в наступному.

За додатним перепадом CLK на управляючі входи ПЕ подається слово мікрокоманди (рис. 2.1). В регістри RA і RB з відповідних регістрів НОЗП, які визначаються адресами на входах AA і AB , записуються дані. Відповідно до значень розрядів $I_8 - I_0$ мікрокоманди схема управління (СУ) виробляє необхідні управляючі сигнали (УС),

в результаті цього обираються джерела операндів. Далі відбувається перетворення операндів в АЛБ і видається результат на вихід F . Якщо мультиплексор MU відкритий ($\overline{OE} = 0$), результат видається на шину Y . Залежно від результату, одержуваного в АЛБ, на відповідних виходах ПЕ формуються ознаки Z , $F3$, OVR .

Одиничний сигнал на виході Z відповідає нульовому значенню результату. Ознакою арифметичного переповнювання в АЛБ є одиничний сигнал на виході OVR . Вивід $F3$ є виходом старшого розряду АЛБ і використовується для аналізу знаку результату.

За від'ємним перепадом синхросигналу $CLK = 0$ результат фіксується у вибраному приймачеві результату – в регістрах НОЗП або в регістрі RQ . Для забезпечення правильної роботи ПЕ необхідно, щоб сигнали на управляючих входах ПЕ не змінювали свого значення за $CLK = 0$.

Часові параметри ІС K1804BC1 наведені в табл. 2.5.

Таблиця 2.5. Часові параметри ІС K1804BC1

Параметри ІС K1804BC1	Значення, нс	Параметри ІС K1804BC1	Значення, нс
Тривалість сигналу $CLK = 0$ (min)	30	від I до Y , CO , $F3$	50
Час попереднього встановлення відносно додатного перепаду CLK сигналів на входах (min):		від I до G , P	42
RA , RB	93	від I до Z	65
D , I	70	від I до OVR	59
CI	55	від I до $SB3$, $SB0$	70
$SQ3$, $SQ0$, $SB3$, $SB0$	20	від D до Y	39
Час затримки		від B до G , D	31
від RA , RB до y , $F3$	75	від D до Z	55
від RA , RB до G , P	59	від D до CO , $F3$	41
від RA , RB до Z	85	від D до OVR	45
від RA , RB до CO	70	від D до $SB3$, $SB0$	53
від RA , RB до OVR	76	від CI до Y	27
від RA , RB до $SB3$, $SB0$	90	від CI до Z	46
		від CI до CO	20
		від CI до $F3$	24
		від CI до OVR	26
		від CI до $SB3$, $SB0$	45

Приклад 2.1. Розробити мікропрограму для процесорного елементу, що реалізує заданий мікроалгоритм (рис. 2.5).

Вихідні дані: $X1 = -3$, $X2 = -9$, $X3 = 11$.

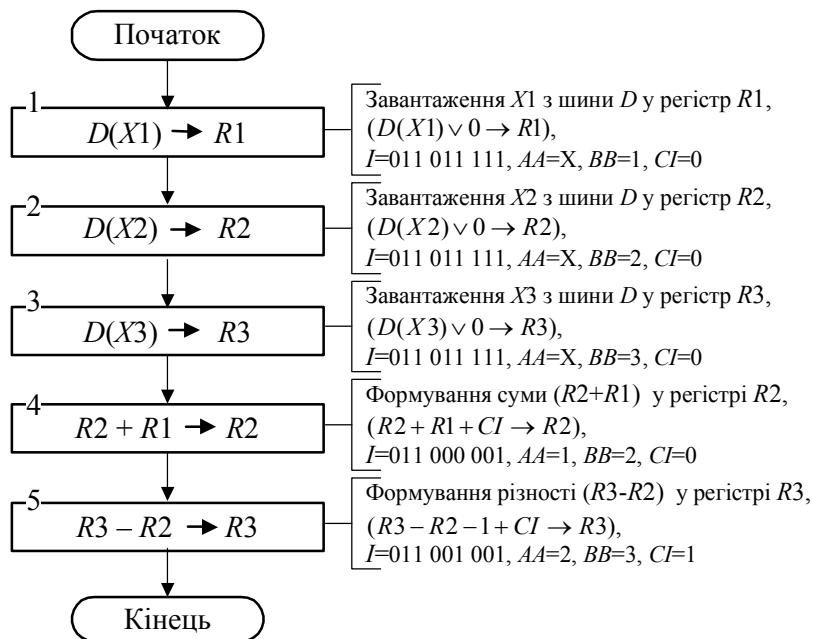


Рис. 2.5. Вихідний мікроалгоритм

Вважатимемо, що для подання операндів використовується доповнювальний код. В коментарях до кожної операторної вершини (рис. 2.5) наведені значення управляючих сигналів АЛБ_МІ, AA , BB , CI , що забезпечують виконання відповідної мікрооперації.

Послідовність мікрокоманд, що реалізують заданий мікроалгоритм, відображена у табл. 2.6. Цифрова діаграма стану регістрів показана у табл. 2.7.

Розглянемо мікрооперацію завантаження $D(X1) \rightarrow R1$, що виконується в першому такті. Зазначимо, що дані в регістри НОЗП можуть бути завантажені лише, як результат виконання мікрооперації в АЛБ. Для завантаження даних з шини даних виконаємо мікрооперацію логічного додавання. Джерелами даних, що надходять на

входи S і R АЛБ (рис. 2.1), є машинний нуль і дані на вхідній шині D ($D(X1) \vee 0 \rightarrow R1$) Відповідно до табл. 2.4. визначаємо розряди $I_2I_1I_0$ поля АЛБ_МС мікрокоманди $I_2I_1I_0 = 111$. Мікрооперація логічне І, що здійснюється у цьому такті, визначається розрядами $I_5I_4I_3 = 011$ поля АЛБ_МС мікрокоманди (табл. 2.3). Результат, отриманий на виході Y АЛБ записується в регістр НОЗП за адресою, виставленою на шині AB . За умовою завдання результат має бути записаний у регістр $R1$, тому у полі B мікрокоманди (рис. 2.9) розміщується адреса відповідного регістру у шістнадцятирічному поданні $B = 1$. При цьому кодуємо розряди $I_8I_7I_6$ поля АЛБ_МС мікрокоманди (табл. 2.2), отримуємо $I_8I_7I_6 = 011$. Мікрооперація додавання, що здійснюється у четвертому такті, визначається розрядами $I_5I_4I_3 = 000$ поля АЛБ_МІ мікрокоманди (табл. 2.3). при цьому встановлюється $CI=0$, що відповідає розрядам $I_{12}I_{11} = 00$ поля СУСЗ_МІ мікрокоманди (табл. 2.19). У полях A і B мікрокоманди (рис. 2.3) розміщуються номери регістрів НОЗП у шістнадцятирічному поданні, що є джерелами операндів. Результат, отриманий на виході Y АЛБ записується в регістр НОЗП за адресою, виставленою на шині AB , тому у полі B мікрокоманди (рис. 2.3) розміщується номер регістра-приймача результату ($A = 1, B = 2$). Мікрооперація віднімання, що здійснюється у п'ятому такті, визначається розрядами $I_5I_4I_3 = 001$ поля АЛБ_МІ мікрокоманди (табл. 2.3). при цьому встановлюється $CI=1$, що відповідає розрядам $I_{12}I_{11} = 01$ поля СУСЗ_МІ мікрокоманди (табл. 2.19).

Наведемо мікропрограму у мнемонічних кодах мікроасемблеру:

```

    /-- Завантаження даних в регістри НОЗП---
    accept r1: 0fffdh /-- (-3)ДК → R1
    accept r2: 0fff7h /-- (-9)ДК → R1
    accept r3: 0000bh /-- (11)ДК → R1

0003 /--Область програми-----
0004 {add r2, r2, r1, z; } /--Обчислення суми
    {sub r3, r2, r3, nz; } /--Обчислення різниці
    {}

```

Таблиця 2.6. Кодова карта

№ так- ту	Ад- реса [16]	Константа БОД		БОД													
				BC1 (АЛБ)				BP2 (СУС3)									
		D [16]	\overline{OED}	АЛБ MI [2]	A [16]	B [16]	\overline{OEY}	СУС3 MI [2]	\overline{EC}	\overline{EZ}	\overline{EN}	\overline{EV}	\overline{CEN}	\overline{CEM}	\overline{OECT}	\overline{SE}	
1	0000	FFFD	0	011 011 111	*	1	0	00 00000 000000	0	0	0	0	1	1	1	1	
2	0001	FFF7	0	011 011 111	*	2	0	00 00000 000000	0	0	0	0	1	1	1	1	
3	0002	000B	0	011 011 111	*	3	0	00 00000 000000	0	0	0	0	1	1	1	1	
4	0003	****	1	011 000 001	1	2	0	00 00000 000000	0	0	0	0	1	1	1	1	
5	0004	****	1	011 001 001	2	3	0	01 00000 000000	0	0	0	0	1	1	1	1	

Таблиця 2.7. Цифрова діаграма стану регістрів

№ так-ту	Вихідний стан				Інформація у приймачі результату, [16]	Мікрооперація
	Шина D , [16]	$R1$, [16]	$R2$, [16]	$R3$, [16]		
1	FFFD	****	****	****	$R1 := FFFD$	$R1 := D(FFFD) \vee 0$
2	FFF7	FFFD	****	****	$R2 := FFF7$	$R2 := D(FFF7) \vee 0$
3	000b	FFFD	FFF7	****	$R3 := 000B$	$R3 := D(000B) \vee 0$
4	****	FFFD	FFF7	000B	$R2 := FFF4$	$R2 := R2 + R1 + 0(CI)$
5	****	FFFD	FFF4	000B	$R3 := 0017$	$R3 := R2 - R3 - 1 + 1(CI)$

2.3. Схема управління станами та зсувами

2.3.1. Загальна структура та принцип функціонування

Схема управління станами та зсувами (СУСЗ) призначена для спільної роботи з ПЕ K1804BC1, разом з якими складає блок обробки даних (БОД). Схема управління станами та зсувами реалізована на ІС K1804BP2, структурна схема якої наведена на рис. 2.6. Загальна структура БОД та поєднання виводів мікросхем K1804BP2 та наведені на рис. 2.7.

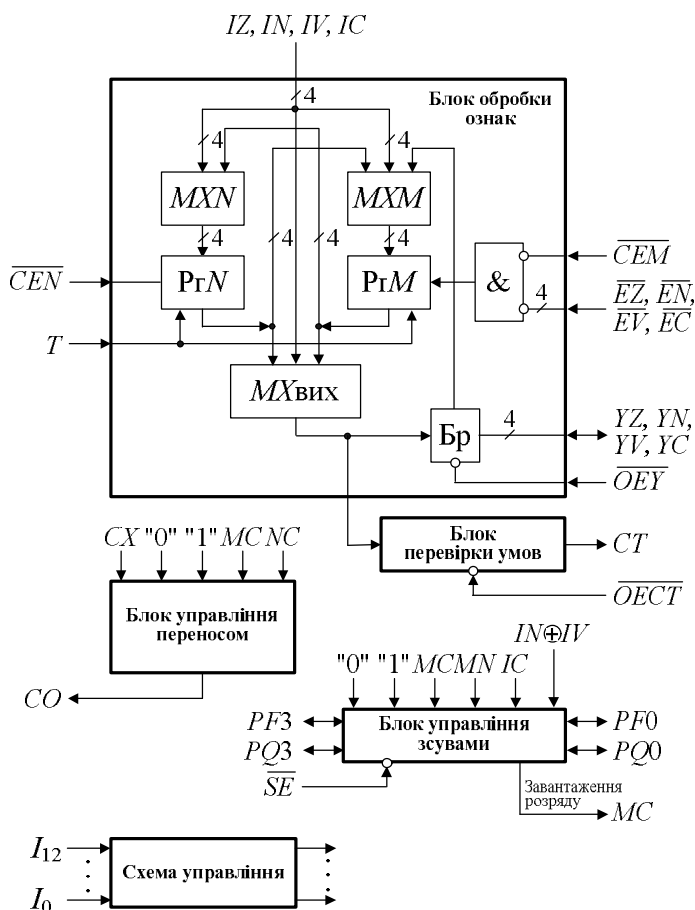
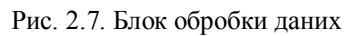


Рис. 2.6. Структурна схема СУСЗ ІС K1804BP2



Для управління БОД у структурі мікрокоманди відведено 43 розряди. На рис. 2.8 зображені відповідні поля МК.

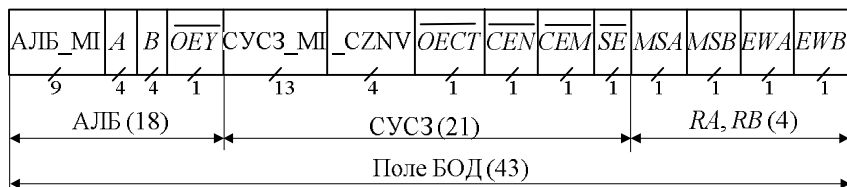


Рис. 2.8. Поля мікрокоманди, які використовуються для управління блоком обробки даних

Умовне графічне позначення ІС К1804ВР2 та нумерація виводів приведені на рис. 2.9, а їх функціональне призначення – в табл. 2.7.

Таблиця 2.7. Функціональне призначення виводів ІС К1804ВР2

Позначення	Назва	Призначення
$I_{12} - I_0$	Інформаційні входи МК	Визначають операцію, що виконується СУСЗ
IC, IN, IV, IZ	Входи ознак стану	Використовуються для передачі ознак стану АЛБ (переносу, знаку, переповнювання, рівність нулю) в СУСЗ
\overline{CEM}	Вхід дозволу запису ознак в RgM	При значенні 0 на вході \overline{CEM} ($\overline{CEM} = 0$) запис в RgM дозволений, інакше ($\overline{CEM} = 1$) – заборонений
\overline{CEN}	Вхід дозволу запису ознак в PrN	При значенні 0 на вході \overline{CEN} ($\overline{CEN} = 0$) запис в PrN дозволений, інакше ($\overline{CEN} = 1$) – заборонений
$\overline{EC}, \overline{EN}, \overline{EV}, \overline{EZ}$	Входи дозволу запису ознак	При значенні 1 на входах $\overline{EC}, \overline{EN}, \overline{EV}, \overline{EZ}$ запис у відповідні розряди RgM заборонений, а при значенні 0 – дозволений, якщо одночасно встановлений сигнал дозволу запису $\overline{CEM} = 0$

YC, YN, YV, YZ	Двоспрямова-на шина даних	Застосовується як вхід даних під час запису в регістр RgM і як вихід даних під час виводу інформації з регістрів PrN, RgM або з входів IC, IN, IV, IZ СУСЗ
\overline{OEY}	Вхід управляючого сигналу дозволу виводу інформації	За значення 0 на вході \overline{OEY} ($\overline{OEY} = 0$) дозволений вивід інформації зі СУСЗ через шину Y . При $\overline{OEY} = 1$ шина Y знаходиться в стані високого опору – вивід даних на шину Y заборонений
\overline{OECT}	Вхід управляючого сигналу дозволу формування коду умови	За значення 0 на вході \overline{OECT} ($\overline{OECT} = 0$) дозволений вивід коду умови на вихід CT . Якщо $\overline{OECT} = 1$, формування коду умови на виході CT заборонене
CT	Вихід умови	Результат перевірки умови
$PF0, PF3, PQ0, PQ3$	Двоспрямовані виводи зсуву	Призначені для організації зсувів в МПС шляхом поєднання їх із відповідними виводами зсувів
\overline{SE}	Вхід дозволу зсуву	За значення 0 на вході \overline{SE} ($\overline{SE} = 0$) виконання зсувів дозволено, інакше ($\overline{SE} = 1$) – заборонено, під час цього виводи зсуву СУСЗ знаходяться в стані високого опору
CO	Вихід формування переносу	Сформований на виході CO сигнал використовується в якості вхідного переносу CI АЛБ
CX	Вхід формування переносу	Використовується як одне з джерел при формуванні сигналу вихідного переносу CO
CLK	Тактовий вхід	Запис інформації в регістр стану відбувається за додатним перепадом тактового сигналу CLK . Решта схем є комбінаційними – їх функціонування не залежать від значення сигналу CLK

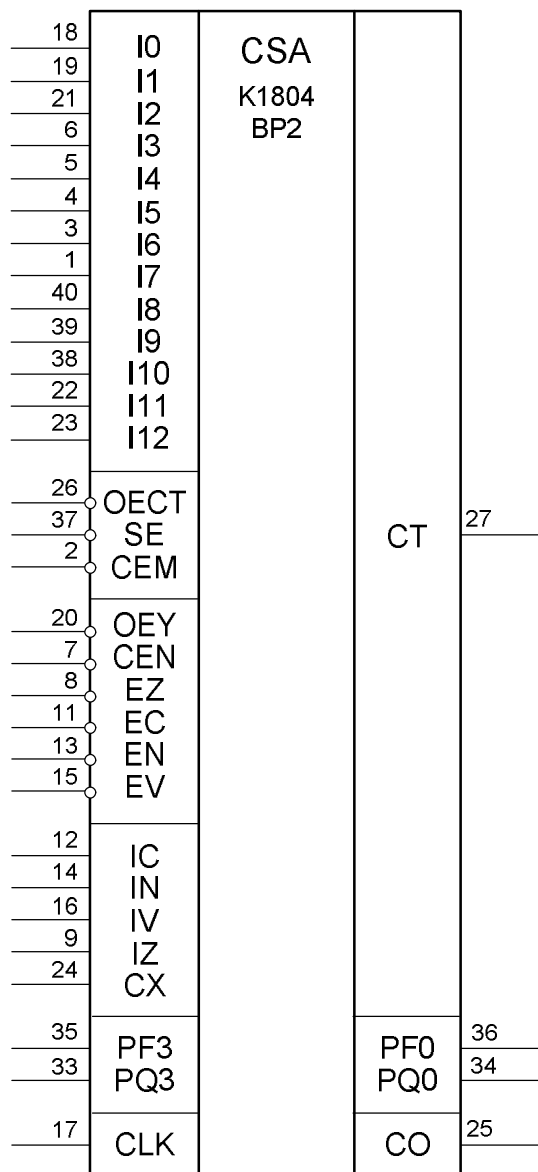


Рис. 2.9. Умовне графічне позначення ІС К1804ВР2

Функціональні блоки, що входять до складу СУСЗ (рис. 2.6) реалізують наступні функції:

- виконання операцій із вмістом регістрів станів (операцій з ознаками);
- формування сигналу вхідного переносу CI для ПЕ і СУП;
- організація арифметичних, логічних і циклічних зсувів слів різної довжини на підставі ознак Z , C , N , V результату виконання мікрооперації у АЛБ – реалізує 32 типи зсувів;
- видача одного з шістнадцяти сигналів логічної умови на вихід CT , який надходить на БМУ і використовується для організації розгалуження у мікропрограмах.

Розглянемо детально склад та призначення функціональних блоків СУСЗ.

2.3.2. Схема управління

Схема управління (рис.2.6) під дією сигналів мікрокоманди $I_{12} - I_0$ формує внутрішні сигнали, що управляють функціональними блоками СУСЗ.

Для управління СУСЗ в структурі мікрокоманди (рис.2.8) відведено 21 розряд, 13 з яких займає поле інформаційного слова СУСЗ_IC, що задає операцію виконувану в СУСЗ у поточному такті (рис. 2.10).

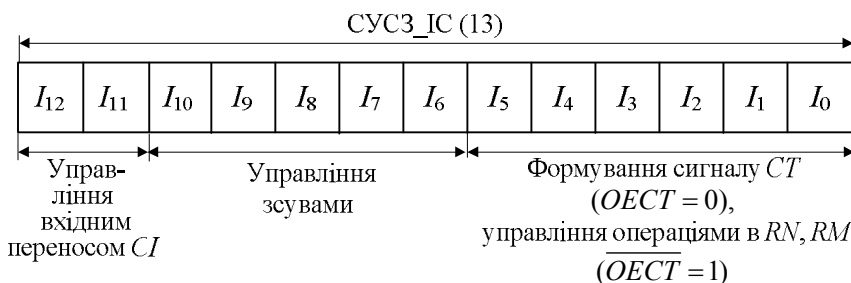


Рис. 2.10. Структура інформаційного поля СУСЗ_IC мікрокоманди

2.3.3. Блок обробки ознак

Блок обробки ознак (рис. 2.6) складається з двох чотирирозрядних регістрів стану (PgN , PgM), двох вхідних мультиплексорів – MXN , MXM і одного вихідного мультиплексора – $MX_{вих}$. Даний блок призначений для зберігання і модифікації ознак, які формуються в ПЕ. Під час виконання мікрооперацій у АЛБ формуються ознаки – переносу (C), знаку результату (N), переповнювання (OVR) і

рівність результату нулю (Z), які з виходів C , N , V , Z ПЕ поступають на входи IC , IN , IV , IZ ознак стану СУСЗ (рис. 2.1, рис. 2.6).

Ознаки зберігаються в регістрах стану PгN і PгM СУСЗ (рис. 2.11). Запис інформації в регістри PгN і PгM відбувається за додатним перепадом тактового імпульсу T (перехід від 0 до 1) за наявності сигналу дозволу запису $\overline{CEN} = 0$ або $\overline{CEM} = 0$ відповідно.

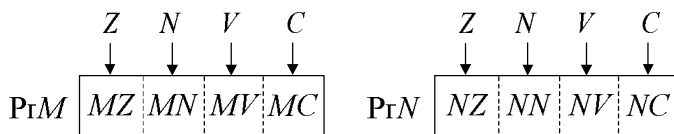


Рис. 2.11. Структура регістрів стану PгN і PгM

2.3.4. Виконання операцій з вмістом регістрів станів

У полі інформаційного слова СУСЗ_IC якому відповідають розряди $I_5 - I_0$ (рис. 2.10) при встановленні управляючого сигналу $\overline{OECT} = 1$ кодуються операції що можуть бути виконані із вмістом регістрів стану (рис. 2.12).

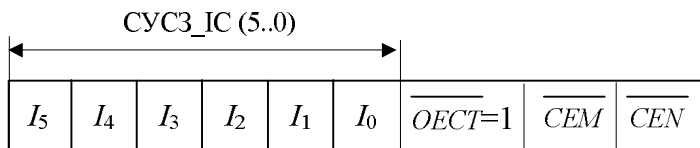


Рис. 2.12. Управління виконанням операцій із вмістом регістрів станів

Операції, що виконуються із вмістом регістрів станів PгN і PгM , можна розділити на наступні групи:

- операції з бітами,
- регістрові операції,
- операції завантаження регістра.

Операції з вмістом регістру стану PгN

В PгN інформація надходить з виходу двоканального мультіплексора MXN (рис. 2.6).

Залежно від сигналів мікрокоманди $I_5 - I_0$ в PгN може бути записане слово стану з входів ознак стану СУСЗ – IC , IN , IV , IZ , або з виходів регістру PгM – MC , MN , MV , MZ . Окрім того, може бути

встановлено в значення 0 або 1, як усе інформаційне слово записане у регістрі RgN , так і кожний окремий розряд цього регістру.

Під час запису інформації у регістр RgN на вході дозволу запису встановлюється значення 0 ($\overline{CEN} = 0$). При $\overline{CEN} = 1$ запис в RgN заборонений.

Операції з бітами відповідають встановленню в значення 0 або 1 одного з розрядів RgN в залежності від сигналів $I_5 - I_0$ закодованих у мікрокоманді (табл. 2.9).

Регістрові операції є операціями з інформаційним словом, записаним в RgN . Управління виконанням операцій здійснюється розрядами I_1 і I_0 мікрокоманди за встановленням нульових значень розрядів мікрокоманди $I_5 - I_2$ (табл. 2.10).

Таблиця 2.9. Операції з бітами регістра стану PrN ($\overline{CEN} = 0$)

I_5	I_4	I_3	I_2	I_1	I_0	Операція	Коментар
0	0	1	0	0	1	$1 \rightarrow NZ$	Встановлення в 1 ознаки Z
0	0	1	0	1	0	$0 \rightarrow NC$	Встановлення в 0 ознаки C
0	0	1	0	1	1	$1 \rightarrow NC$	Встановлення в 1 ознаки C
0	0	1	1	0	0	$0 \rightarrow NN$	Встановлення в 0 ознаки N
0	0	1	1	0	1	$1 \rightarrow NN$	Встановлення в 1 ознаки N
0	0	1	1	1	0	$0 \rightarrow NV$	Встановлення в 0 ознаки OVR
0	0	1	1	1	1	$1 \rightarrow NV$	Встановлення в 1 ознаки OVR

Таблиця 2.10. Регістрові операції з вмістом регістру стану PrN , де $i = Z, C, N, V$ ($\overline{CEN} = 0$)

I_5	I_4	I_3	I_2	I_1	I_0	Операція	Коментар
0	0	0	0	0	0	$M_i \rightarrow N_i$	Запис вмісту регістру RgM у регістр RgN
0	0	0	0	0	1	$1 \rightarrow N_i$	Встановлення в значення 1 всіх розрядів регістру RgN
0	0	0	0	1	0	$M_i \leftrightarrow N_i$	Регістровий обмін між регістрами RgM та RgN
0	0	0	0	1	1	$0 \rightarrow N_i$	Встановлення в значення 0 всіх розрядів регістру PrN

Операції завантаження регістра відповідають запису ознак стану в PrN з входів ознак стану (IC, IN, IV, IZ) (табл. 2.11).

Таблиця 2.11. Операції завантаження регістра стану PrN ($\overline{CEN} = 0$)

I_5	I_4	I_3	I_2	I_1	I_0	Операція	Коментар
0	0	0	1	1	0	$IZ \rightarrow NZ$	Використовується при виконанні ряду арифметичних операцій коли немає необхідності перевіряти ознаку переповнювання після кожної операції, а достатньо встановити, що переповнювання було хоча б під час однієї з операцій
0	0	0	1	1	1	$IC \rightarrow NC$	
						$IN \rightarrow NN$	
						$IV \vee NV \rightarrow NV$	
0	1	1	0	0	*	$IZ \rightarrow NZ$	Завантаження з інверсією ознаки переносу
1	0	1	0	0	*	$\overline{IC} \rightarrow NC$	
1	1	1	0	0	*	$IN \rightarrow NN$	
0	0	0	1	0	*	$IV \rightarrow NV$	
0	1	0	*	*	*	$IZ \rightarrow NZ$	Завантаження безпосередньо з входів ознак стану
0	1	1	0	1	*	$IC \rightarrow NC$	
0	1	1	1	*	*	$IN \rightarrow NN$	
1	0	0	*	*	*	$IV \rightarrow NV$	
1	0	1	0	1	*		
1	0	1	1	*	*		
1	1	0	*	*	*		
1	1	1	0	1	*		
1	1	1	1	*	*		

Операції з вмістом регістру стану PrM

В PrM інформація надходить з виходів мультиплексора MXM (рис. 2.7). Залежно від значення сигналів $I_5 - I_0$ в PrM може бути записана інформація з входів ознак стану $\text{СУСЗ} - IC, IN, IV, IZ$ (ціле слово стану, або порозрядно), з виходів регістру $\text{PrN} - NC, NN, NV, NZ$ або з двоспрямованої шини $Y - YC, YN, YV, YZ$. Окрім того в кожен розряд PrM може бути записане значення 0 або 1. Для виконання запису необхідно, щоб на вході дозволу запису в PrM було встановлене значення $\overline{CEM} = 0$. При $\overline{CEM} = 1$ запис в PrM заборонений.

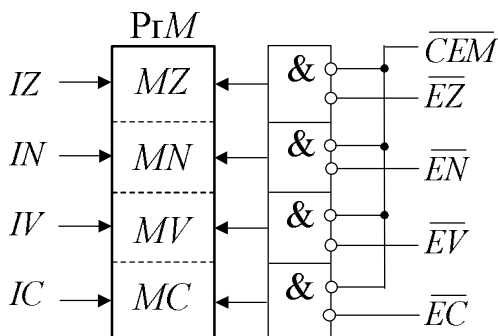


Рис. 2.13. Операції з бітами у регістрі PGM

Операції *запису у біти* регістру PGM ознак результату виконуються з встановленням сигналів дозволу запису $\overline{EZ} = \overline{EN} = \overline{EV} = \overline{EC} = 0$ (рис. 2.13). За встановлення управляючого сигналу $\overline{CEM} = 0$ і наявності сигналу дозволу запису у біти регістру відбувається запис ознаки у відповідний розряд регістру PGM. Якщо сигнал на вході дозволу запису ознаки $\overline{CEM} = 0$, а значення будь-яких сигналів дозволу запису бітів дорівнює 1, то вміст відповідних розрядів регістру PGM не зміниться. Тобто, під час встановлення сигналів $\overline{EZ} \vee \overline{EN} \vee \overline{EV} \vee \overline{EC} = 1$ відбувається заборона запису у відповідний розряд регістру PGM. За значення $\overline{CEM} = 1$ забороняється запис у всі розряди регістру PGM.

Управління виконанням регістрових операцій і операції завантаження, що виконуються у регістрі PGM, наведено в табл. 2.12 – 2.13.

Таблиця 2.12. Регістрові операції регістра PGM

($\overline{EZ} = \overline{EN} = \overline{EV} = \overline{EC} = \overline{CEM} = 0$)

I_5	I_4	I_3	I_2	I_1	I_0	Операція	Коментар
0	0	0	0	0	0	$Y_i \rightarrow M_i$	Запис інформації з шини Y в PGM
0	0	0	0	0	1	$1 \rightarrow M_i$	Встановлення в 1 всіх розрядів PGM
0	0	0	0	1	0	$N_i \leftrightarrow M_i$	Регістровий обмін
0	0	0	0	1	1	$0 \rightarrow M_i$	Встановлення в 0 всіх розрядів PGM
0	0	0	1	0	1	$\overline{M_i} \rightarrow M_i$	Інвертування вмісту регістру PGM

Таблиця 2.13. Операції завантаження регістру RgM

$$(\overline{EZ} = \overline{EN} = \overline{EV} = \overline{EC} = \overline{CEM} = 0)$$

I_5 I_4 I_3 I_2 I_1 I_0	Операція	Коментар
0 0 0 1 0 0	$IZ \rightarrow NZ$ $IV \rightarrow MV$ $IN \rightarrow MN$ $IC \rightarrow MC$	Використовується при організації зсувів з використанням ознаки переповнювання, а не переносу
0 0 1 0 0 *	$IZ \rightarrow MZ$	Завантаження з інверсією ознаки переносу
0 1 1 0 0 *	$\overline{IC} \rightarrow MC$	
1 0 1 0 0 *	$IM \rightarrow MN$	
1 1 1 0 0 *	$IV \rightarrow MV$	
0 0 0 1 1 *	$IZ \rightarrow MZ$	Завантаження безпосередньо з входів ознак стану
0 0 1 0 1 *	$IC \rightarrow MC$	
0 0 1 1 * *	$IN \rightarrow MN$	
0 0 1 1 * *	$IV \rightarrow MV$	
0 1 0 * * *		
0 1 1 0 1 *		
0 1 1 1 * *		
1 0 0 * * *		
1 0 1 0 1 *		
1 0 1 1 * *		
1 1 0 * * *		
1 1 1 * * *		

Приклад 2.2. Розробити мікропрограму для БОД, що реалізує заданий мікроалгоритм (рис. 2.14).

Виконання завдання

У заданому мікроалгоритмі на рис. 2.14 мнемоніка *LOAD* визначає операції завантаження регістрів стану (*RN*, *RM*) або відповідних бітів цих регістрів (*MC*, *MZ* та інших). У табл. 2.14 наведена кодова карта мікропрограми. У структурі закодованих мікрокоманд умовно показані тільки ті поля, що управляють АЛБ та СУСЗ. В табл. 2.15 наведена діаграма вмісту регістру стану *RM* на протязі виконання мікропрограми.

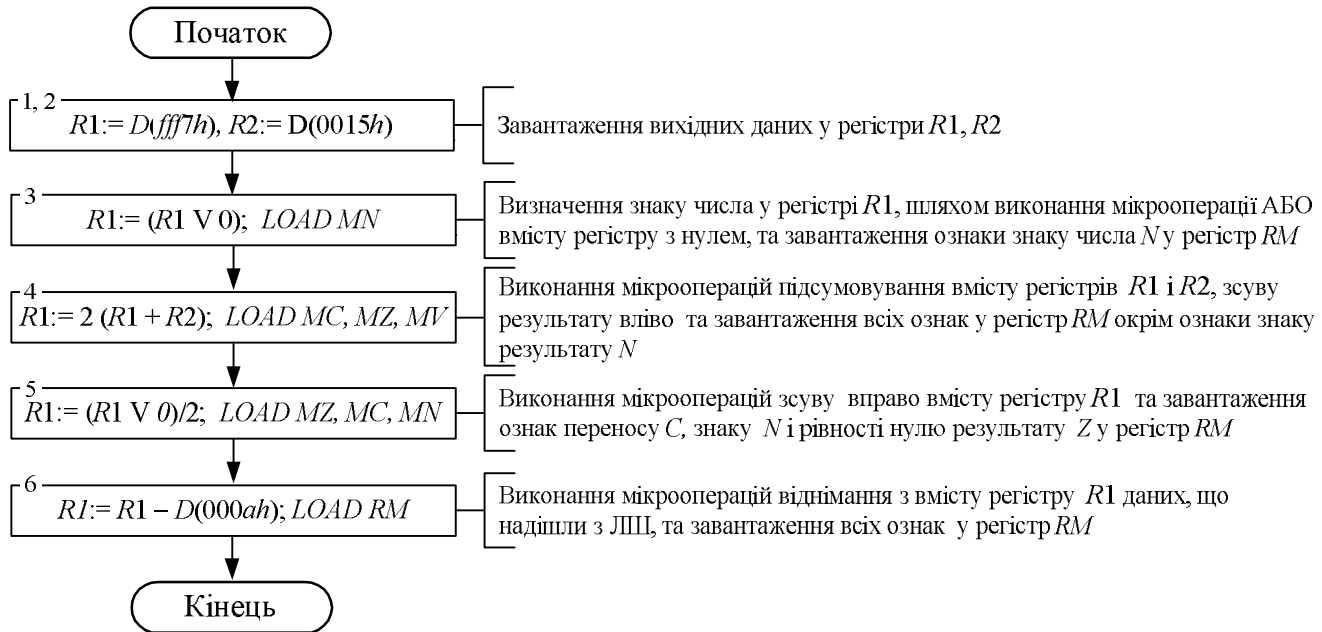


Рис. 2.14. Вихідний мікроалгоритм

Таблиця 2.14. Кодова карта

№ так- ту	Ад- реса [16]	Константа БОД		БОД													
				BC1 (АЛБ)				BP2 (СУС3)									
		D [16]	\overline{OED}	АЛБ_МІ [2]	A [16]	B [16]	\overline{OEY}	СУС3_МІ [2]	\overline{EC}	\overline{EZ}	\overline{EN}	\overline{EV}	\overline{CEN}	\overline{CEM}	\overline{OECT}	\overline{SE}	
1	0001	FFF7	0	011 011 101	1	1	1	00 00000 000000	0	0	0	0	1	1	1	1	
2	0002	0015	0	011 011 101	2	2	1	00 00000 000000	0	0	0	0	1	1	1	1	
3	0003	****	1	011 011 100	1	1	1	00 00000 000110	1	1	0	1	1	0	1	1	
4	0004	****	1	111 000 001	2	1	1	00 10000 000110	0	0	1	0	1	0	1	0	
5	0005	****	1	101 011 100	1	1	1	00 00010 000110	0	0	0	1	1	0	1	0	
6	0006	001A	0	011 001 101	1	1	1	01 00000 000110	0	0	0	0	1	0	1	1	

Таблиця 2.15. Діаграма вмісту регістру стану RM

№ такту	MC	MZ	MN	MV
3	*	*	1	*
4	1	0	*	0
5	0	0	*	0
6	0	0	1	0

Результат налагодження мікропрограми у моделюючій програмі *COMPLEX* представлений на рис. 2.15.

Исходный файл :SUSZ_K.PMK Страница: 1.1

Адр	Конст.		Б О Д												
	БОД	БМУ	В С 1				В Р 2								
			МІ ×	А	В	ОЕУ	МІ ×	С	З	Н	У	ОЕСТ	СЕН	СЕМ	SE
000	FFF?	0	011.011.101	1	1	1	00.00000.000000	0	0	0	0	1	1	1	1
001	0015	0	011.011.101	2	2	1	00.00000.000000	0	0	0	0	1	1	1	1
002	0000	1	011.011.100	1	1	1	00.00000.000110	1	1	0	1	1	1	0	1
003	0000	1	111.000.001	2	1	1	00.10000.000110	0	0	1	0	1	1	0	1
004	0000	1	101.011.100	1	1	1	00.00010.000110	0	0	0	1	1	1	0	1
005	001A	0	011.001.101	1	1	1	01.00000.000110	0	0	0	0	1	1	0	1
006	0000	1	001.000.000	0	0	1	00.00000.000000	0	0	0	0	1	1	1	1

Исходный файл :SUSZ_K.PMK Страница: 1.2

Адр	БМУ и БПП												ОП, ВУ, РА								
	РАА, РАВ				ВУ4					М	ВН1		ПАВ	ВУ				ОП		РгA	
	MSA	MSB	EWA	EWB	МІ ×	CCE	COM	CI	RLD		МІ ×	EINS		EV	I	O	LCK	IA	R	W	EWB
000	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
001	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	
002	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	
003	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	
004	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	
005	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	
006	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	

Нажмите любую клавишу для окончания

Исходный файл :SUSZ.ASM Страница: 1

```

accept r1:0fff7h
accept r2:0015h
{or r1,r1,z;load rm,flags;cem_c;cem_z;cem_v;}
{add sll,r1,r1,r2,z;load rm,flags;cem_n;}
{or srl,r1,r1,z;load rm,flags;cem_v;}
{sub r1,r1,001ah,nz;load rm,flags;}
{}
```

Рис. 2.15. Мікропрограма управління станами

Вивід вмісту регістрів PgN , PgM на двоспрямовану шину Y

Інформація з виходів PgN , PgM або з входів ознак стану СУСЗ – IC , IN , IV , IZ , через вихідний мультиплексор із трьохстабільними виходами $MX_{\text{вих}}$ (рис. 2.6) надходить на двоспрямовану шину Y . Якщо на входах $I_5 - I_0$ встановлені нульові значення, то шина Y є вхідною незалежно від сигналу дозволу виводу інформації (\overline{OEY}). В інших випадках шина Y є вихідною. Управління виводом інформації через шину Y за допомогою сигналів \overline{OEY} , I_5 та I_4 надане в табл. 2.16.

Таблиця 2.16. Управління виводом інформації через шину Y ($i = Z, C, N, V$)

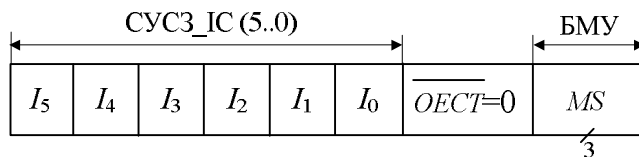
\overline{OEY}	I_5	I_4	Y	Коментар
1	*	*	Z	Стан високого опору
0	0	*	$N_i \rightarrow Y_i$	Вивід вмісту PgN
0	1	0	$M_i \rightarrow Y_i$	Вивід вмісту PgM
0	1	1	$I_i \rightarrow Y_i$	Вивід інформації з входів ознак стану
*	$(I_5..I_0) = 000000$ $Y_i \rightarrow PgM$			При значенні 0 на входах $I_5 - I_0$ шина Y є вхідною незалежно від значення сигналу \overline{OEY}

2.3.5. Блок перевірки умови

Блок перевірки умови (рис. 2.6) складається з комбінаційної схеми перевірки умови та мультиплексора зі схемою управління полярністю.

Блок перевірки умови призначений для формування вихідного сигналу коду умови CT , яка надходить на один з входів мультиплексора умов MU (рис. 2.7) БМУ і використовується для організації різноманітних переходів у мікропрограмах.

Управління формуванням умови здійснюється розрядами $I_5 - I_0$ поля СУСЗ_ІС МК (рис. 2.10) за встановлення сигналу $\overline{OECT} = 0$ (рис. 2.16).

Рис. 2.16. Управління формуванням сигналу CT

Блок перевірки умови виконує одну з 16 можливих операцій (табл. 2.17) формування умови, результат якої видається на вихід коду умови CT . Вибір операндів для виконання операцій в даному блоці управляється сигналами мікрокоманди I_4, I_5 .

Найбільш поширеними операціями, що виконуються блоком перевірки умови є передавання однієї з ознак стану на вихід CT (чотири операції). Ще чотири операції є більш складними і застосовуються під час виконання в АЛБ операції віднімання ($A - B$) для забезпечення умов $A = B$, $A \neq B$, $A \geq B$ та інших. (табл. 2.18). При цьому аргументи A і B мають бути подані в доповнювальному коді або як числа без знаку.

Далі результат однієї з виконаної у блоці операції формування умови через мультиплексор передається на вхід **схеми управління полярністю**, яка за необхідності інвертує отриманий результат. Таким чином решта вісім операцій є інверсією вже розглянутих операцій. Результат перевірки умови з виходу схеми управління полярністю CO надходить на трьохстабільну шину CT , що управляється сигналом дозволу коду умови \overline{OECT} .

Для підключення виходу CT СУСЗ до одного з входів мультиплексора МУ БМУ в трирозрядному полі MS мікрокоманди, що належить до полів управління БМУ необхідно встановити номер входу мультиплексора (рис. 2.16).

При $\overline{OECT} = 0$ дозволяється вивід коду умови через шину CT . Якщо $\overline{OECT} = 1$, то вивід коду умови заборонений, а шина CT знаходиться в стані високого опору.

Таблиця 2.17. Управління виходом коду умови CT ($\overline{OECT} = 0$)

$I_3 \ I_2 \ I_1 \ I_0$	$I_5 = I_4 = 0$	$I_5 = 0, I_4 = 1$	$I_5 = 1, I_4 = 0$	$I_5 = I_4 = 1$
0 0 0 0	$(NN \oplus NV) \vee NZ$	$(NN \oplus NV) \vee NZ$	$(MN \oplus MV) \vee MZ$	$(IN \oplus IV) \vee IZ$
0 0 0 1	$\overline{(NN \oplus NV)} \& \overline{NZ}$	$\overline{(NN \oplus NV)} \& \overline{NZ}$	$\overline{(MN \oplus MV)} \& \overline{MZ}$	$\overline{(IN \oplus IV)} \& \overline{IZ}$
0 0 1 0	$NN \oplus NV$	$NN \oplus NV$	$MN \oplus MV$	$IN \oplus IV$
0 0 1 1	$\overline{NN \oplus NV}$	$\overline{NN \oplus NV}$	$\overline{MN \oplus MV}$	$\overline{IN \oplus IV}$
0 1 0 0	NZ	NZ	MZ	IZ
0 1 0 1	\overline{NZ}	\overline{NZ}	\overline{MZ}	\overline{IZ}
0 1 1 0	NV	NV	MV	IV
0 1 1 1	\overline{NV}	\overline{NV}	\overline{MV}	\overline{IV}
1 0 0 0	$NC \vee NZ$	$NC \vee NZ$	$MC \vee MZ$	$IC \vee IZ$
1 0 0 1	$\overline{NC} \& \overline{NZ}$	$\overline{NC} \& \overline{NZ}$	$\overline{MC} \& \overline{MZ}$	$\overline{IC} \& \overline{IZ}$
1 0 1 0	NC	NC	MC	IC
1 0 1 1	\overline{NC}	\overline{NC}	\overline{MC}	\overline{IC}
1 1 0 0	$\overline{NC} \vee NZ$	$\overline{NC} \vee NZ$	$\overline{MC} \vee MZ$	$\overline{IC} \vee IZ$
1 1 0 1	$NC \& \overline{NZ}$	$NC \& \overline{NZ}$	$MC \& \overline{MZ}$	$IC \& \overline{IZ}$
1 1 1 0	$IN \oplus MN$	NN	MN	IN
1 1 1 1	$\overline{IN \oplus MN}$	\overline{NN}	\overline{MN}	\overline{IN}

Таблиця 2.18. Перевірка відношення чисел А і В після виконання операції (А – В)

Відношення	Числа без знаку								
	Стан	$CT = 1$				$CT = 0$			
		I_3	I_2	I_1	I_0	I_3	I_2	I_1	I_0
$A = B$	$Z = 1$	0	1	0	0	0	1	0	1
$A \neq B$	$Z = 0$	0	1	0	1	0	1	0	0
$A \geq B$	$C = 1$	1	0	1	0	1	0	1	1
$A < B$	$C = 0$	1	0	1	1	1	0	1	0
$A > B$	$C \& Z = 1$	1	1	0	1	1	1	0	0
$A \leq B$	$C \vee Z = 1$	1	1	0	0	1	1	0	1

Відношення	Числа в доповню вальному коді								
	Стан	$CT = 1$				$CT = 0$			
		I_3	I_2	I_1	I_0	I_3	I_2	I_1	I_0
$A = B$	$Z = 1$	0	1	0	0	0	1	0	1
$A \neq B$	$Z = 0$	0	1	0	1	0	1	0	0
$A \geq B$	$\overline{N \oplus V} = 1$	0	0	1	1	0	0	1	0
$A < B$	$N \oplus V = 1$	0	0	1	0	0	0	1	1
$A > B$	$[\overline{N \oplus V}] \& \overline{Z} = 1$	0	0	0	1	0	0	0	0
$A \leq B$	$[N \oplus V] \& Z = 1$	0	0	0	0	0	0	0	1

2.3.6. Блок управління переносами

Блок управління переносами (рис. 2.6) формує сигнал вихідного переносу CO , який використовується для формування вхідного переносу CI АЛБ процесорного елементу. Управління формуванням вихідного переносу здійснюється сигналами $I_{12}, I_{11}, I_5, I_3, I_2, I_1$ поля СУЗС_IC мікрокоманди як показано в табл. 2.19.

Розряди I_{12}, I_{11} використовуються для присвоєння вхідному переносу значення 0 або 1. Якщо $I_{12}I_{11} = 00$, то $CI = 0$, та якщо $I_{12}I_{11} = 01$, $CI = 1$.

Якщо для формування вихідного переносу окрім розрядів I_{12}, I_{11} (коли $I_{12}I_{11} = 11$) використовують також розряди I_5, I_3, I_2, I_1 поля СУЗС_IC МК то вхідному переносу CI можна при-

своїти значення розрядів переносу MC , NC регістрів R_6M та R_6N , або їх заперечення.

Таблиця 2.19. Управління формуванням сигналу вхідного переносу CI АЛБ

I_{12}	I_{11}	I_5	I_3	I_2	I_1	CO	Коментар
0	0	*	*	*	*	0	
0	1	*	*	*	*	1	
1	0	*	*	*	*	CX	
1	1	0	*	*	*	NC	$I_3I_2I_1 \neq 100$
		1	*	*	*	MC	$I_3I_2I_1 \neq 100$
		0	1	0	0	\overline{NC}	
		1	1	0	0	\overline{MC}	

Таким чином, в якості вхідного переносу CI використовується одне з наступних джерел – 0, 1, CX , NC , MC , \overline{NC} , \overline{MC} , що дозволяє легко реалізувати операції додавання та віднімання чисел як звичайної так і подвійної довжини. Вхід CX (коли $I_{12}I_{11} = 10$) служить для організації виконання ПЕ K1804BC2 ряду спеціальних функцій при поєднанні входу CX з виходом Z ПЕ.

2.3.7. Блок управління зсувами

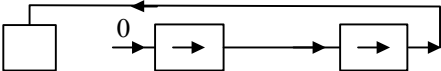
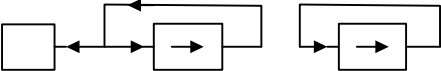
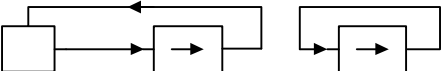
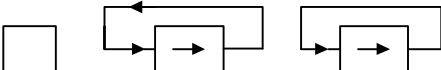
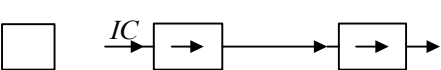
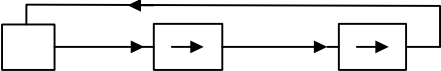
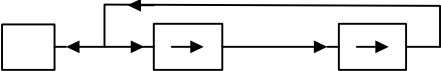
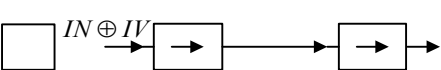
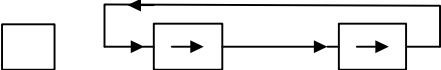
Блок управління зсувами (рис. 2.6) призначений для організації різних варіантів арифметичних, логічних та циклічних зсувів. Для управління завданням типу зсувів використовуються сигнали $I_{10} - I_6$ поля СУСЗ_ІС мікрокоманди. Загалом можливі 32 типи зсувів – 16 вліво і 16 вправо (табл. 2.20).

Виводи зсуву $PF0$, $PF3$, $PQ0$, $PQ3$ є трьохстабільними і управляються сигналом \overline{SE} . Під час встановлення $\overline{SE} = 0$ зсув дозволений, інакше $\overline{SE} = 1$ зсув заборонений, а всі виводи зсуву знаходяться в стані високого опору.

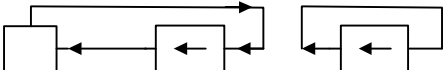
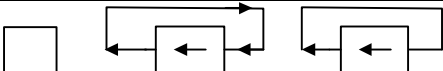
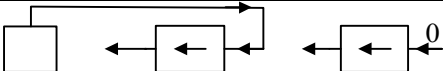
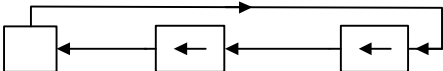
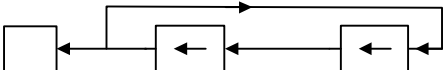
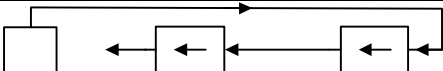
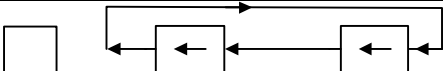
Таблиця 2.20. Управління зсувами ($\overline{SE} = 0$)

Розряди МК					Модифікація зсуву			Сигнали на виводах BP2				
1					2			3				
I_{10}	I_9	I_8	I_7	I_6	MC	$3CB$	$3CQ$	$PF0$	$PF3$	$PQ0$	$PQ3$	MC
0	0	0	0	0		$\xrightarrow{0}$	$\xrightarrow{0}$	Z	0	Z	0	–
0	0	0	0	1		$\xrightarrow{1}$	$\xrightarrow{1}$	Z	1	Z	1	–
0	0	0	1	0		$\xrightarrow{0}$	\xrightarrow{MN}	Z	0	Z	MN	PF0
0	0	0	1	1		$\xrightarrow{1}$		Z	1	Z	PF0	–
0	0	1	0	0				Z	MC	Z	PF0	–
0	0	1	0	1		\xrightarrow{MN}		Z	MN	Z	PF0	–
0	0	1	1	0		$\xrightarrow{0}$		Z	0	Z	PF0	–

Продовження таблиці 2.16.

1					2	3				
0	0	1	1	1		Z	0	Z	PF0	PQ0
0	1	0	0	0		Z	PF0	Z	PQ0	PF0
0	1	0	0	1		Z	MC	Z	PQ0	PF0
0	1	0	1	0		Z	PF0	Z	PQ0	—
0	1	0	1	1		Z	IC	Z	PF0	—
0	1	1	0	0		Z	MC	Z	PF0	PQ0
0	1	1	0	1		Z	PQ0	Z	PF0	PQ0
0	1	1	1	0		Z	$IN \oplus IV$	Z	PF0	—
0	1	1	1	1		Z	PQ0	Z	PF0	—

Продовження таблиці 2.16.

1	1	0	0	1		MC	Z	PQ3	Z	PF3
1	1	0	1	0		PF3	Z	PQ3	Z	–
1	1	0	1	1		MC	Z	0	Z	–
1	1	1	0	0		PQ3	Z	MC	Z	PF3
1	1	1	0	1		PQ3	Z	PF3	Z	PF3
1	1	1	1	0		PQ3	Z	MC	Z	–
1	1	1	1	1		PQ3		PF3		–

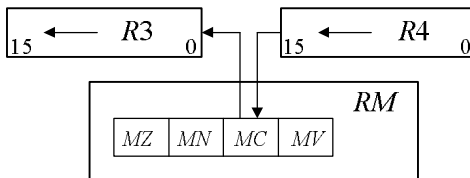
Приклад 2.3. Розробити фрагмент мікропрограми для зсуву вліво слова подвійної довжини. Старша і молодша частини слова записані у регістрах $R3$, $R4$ відповідно.

Виконання завдання:

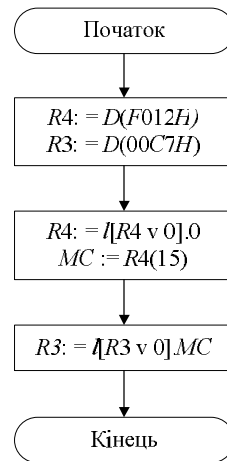
Операційна схема та мікроалгоритм виконання операції наведена на рис. 2.17, а, б. Для реалізації зсуву слова подвійної довжини виконаємо два типи зсувів: по-перше – логічний зсув $SL.16$ ($I_{10}..I_6 = 10000$) молодшої частини слова $R4$, під час чого старший розряд, що виходить за межі розрядної сітки, зберігається у розряді MC регістру стану RM . Далі виконаємо циклічний зсув $SL.25$ ($I_{10}..I_6 = 11001$) старшої частини слова $R3$, під час чого у молодший розряд, звільнений за зсуву, записується вміст розряду MC регістру RM , в якому в подальшому зберігається старший розряд слова, що вийшов за межі розрядної сітки.

Цифрова діаграма стану регістрів наведена у табл. 2.21. Кодова карта розробленої мікропрограми наведена у табл. 2.22.

Мікропрограма зсуву слів подвійної довжини, налагоджена за допомогою середи модулювання *COMPLEX*, наведена на рис. 2.18.



а



б

Рис. 2.17. Реалізація зсуву вліво слова подвійної довжини: а – операційна схема; б – мікроалгоритм.

Таблиця 2.21. Цифрова діаграма стану регістрів

№ так-ту	Вихідний стан			Інформація у приймачі результату, [16]	МС	Мікрооперація
	Шина D, [16]	R3, [16]	R4, [16]			
1	00C7	0000	0000	$R3 := 00C7$	*	$R3 := D(007C) \vee 0$
2	F012	00C7	0000	$R4 := F012$	*	$R4 := D(F012) \vee 0$
3	****	00C7	F012	$R4 := E024$	1	$R4 := SLL(R4 \vee 0)$
4	****	0070	E024	$R3 := 018F$	1	$R3 := S.25(R3 \vee 0)$
		018F	E024			Результат

Примітка: SLL, S.25 – мнемонічне позначення зсувів логічного вправо та циклічного відповідно.

Таблиця 2.22 Кодова карта мікропрограми зсуву слів подвійної довжини

№ так- ту	Адреса [16]	Константа БОД		БОД													
				BC1 (АЛБ)				BP2 (СУСЗ)									
		D [16]	OED	АЛБ MI [2]	A [16]	B [16]	OЕY	СУСЗ MI [2]	EC	EZ	EN	EV	CEN	CEM	OECT	SE	
1	0000	00C7	0	011 011 101	3	3	1	00 00000 000000	0	0	0	0	1	1	1	1	
2	0001	F012	0	011 011 101	4	4	1	00 00000 000000	0	0	0	0	1	1	1	1	
3	0002	****	1	111 011 100	4	4	1	00 10000 000000	0	0	0	0	1	1	1	0	
4	0003	****	1	111 011 100	3	3	1	00 11001 000000	0	0	0	0	1	1	1	0	

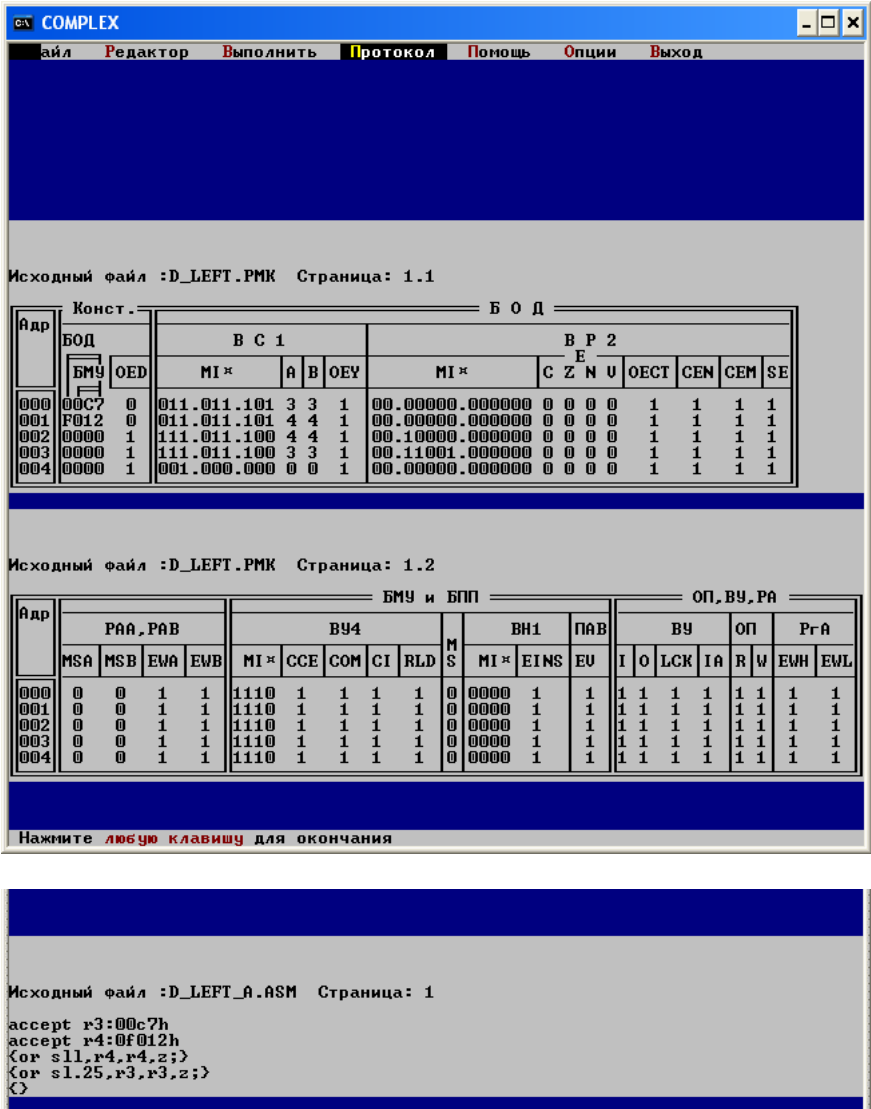


Рис. 2.18.Мікропрограма зсуву вліво слова подвійної довжини

Приклад 2.4. Розробити фрагмент мікропрограми для додавання слів подвійної довжини. Доданки записані у регістрах $R1$, $R2$ та $R3$, $R4$ відповідно.

Виконання завдання:

Операційна схема та мікроалгоритм виконання операції наведена на рис. 2.19, а, б.

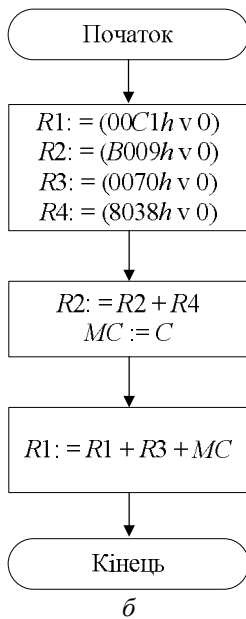
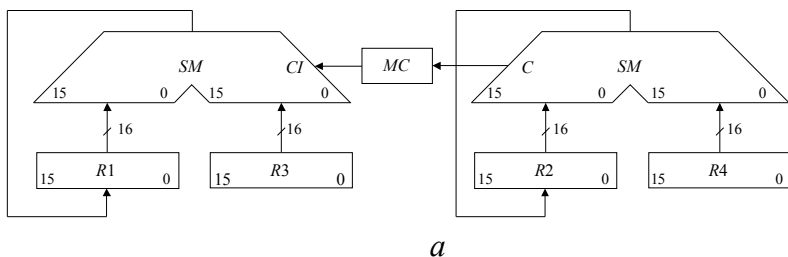


Рис. 2.19. Операція додавання слів подвійної довжини: а – операційна схема; б – мікроалгоритм

Для реалізації додавання слів подвійної довжини виконаємо послідовне додавання молодших і старших частин слова із поширенням переносу із молодшої частини результату у старшу. Для реалізації поширення переносу під час додавання молодших частин слів збережемо ознаку переносу у розряді *MC* регістру стану *RM*:

$$R2 := R2 + R4 ; \text{LOAD } MC,$$

під час додавання старших частин, подамо значення переносу на вхід *CI* АЛБ:

$$R1 := R1 + R3 + CI, \text{ де } CI := MC.$$

Цифрова діаграма стану регістрів наведена у табл. 2.23. Кодова карта розробленої мікропрограми наведена у табл. 2.24.

Мікропрограма додавання слів подвійної довжини налагоджена за допомогою середи модулювання *COMPLEX* наведена на рис. 2.20.

2.3.8. Виконання операції нормалізації

ІС K1804BP2 забезпечує виконання операції нормалізації чисел звичайної і подвійної довжини, як для ПЕ K1804BC1, так і для ПЕ K1804BC2.

Поєднання виводів мікросхем K1804BP2, K1804BC1 і K1804BY4 показано на рис. 2.7.

В мікросхемі K1804BC1 однією з ознак закінчення нормалізації під час виконання чергової мікрооперації служить встановлення сигналу на виході *CO* – старшої секції результату (під час реалізації обчислень у МПС із словами довжиною більш ніж чотири розряди ПЕ поєднуються у ланцюги, див розділ 2.4 рис. 2.22). При цьому на виході *CO* формується сигнал, як результат виконання операції ВИКЛЮЧНЕ АБО над двома старшими розрядами.

Другою ознакою, що інформує про закінчення нормалізації служить сигнал на виході *OVR*, що є результатом операції ВИКЛЮЧНЕ АБО над двома середніми розрядами старшої секції результату. Ця ознака встановлюється за один такт до закінчення операції. Сигнали *CO* або *OVR* через блок перевірки умови СУСЗ виставляються на виході *CT* K1804BC2 та надходять на вхід умови *CC* БМУ ІС K1804BY4 (див. розділ 2.5, рис. 2.27).

Таблиця 2.23. Цифрова діаграма стану регістрів

№ так- ту	Вихідний стан					Інформація у приймачі результату, [16]	МС	Мікрооперація
	Шина <i>D</i> , [16]	<i>R1</i> , [16]	<i>R2</i> , [16]	<i>R3</i> , [16]	<i>R4</i> , [16]			
1	00C1	0000	0000	0000	0000	$R1 := 00C1$	*	$R1 := D(00C1) \vee 0$
2	0009	00C1	0000	0000	0000	$R2 := 0009$	*	$R2 := D(0009) \vee 0$
3	0070	00C1	0009	0000	0000	$R3 := 0070$	*	$R3 := D(0070) \vee 0$
4	8038	00C1	0009	0070	0000	$R4 := 8038$	*	$R4 := D(8038) \vee 0$
4	****	00C1	0009	0070	8038	$R2 := 3041$	1	$R2 := R2 + R4 + CI$
5	****	00C1	3041	0070	8038	$R1 := 00CB$	*	$R1 := R3 + R1 + 1(MC)$
		00CB	3041					Результат

Таблиця 2.24 Кодова карта мікропрограми додавання слів подвійної довжини

№ так- ту	Адреса [16]	Константа БОД		БОД													
				BC1 (АЛБ)				BP2 (СУСЗ)									
		D [16]	\overline{OED}	АЛБ_MI [2]	A [16]	B [16]	\overline{OEY}	СУСЗ_MI [2]	\overline{EC}	\overline{EZ}	\overline{EN}	\overline{EV}	\overline{CEN}	\overline{CEM}	\overline{OECT}	\overline{SE}	
1	0000	00C1	0	011 011 101	1	1	1	00 00000 000000	0	0	0	0	1	1	1	1	
2	0001	0009	0	011 011 101	2	2	1	00 00000 000000	0	0	0	0	1	1	1	1	
3	0002	0070	0	011 011 101	3	3	1	00 00000 000000	0	0	0	0	1	1	1	1	
4	0003	8038	0	011 011 101	4	4	1	00 00000 000000	0	0	0	0	1	1	1	1	
5	0004	****	1	011 000 001	4	2	1	00 00000 000110	0	0	0	0	1	0	1	1	
6	0005	****	1	011 000 001	3	1	1	11 00000 100000	0	0	0	0	1	1	1	1	

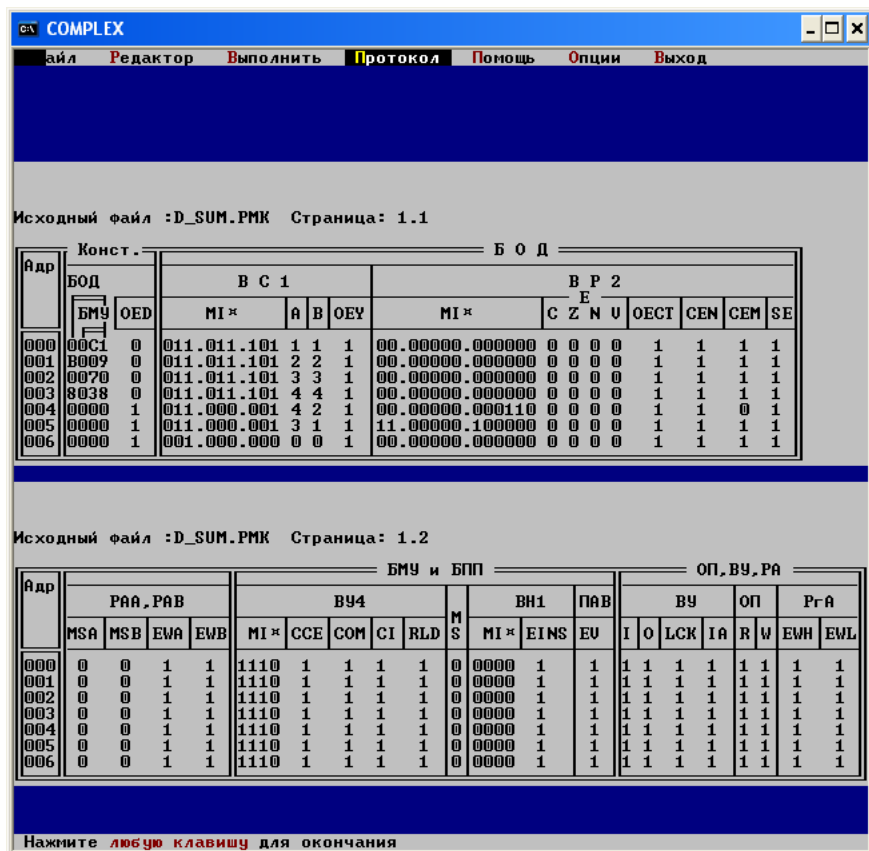


Рис. 2.20. Мікропрограма додавання слів подвійної довжини

Оскільки розглянуті ознаки записуються в регістр стану з запізненням на один такт, то при використанні в якості ознаки закінчення нормалізації сигналу сформованого на виході $C4$ необхідно останнім кроком нормалізації виконати зсув вбік молодших розрядів. Такий зсув виконується СУСЗ спеціальною операцією зсуву за надходження на входи $I_{10} - I_6$ відповідних кодів: $I_{10}, I_9, I_8, I_7, I_6 = 01001$ – нормалізація чисел подвійної довжини або $I_{10}, I_9, I_8, I_7, I_6 = 00010$ – нормалізація чисел звичайної довжини.

В процесорному елементі K1804BC1 не передбачена спеціальна мікрооперація нормалізації. Нормалізація виконується з застосуванням мікрооперації зсуву. Для виконання функції елемента ВИКЛЮЧЕНЕ АБО, який відсутній в мікросхемі K1804BC1, в СУСЗ передбачена спеціальна операція ($MN \oplus IN$), що виконується із знаковими розрядами числа сформованими у попередньому такті обчислень (MN) і у поточному такті (IN).

2.3.9. Реалізація переривань

Мікросхема K1804BP2 може використовуватися для організації переривань як на програмному, так і на мікропрограмному рівні. При виконанні переривань відбувається передавання ознак стану з регістру RgM (під час переривань програмного рівня) або з регістрів RgN , RgM (під час переривань мікропрограмного рівня) в зовнішні регістри. Таке передавання ознак стану реалізується через двоспрямовану шину Y (див. табл. 2.16).

Після закінчення переривання в RgM і RgN повинен бути відновлений стан, що передував перериванню. Вміст RgN , що передував перериванню, спочатку записується в RgM , а далі за запису в RgM ознак стану, що передували перериванню, вміст RgM передається в RgN .

Організація системи переривань в мікропроцесорних системах розглянута у розділі 2.7

2.4. Блоки обробки даних

Блоки обробки даних будуються на основі процесорних елементів, наприклад, ІС К1804ВС1. Нарощування довжини розрядної сітки процесора забезпечується поєднанням декількох ПЕ, як показано на рис. 2.21. для випадку, коли $n = 16$. Шини AA , AB , I і CLK , які умовно не зображені, підключаються паралельно до кожної ІС ВС1, аналогічно шині OE . З ціллю зменшення часу розповсюдження переносу використовують ІС К1804ВР1, умовне графічне позначення якої показано на рис. 2.22. В цьому випадку (коли $n = 16$) ланцюги розповсюдження переносу реалізують, відповідно схемі зображених на рис. 2.23.

Мікросхеми ВС1 мають відповідні виходи P і G , що дозволяє при побудові процесорів з довжиною слова $n > 16$ здійснювати каскадне підключення цих мікросхем (рис. 2.23).

Часові параметри ІС К1804ВР1 приведені в табл. 2.25.

Таблиця 2.25. Параметри ІС К1804ВР1.

Час затримки сигналів	Значення, нс
від CI до CO	10
від $P0 - P3$ до P	8,5
від $P0 - P3$, $G0 - G3$ до CX , CY , CZ	7
від $P0 - P3$, $G0 - G3$ до G	9

Виходи ознак і зсувачів підключають до однойменних входів ІС К1804ВР2 (рис. 2.9). Відповідні виходи цієї ІС забезпечують подачу на блок управління різних ознак і вхідного переносу для молодшої секції процесору.

Для управління роботою шістнадцятирозрядного операційного блоку у структурі мікрокоманди задіяні наступні поля (рис. 2.8):

що управляють ІС К1804ВС1 – $АЛБ_MI$, AB , AA , \overline{OE} ;

що управляють ІС К1804ВР2 – $СУСЗ_MI$, \overline{OECT} , \overline{EZ} , \overline{EN} , \overline{EC} , \overline{EV} , \overline{CEN} , \overline{CEM} , \overline{OEY} , \overline{SE} .

Дані приймаються в операційний блок по шині D , а видаються по шині Y ІС К1804ВС1. Двоспрямована шина Y ІС К1804ВР2 дозволяє, наприклад, зберегти в стеку і завантажити ознаки при роботі з підпрограмами.

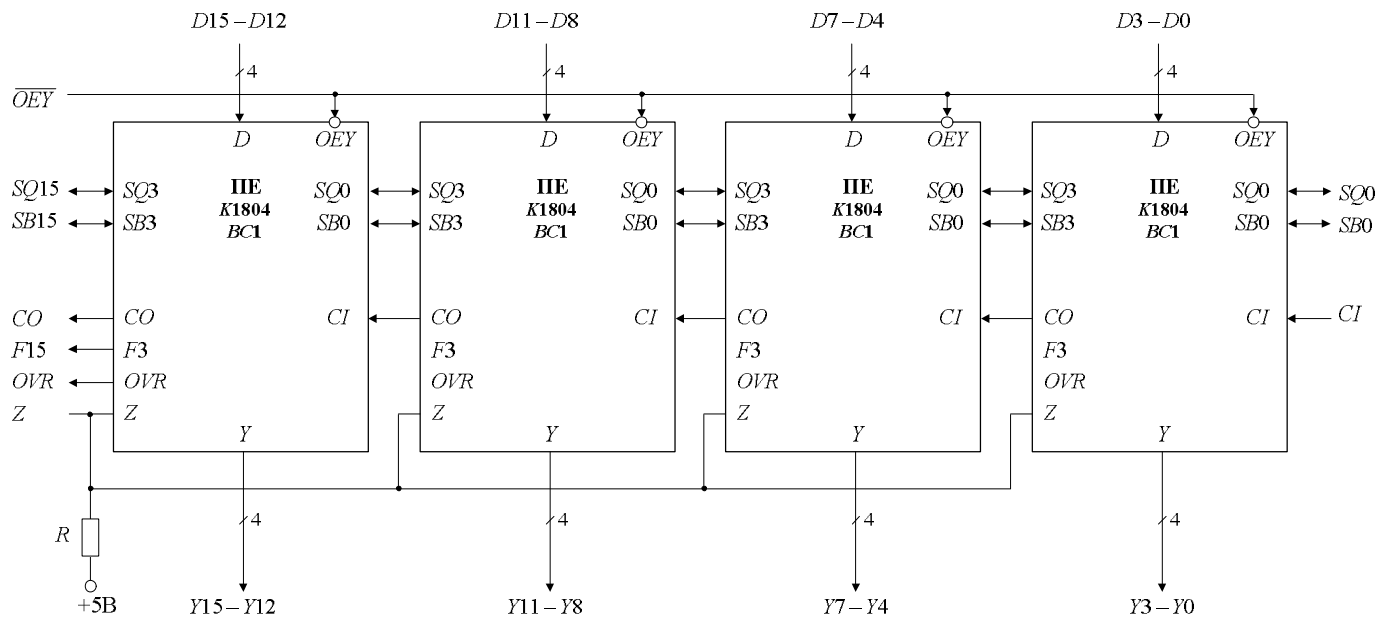


Рис 2.21. Структурна схема шістнадцятирозрядного ПЕ

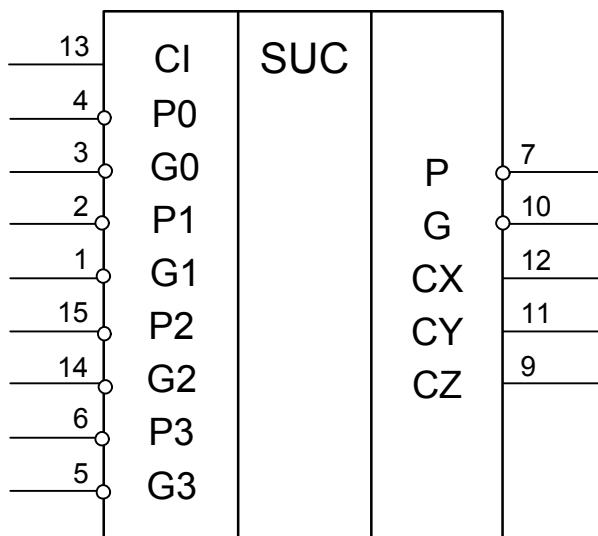


Рис. 2.22 Умовне графічне позначення ІС К1804ВР1

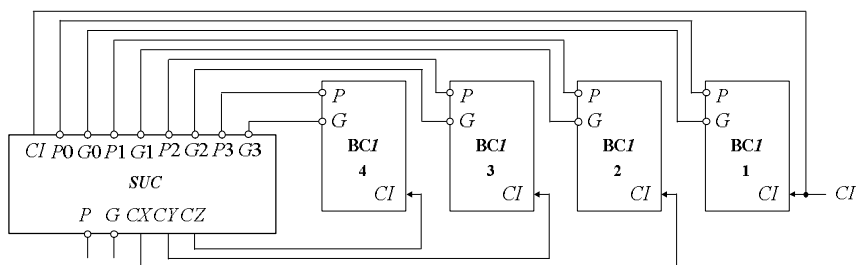


Рис. 2.23. Організація ланцюгів розповсюдження переносу шістнадцяти-розрядного ПЕ

Приклад 2.5. Розробити мікропрограму обчислення заданої функції для шістнадцятирозрядного блоку обробки даних.

$$F = 2X3 \& (X2 - X1) / 2$$

Вихідні дані: $X1 = -3$, $X2 = 15$, $X3 = 7$.

Вважатимемо, що для подання операндів використовується доповнювальний код. Мікроалгоритм обчислення функції F показаний

на рис. 2.24. В коментарях до кожної операторної вершини наведені значення управляючих сигналів IC , AA , BB , CI , $SB15$, $SB0$, $SQ15$, $SQ0$, що забезпечують виконання відповідної мікрооперації.

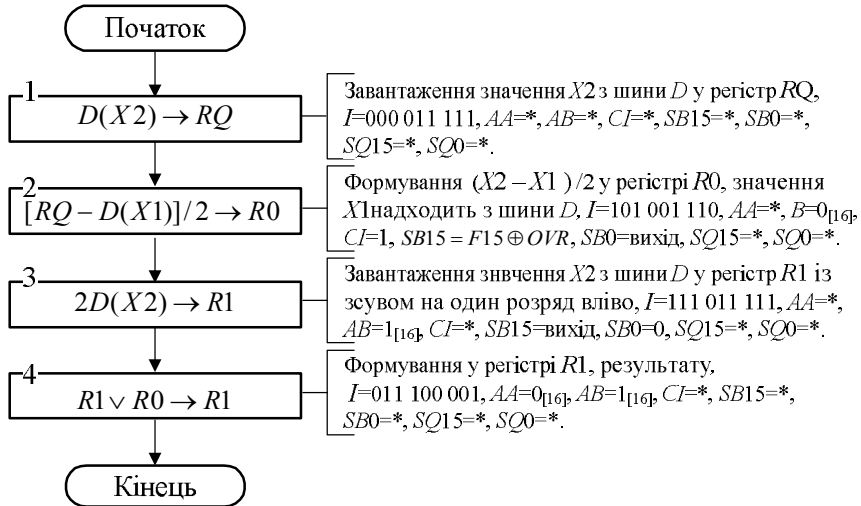


Рис. 2.24. Мікроалгоритм обчислення функції

Послідовність мікрокоманд, що реалізують заданий мікроалгоритм, відображена у табл. 2.25.

Розглянемо мікрооперацію $[RQ - D(X1)]/2 \rightarrow R0$, що виконується в другому такті. Джерелами даних, що надходять на входи S і R АЛБ (рис. 2.1), є регістр RQ і дані на вхідній шині D . Відповідно до табл. 2.4. визначаємо розряди $I_2 I_1 I_0$ поля АЛБ_МІ мікрокоманди $I_2 I_1 I_0 = 110$. Мікрооперація віднімання, що здійснюється у цьому такті, визначається розрядами $I_5 I_4 I_3 = 001$ поля АЛБ_МІ мікрокоманди (табл. 2.3). при цьому встановлюється $CI=1$, що відповідає розрядам $I_{12} I_{11} = 01$ поля СУСЗ_МІ мікрокоманди (табл. 2.19). Результат, отриманий на виході Y АЛБ записується в регістр НОЗП за адресою, виставленою на шині AB . За умовою завдання результат має бути записаний у регістр $R0$, тому у полі B АЛБ_МІ мікрокоманди (табл. 2.3) розміщується адреса відповідного регістру у шістнадцятирічному поданні $B = 0$. При цьому кодуємо розряди $I_8 I_7 I_6$ поля АЛБ_МІ мікрокоманди (табл. 2.2), враховуючи те, що результат

мікроопервції має бути зсунутий на один розряд вправо (що відповідає діленню на два за умовою завдання), отримуємо $I_8 I_7 I_6 = 101$.

При арифметичному зсуві вправо розряд, що звільнився, заповнюється знаком числа. Оскільки зсув відбувається на зсувачі ЗСВ, то з урахуванням можливого переповнювання розрядної сітки необхідно забезпечити $SB15 = F15 \oplus OVR$ (де, $F15$ – знак результату, OVR – ознака переповнювання).

Відповідну комутацію виводів зсувача та формування сигналу переносу CO забезпечує ІС ВР2. Подача нуля на вхід переносу ($CI = 0$) забезпечується встановленням розрядів поля СУСЗ_МС мікрокоманди $I_{12} I_{11} = 00$, подача одиниці ($CI = 1$) – встановленням $I_{12} I_{11} = 01$ поля СУСЗ_МІ мікрокоманди.

Якщо зсув не виконується, виводи ВР2, пов'язані із зсувачем, відключаються сигналом $\overline{SE} = 1$, що відповідає розрядам $I_{12} I_{11} = 01$ поля СУСЗ_МІ мікрокоманди (табл. 2.19). Під час виконання зсувів встановлюється $\overline{SE} = 0$.

Різні типи зсувів кодуються розрядами ($I_{10}..I_6$) поля СУСЗ_МІ мікрокоманди (табл. 2.19) відповідно до табл. 2.16. Арифметичним зсувам відповідають наступні управляючі сигнали, що надходять на входи ВР2: зсув вправо $I_{10}..I_6 = 01110$, вліво – $I_{10}..I_6 = 10010$.

Таблиця 2.25. Кодова карта

№ так- ту	Адреса [16]	Константа БОД		БОД													
				BC1 (АЛБ)				BP2 (СУСЗ)									
		D [16]	\overline{OED}	АЛБ_MI [2]	A [16]	B [16]	\overline{OEY}	СУСЗ_MI [2]	\overline{EC}	\overline{EZ}	\overline{EN}	\overline{EV}	\overline{CEN}	\overline{CEM}	\overline{OECT}	\overline{SE}	
1	0000	000F	0	000 011 111	*	*	1	00 00000 000000	0	0	0	0	1	1	1	1	
2	0001	FFFD	0	101 001 110	*	0	1	01 01110 000000	0	0	0	0	1	1	1	0	
3	0002	0007	0	111 011 111	*	1	1	00 10010 000000	0	0	0	0	1	1	1	0	
4	0003	****	1	011 100 001	0	1	1	00 00000 000000	1	1	1	1	1	1	1	1	

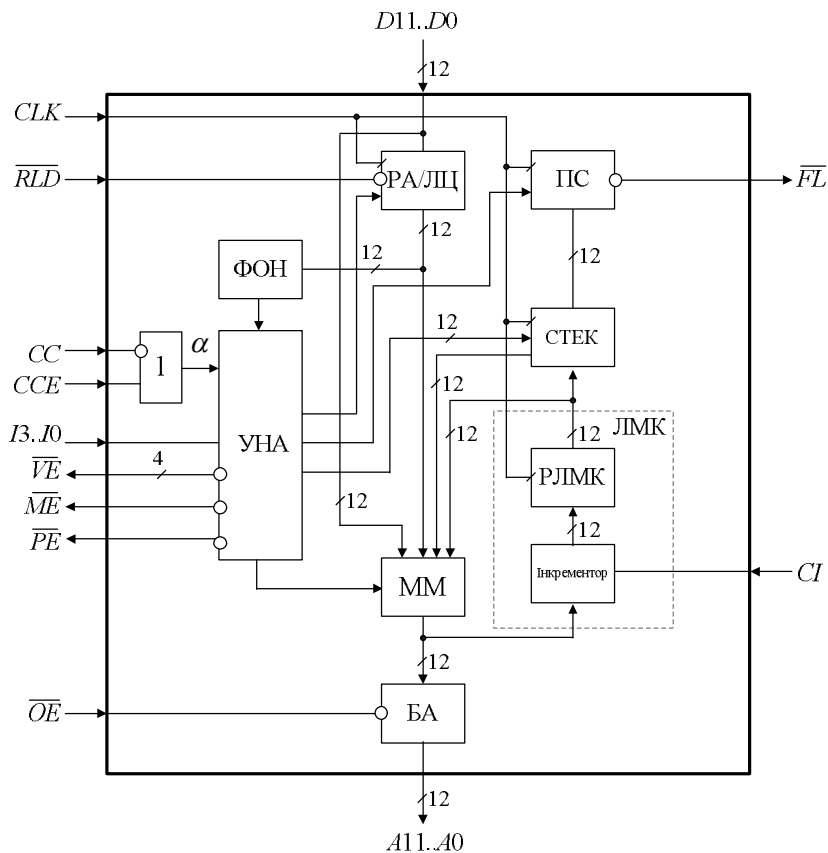
Таблиця 2.26. Цифрова діаграма стану регістрів

№ так- ту	Вихідний стан				Інформація у приймачі результату, [16]	Мікрооперація
	Шина D, [16]	R0, [16]	R1, [16]	RQ, [16]		
1	000F	****	****	****	$RQ := 000F$	$RQ := D(000F) \vee 0$
2	FFFD	****	****	000F	$R0 := 0009$	$R0 := SRA[RQ - D(FFFD) - 1 + 1(CI)]$
3	0007	0009	****	000F	$R1 := 000E$	$R1 := SLA[D(0007) \vee 0]$
4	****	0009	000E	000F	$R1 := 0008$	$R1 := R1 \vee R0$
			0008			Результат

Примітка: SRA, SLA – мнемонічне позначення зсувів арифметичних вправо та вліво відповідно.

2.5. Блоки мікропрограмного управління

Мікросхема K1804BY4 призначена для управління послідовністю вибірки мікрокоманд з пам'яті мікрокоманд (ПМК). Структура ІС K1804BY4 зображена на рис. 2.25.



с 2.25. Структурна схема ІС K1804BY4

Ри

Мікросхема K1804BY4 складається з наступних функціональних частин:

- УНА — схема управління вибіркою адреси наступної мікрокоманди;
- ПА/ЛЦ — регістр адреси/лічильник циклів;

ЛМК	–	лічильник мікрокоманд;
ПС	–	покажчик стека;
РЛМК	–	регістр лічильника мікрокоманд;
ММ	–	мультиплексор мікрокоманди;
ФОН	–	формував ознаки нуля;
БА	–	буфер адреси.

Умовне графічне позначення ІС К1804ВУ4 показано на рис. 2.26, а призначення виводів наведене в табл. 2.26.

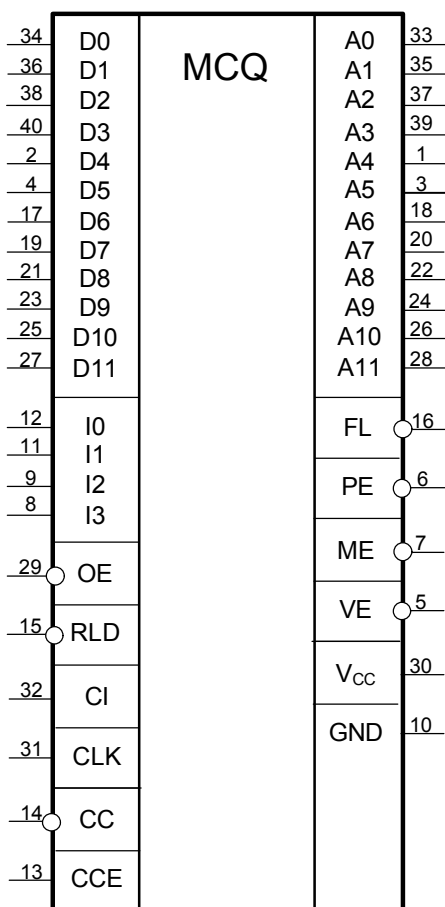


Рис. 2.26. Умовне графічне позначення ІС К1804ВУ4

ІС К1804ВУ4 виконує функції формувача адреси наступної мікрокоманди (ФАМ) і забезпечує формування дванадцятирозрядних адрес мікрокоманд у ПМК. Внаслідок чого ємність ПМК складає 4К 83-розрядних слів МК.

Виходи *A* адреси мікрокоманди (рис. 2.26) мають три стани (0, 1 і високоомний стан), що дозволяє, за необхідності, забезпечити зовнішній доступ до входів адреси ПМК, наприклад, для тестової перевірки послідовності мікрокоманд, виходу на початкову мікропрограму ініціалізації системи і таке інше.

Таблиця 2.26. Функціональне призначення виводів ІС ВУ4

Номер виводу	Позначення виводу	Функціональне призначення
33, 35, 37, 39, 1, 3, 18, 20, 22, 24, 26, 28	<i>A11 – A0</i>	Виходи адреси мікрокоманд
34, 36, 38, 40, 2, 4, 17, 19, 21, 23, 25, 27	<i>D11 – D0</i>	Входи адреси розгалуження
5, 6, 7	<i>VE, PE, ME</i>	Виходи дозволу підключення до входів <i>D</i> Буферу <i>V</i> , Буферу <i>P</i> та Буферу <i>M</i> відповідно
12, 11, 9, 8	<i>I3 – I0</i>	Входи коду інформаційного слова мікроінструкції переходу
10	<i>GND</i>	Вивід живлення (загальний)
13	<i>CC</i>	Вхід умови
14	<i>CCE</i>	Вхід дозволу умови
15	<i>RLD</i>	Вхід дозволу запису в РА/ЛС
16	<i>FL</i>	Вихід ознаки заповнення стеку
29	<i>OE</i>	Вхід дозволу видачі адреси
30	<i>VCC</i>	Вихід живлення (+5В)
31	<i>CLK</i>	Вхід синхросигнала
32	<i>CI</i>	Вхід переносу в лічильник мікрокоманд

Принцип формування адреси наступної мікрокоманди

Спосіб формування адреси мікрокоманди визначається зовнішніми управляючими сигналами на входах *I3 – I0*, \overline{CC} , *CCE* (рис. 2.25) і внутрішнім сигналом, який формується у схемі ФОН.

Схема УНА, що є перетворювачем кодів, виробляє набір внутрішніх управляючих сигналів і зовнішні сигнали \overline{PE} , \overline{VE} і \overline{ME} . Внутрішні управляючі сигнали виконують настройку блоків ІС схеми на виконання необхідних функцій. Зовнішні управляючі сигнали використовують для підключення до входів $D11 - D0$ ІС відповідних джерел інформації.

Мультиплексор ММ управляється внутрішніми сигналами, що надходять зі схеми УНА, та виконує функцію вибору одного з чотирьох джерел надходження адреси наступної мікрокоманди.

Таким чином, джерелами адреси наступної мікрокоманди є:

- входи $D11 - D0$,
- регістр адреси/лічильник циклів – РА/ЛЦ,
- СТЕК,
- лічильник мікрокоманд – ЛМК.

Обрана адреса через БА видається на виходи A за встановлення управляючого сигналу $\overline{OE} = 0$.

Дані в дванадцятирозрядний регістр РА/ЛЦ надходять з входів $D11 - D0$ та записуються за додатним перепадом синхросигналу CLK , якщо виконується певна управляюча мікроінструкція, що надходить на входи $I3 - I0$ ІС. Під час запису даних у РА/ЛЦ встановлюється управляючий сигнал $\overline{RLD} = 0$, при цьому дані у РА/ЛЦ можуть бути записані і незалежно від виконуваної мікроінструкції.

При виконанні відповідних мікроінструкцій ($I3 - I0$) вміст РА/ЛЦ зменшується на одиницю за додатним сигналом CLK , що дозволяє використовувати РА/ЛЦ в якості лічильника циклів. Ознака того, що вміст лічильника циклів дорівнює нулю – РА/ЛЦ = 0, формується у схемі ФОН.

Таким чином, у РА/ЛЦ надходить інформація з зовнішніх входів D (при умові встановлення $\overline{RLD} = 0$), і у подальшому використовується або в якості адреси наступної мікрокоманди, або як кількість повторів циклів.

Лічильник мікрокоманд складається з дванадцятирозрядного РЛМК та ІНКРЕМЕНТОРА, що є комбінаційною схемою додавання вхідного переносу CI до вхідного слова. Будь-яка адреса, з виходу мультиплексора ММ записується за додатним перепадом CLK в регістр РЛМК, під час встановлення сигналу $CI = 0$, або збільшеною на одиницю, під час встановлення сигналу $CI = 1$. Таким чином на-

прикінці кожного такту у ЛМК формується адреса наступної мікрокоманди у вигляді $A_{i+1} = A_i + 1$, яка у наступному такті при виконанні лінійних переходів є джерелом адреси наступної мікрокоманди.

Стек складається з покажчика стеку (ПС) і накопичувача (СТЕК), що має п'ять регістрів (глибина стеку дорівнює п'яти). Схема ПС є реверсивним лічильником, який управляється внутрішніми сигналами і змінює свій стан за додатним перепадом CLK .

Стек організований за принципом *LI/FO* (*Last Input / First Output* – останній прийшов / перший обслугований). Схема ПС вказує на регістр, у який за останнім зверненням здійснювався запис даних.

Залежно від виконуваних мікроінструкцій ($I3 - I0$) стек працює в наступних режимах:

- *очищення стека* – встановлення покажчика стеку в нуль (ПС = 0);
- *збереження інформації* – ПС не змінює свого стану, з відповідного регістру накопичувача СТЕК (визначеному ПС) здійснюється зчитування даних;
- *завантаження стеку* – за додатним перепадом CLK значення ПС збільшується на одиницю в послідовності 0, 1, 2, 3, 4, 5, після чого здійснюється запис інформації в регістр накопичувача СТЕК; при заповненні стека (ПС = 5) встановлюється сигнал $\overline{FL} = 0$ і при подальшому завантаженні стеку ПС не змінює свого стану;
- *виштовхування із стеку* – відбувається зчитування інформації із стеку і зменшення ПС на одиницю в послідовності 5, 4, 3, 2, 1, 0; при встановленні ПС = 0 операція виштовхування із стеку приводить до зчитування невизначеної інформації, при цьому ПС не змінює свого стану.

ІС К1804ВУ4 дозволяє реалізувати 16 мікроінструкцій, які застосовуються для розгалуження мікропрограм і забезпечують:

- лінійні переходи – формування наступної адреси (інкремент);
- багаторазове повторення однієї й той самої адреси;
- умовні та безумовні переходи;
- організацію циклів;
- умовні та безумовні виклики мікропідпрограм.

Мікроінструкції є безумовними і умовними, тобто виконуваними при виконанні певної умови. В залежності від сигналів на вхо-

дах \overline{CC} і CCE формується ознака виконання умови $\alpha = \overline{CC} \vee CCE$ (рис. 2.25). Якщо умова виконується встановлюється $\alpha = 1$, інакше $\alpha = 0$. Таким чином умова виконується, якщо $\overline{CC} = 0$ або $CCE = 1$. Незалежно від значення сигналу на вході \overline{CC} , можна примусово забезпечити виконання умови $\alpha = 1$, встановив $CCE = 1$, тобто забезпечити безумовний перехід.

Характер переходів під час виконання мікроінструкцій наведено у табл. 2.27.

На базі ІС K1804BY4, призначеної для формування та управління послідовністю вибору мікрокоманд із ПМК можна побудувати БМУ, структурна схема якого зображена на рис. 2.27.

БМУ складається з наступних функціональних частин:

- ФАМ — схема формування вибіркою адреси наступної мікрокоманди, реалізована на ІС BY4;
- ПМК — пам'ять мікрокоманд;
- РМК — регістр мікрокоманди;
- МУ — мультиплексор вибору логічних умов;
- ІНВЕРТОР — інвертор;
- три буфери джерел адрес мікрокоманд:
- Буфер M , — буфер регістра адреса,
- Буфер V , — буфер регістра вектора переривань,
- Буфер P — буфер регістра константи;
- ШАР — дванадцяти розрядна шина адреси розгалуження.

Для управління БМУ у структурі мікрокоманди відведено два-цять три розряди (рис. 2.28):

- P — Дванадцятирозрядна частина поля констант МК (поле P), що містить адресу переходу або константу, наприклад, кількість повторень циклів;
- ФАМ_ІС — чотирирозрядне інформаційне слово, що визначає мікроінструкцію переходу;
- CCE — дозвіл аналізу умови;
- COM — інвертування сигналу логічної умови;
- CI — дозвіл формування адреси наступної МК;
- \overline{RLD} — дозвіл запису константи з поля P МК в РА/ЛЦ ФАМ;
- MS — трирозрядне поле управління вибором входу МУ.

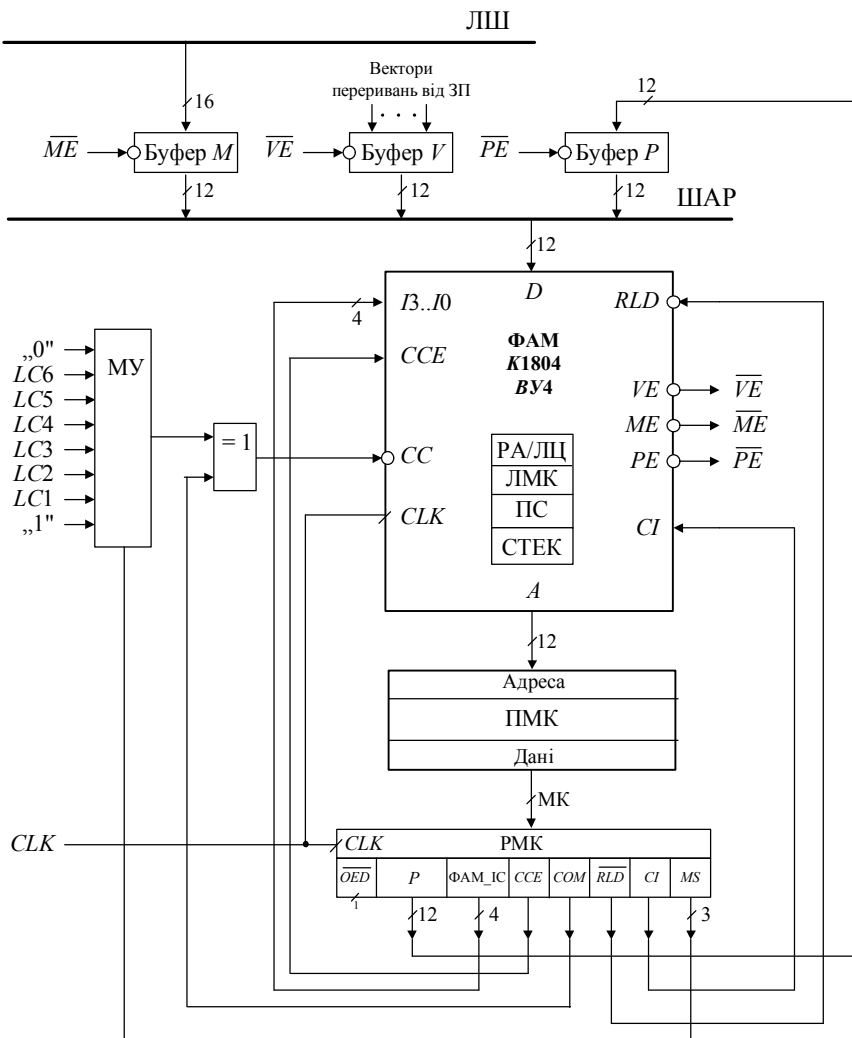


Рис 2.27. Структурна схема БМУ

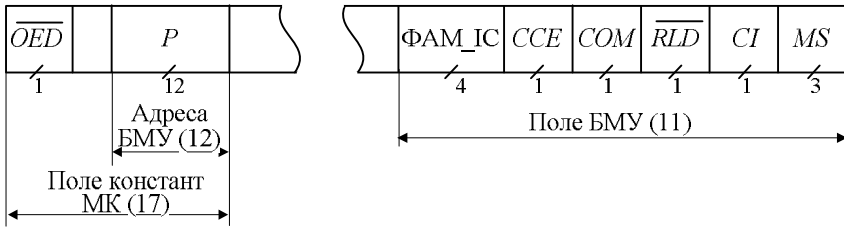


Рис. 2.28. Поля мікрокоманди, які використовуються для управління блоком мікропрограмного управління

Принцип функціонування БМУ

Буфери M , V і P , що мають виходи з трьома станами, призначені для видачі на ШАР дванадцятирозрядних адрес. Сигнали \overline{ME} , \overline{VE} і \overline{PE} що формує УНА (рис. 2.26) відкривають Буфери M , V і P , відповідно – $\overline{ME} = 0$, $\overline{VE} = 0$ і $\overline{PE} = 0$. У поточному машинному такті може бути відкритий тільки один з буферів. Таким чином, якщо джерелом адреси наступної мікрокоманди у ІС ВУ4 є входи $D11 - D0$, то інформація на ці входи видається з одного з буферів M , V або P по ШАР.

Зазвичай Буфер M та сигнал $\overline{ME} = 0$ використовується для прийому початкової адреси M мікропрограми, який визначається полем коду операції команди та надходить у БМУ з локальної шини МПС.

Через Буфер V ($\overline{VE} = 0$) здійснюється прийом початкової адреси V мікропрограми обслуговування переривання від зовнішнього пристрою за певним вектором.

Сигнал \overline{PE} управляє Буфером P , через який з поля P регістра мікрокоманди на шину ШАР, а далі на шину D схеми ФАМ поступає адреса розгалуження або параметр циклу.

Зовнішні логічні умови $LC1 - LC6$ надходять на вхід МУ, що комутує ці умови, а також значення 0 та 1 на вихід. З виходу МУ логічний сигнал надходить на вхід елементу ВИКЛЮЧНЕ АБО, який виконує функцію керованого інвертора, і далі на вхід \overline{CS} схеми ФАМ. Інвертуванням сигналу умови управляє сигнал COM поля БМУ мікрокоманди. При встановленні сигналу $COM = 1$ логічна умова інвертується.

Слід зазначити, що зовнішні логічні умови, що надходять на входи *LC1–LC6* мультиплексора МУ, формуються на виході СТ БОД (рис. 2.6).

Таким чином, БМУ дозволяє реалізувати такі типові управляючі конструкції, як безумовні і умовні переходи, цикли, мікропідпрограми.

Безумовний перехід означає передачу управління за певною адресою, не залежною від виконання логічних умов. Під час умовного переходу передача управління здійснюється за одною з двох або більше адрес залежно від виконання умов.

Для реалізації умовних переходів можна використовувати наступні мікроінструкції *CJP*, *CJV*, *JRP*, *CJPP* і *TWB*, а безумовних – *IZ*, *JMAP*, *LDCT* і *CONT* (табл. 2.27). Крім того, будь яку умовну мікроінструкцію можна перетворити на безумовну, якщо встановити на вході ВУ4 сигнал $CCE = 1$. При цьому, як вже було зазначено, інструкція виконуватиметься так, як встановиться сигнал $\alpha = 1$.

Циклічні конструкції мікропрограм зазвичай організовують з використанням мікроінструкцій *RFCT*, *RPCT*, *LOOP* і *TWB*. За цього інструкція *LOOP* забезпечує вихід з циклу за умовою, а інші – за нульовим вмістом лічильника циклів (РА/ЛЦ), тобто за кількістю повторень циклу. Для організації циклів можна також використовувати умовні переходи, вказавши в якості адреси переходу адресу мікрокоманди початку циклу.

Для переходу до мікропідпрограм зручно користуватися інструкціями *CJS*, *JSRP*, при виконанні яких адреса повернення із мікропідпрограми зберігається в стеку. Інструкція *CRTN* забезпечує повернення з мікропідпрограми за адресою, що вслід за цим виштовхується із стеку. Наявність стеку дозволяє організувати вкладені мікропідпрограми, тобто забезпечити звернення з однієї мікропідпрограми до іншої. Стек може берегти не більш ніж п'ять адрес повернення, що визначає максимальне число вкладень мікропідпрограм.

При застосуванні мнемонічного запису мікрокоманд, виконуваних схемою ФАМ можна використовувати мнемоніку найбільш розповсюджених логічних умов, що формуються СУСЗ:

- *z* – задалегідь хибна умова (умова ніколи не виконується);
- *nz* – задалегідь істинна умова (умова завжди виконується);
- *zo*, *co*, *no*, *vo* – входи ознак стану *IZ*, *IC*, *IN*, *IV* СУСЗ;

- rm_z, rm_c, rm_n, rm_z – розряди регістру стану RM : MZ, MC, MN, MV ;
- rn_z, rn_c, rn_n, rn_z – розряди регістру стану RN : NZ, NC, NN, NV ;
- $nxorc$ – ознака $\overline{IN \oplus IV}$;
- $zxorc$ – ознака $\overline{IC \oplus IZ}$;
- 11, 12, 13, 14, 15, 16 – безпосередньо аналізувати інформацію на входах МУ: $L1, L2, L3, L4, L5, L6$;
- $ct, rdm, rdd, int, irq0, irq1, \dots, irq7$ – підключати до входів МУ сигнали, що генерують різні пристрої МПС: СУСЗ, пам'ять, зовнішні пристрої, сигнали вимоги переривань.

Для підключення інвертованих логічних умов можна застосовувати ту саму мнемоніку з префіксом `not`, наприклад, `not rm_c`.

Слід зазначити, якщо умова записується з префіксом `not`, то в поле *COM* мікрокоманди встановлюється значення 1.

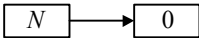

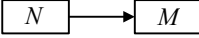
Як уже було зазначено формуванням значення α на вході схеми УНА на основі логічної умови (ЛУ), що надійшла на вхід МУ, управляють поля мікрокоманди *CCE* та *COM* (рис. 2.25, рис. 2.27). Далі розглянуті декілька типових комбінацій цих полів, що забезпечують формування α .

- $CCE = 0, COM = 1 \Rightarrow$ перевірка умови, $\alpha = ЛУ$, наприклад, `{cjp rm_c, addr;};`
- $\overline{CCE} = 0, COM = 0 \Rightarrow$ перевірка інвертованої умови, $\alpha = \overline{ЛУ}$, наприклад, `{cjp not rm_c, addr;};`
- $CCE = 1, COM = * \Rightarrow$ безумовний перехід, $\alpha = 1$, наприклад, `{cjp nz, addr;};`

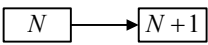
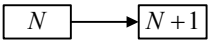
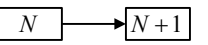
Реалізація мікрокоманд умовних та безумовних переходів наведена у прикладах 2.6 – 2.11. Для виводу коду умови через шину *CT* схеми СУСЗ в розглянутих прикладах встановлюється $\overline{OECT} = 0$, що у наведених фрагментах мікрокоманд умовно не показано.

В прикладах 2.12 – 2.16 представлено застосування управляючих конструкцій БМУ, організація циклів, реалізація мікропідпрограм.

Таблиця 2.27. Управляючі конструкції БМУ

Мнемоніка мікрооперації	$(I_3..I_0)$	Сигнали та мікро- операції	Коментар	Графічна інтерпретація
1	2	3	4	5
Мікрокоманди безумовних переходів				
{jz;} – перехід до ну- льової адреси	0000	$A := 0000h$ $0 \rightarrow$ показчик стека ;	ЛМК встановлюється в значен- ня ; відбувається; очистка стека	
{cont;} – перехід до на- ступної адреси	1110	$A := CMK;$	Мікрокоманда призводить до інкременту ЛМК, і формування на виході A ФАМ адреси $A_{i+1} := A_i + 1$.	
{jmap;} – безумовний пе- рехід за адресою, що передається з Буферу M	0010	$\overline{ME} = 0 ;$ $A := D;$	Використовується під час ему- ляції системи команд для пере- ходу в ПМК до нової мікропод- програми. При цьому на ЛШ має бути виставлена початкова адре- са мікропідпрограми. Під час виконання МК форму- ється сигнал $\overline{ME} = 0$, який від- криває Буфер M , і адреса нової мікропідпрограми з ЛШ надхо- дить на вхід D ФАМ і далі на вихідну шину A ($A := D$).	

Продовження табл. 2.27.

1	2	3	4	5
{ldct val;} – завантаження <i>РА/ЛЦ та перехід до наступної ад- реси</i>	1100	$\overline{PE} = 0;$ $\overline{RLD} = 0;$ $A := CMK;$ $P \rightarrow RA/ЛЦ$	Значення <i>val</i> у цьому випадку є вмістом поля <i>P</i> мікрокоманди. Під час виконання МК виробля- ється сигнал $\overline{PE} = 0$, який відкриває Буфер <i>P</i> , і зна- чення з поля <i>P</i> (кількість повторень циклу) надхо- дить на вхід <i>D</i> ФАМ та записується в РА/ЛЦ. При цьому встановлюється $\overline{RLD} = 0$.	 $P \rightarrow RA/ЛЦ$
{push;} – запис в стек	0100	$A := ЛМК;$ $ЛМК \rightarrow \text{стек};$	Мікрокоманда записує в стек наступну адресу, наприклад, якщо мікрокоманду розміщено в ПМК за адресою $2d3h$, то під час її виконання в стек буде записано значення $2d4h$.	 $N + 1 \rightarrow \text{стек}$
{push cond, val;} – запис в стек та умовне заванта- ження РА/ЛЦ	0100	$\overline{PE} = 0;$ $A := ЛМК;$ $ЛМК \rightarrow \text{стек};$ $P \rightarrow RA/ЛЦ, \text{ за } \alpha = 0$ $\overline{RLD} = 0$	Мікрокоманда записує в стек наступну адресу, завантажує в РА/ЛЦ значення <i>val</i> , якщо вико- нується умова (<i>cond</i> = 1). При цьому встановлюється $\overline{RLD} = 0$. Якщо умова не виконується, то запис <i>val</i> в РА/ЛЦ не відбувається. Мікрокоманда рівноцінна двом мікрокомандам {push; } та {ldct val; }.	 $N + 1 \rightarrow \text{стек}$ за $\alpha = 1, P \rightarrow RA/ЛЦ$

Продовження табл. 2.27.

1	2	3	4	5
Мікрокоманди умовних переходів				
{cjp cond, addr;} {cjp cond, ml1;} – умовний перехід за адресою P	0011 	$\overline{PE} = 0$; за $\alpha = 0$, $A := \text{ЛМК}$; за $\alpha = 1$, $A := D$;	Мікрокоманда cjp (<i>Condition JumP</i>) дозволяє здійснювати розгалуження в мікропрограмах. Під час виконання мікрокоманди формується сигнал $\overline{PE} = 0$, який відкриває Буфер P , і значення $addr$ з поля P мікрокоманди надходить на вхід D ФАМ. Якщо умова виконується ($cond=1$), то $A := addr$, інакше $A := \text{ЛМК}$.	<pre> graph LR N[N] -- alpha=0 --> Nplus1[N+1] N -- alpha=1 --> P[P] </pre>
{cjpp cond, addr;} {cjpp cond, ml1;} – умовний перехід за адресою P і виштовхування із стеку	1011 	$\overline{PE} = 0$; за $\alpha = 0$, $A := \text{ЛМК}$; за $\alpha = 1$, $A := D$, $\text{ПС} = \text{ПС} - 1$;	Мікрокоманда cjpp (<i>Condition JumP and Pop</i>) відрізняється від МК cjp лише тим, що у разі виконання умови ($cond = 1$) додатково з верхівки стека виштовхується значення. Отже, цю мікрокоманду доцільно використовувати для виходу з циклу за умовою.	<pre> graph LR N[N] -- alpha=0 --> Nplus1[N+1] N -- alpha=1 --> P[P] P --- PSminus1[ПС-1] </pre>

Продовження табл. 2.27.

1	2	3	4	5
{jrp cond, addr;} – умовний перехід за адресою P або за адресою з $PA/ЛЦ$	1011	$\overline{PE} = 0$; за $\alpha = 0$, $A := PA/ЛЦ$; за $\alpha = 1$, $A := D$;	Сигнал $\overline{PE} = 0$ відкриває Буфер P , значення $addr$ з поля P мікрокоманди надходить на вхід D ФАМ. Якщо умова виконується ($cond = 1$), то $A := addr$, і відбувається перехід за адресою, вказаною в мікрокоманді; інакше $A := (PA/ЛЦ)$, і відбувається перехід за адресою, яка зберігається в $PA/ЛЦ$.	
{cјv cond;} – умовний перехід на підпрограму обробки переривання	0110	$\overline{VE} = 0$; за $\alpha = 0$, $A := ЛМК$; за $\alpha = 1$, $A := D$;	Мікрокоманда $cјv$ (<i>Condition Jump by Vector</i>) дозволяє здійснити перехід на мікропрограму обслуговування переривання (драйвер). На Буфер V надходять сигнали від зовнішніх пристроїв системи. Під час виконання цієї мікрокоманди формується сигнал $\overline{VE} = 0$, який відкриває Буфер V , і початкова адреса драйвера надходить на вхід D ФАМ. Якщо умова виконується ($cond = 1$), то $A := V$, інакше – $A := ЛМК$.	

Продовження табл. 2.27.

1	2	3	4	5
Мікрокоманди для організації циклів				
{rfct;} – повторити цикл (перехід за адресою з верхівки стека), якщо $РА/ЛЦ \neq 0$	1000	$\overline{PE} = 0;$ за $РА/ЛЦ \neq 0$, $A := \text{Стек}$, $РА/ЛЦ := РА/ЛЦ - 1;$ за $РА/ЛЦ = 0$, $A := \text{ЛМК}$, $ПС := ПС - 1;$	Використовують для багаторазового повторення певної частини мікропрограми, якщо відома кількість повторень. Кількість повторень перед початком виконання циклу записують в $РА/ЛЦ$ за допомогою мікрокоманди {push cond, val;}. При цьому в стек заноситься адреса початку циклу. Після виконання тіла циклу, який слід завершити мікрокомандою {rfct;}, аналізується вміст $РА/ЛЦ$. Якщо $РА/ЛЦ \neq 0$, здійснюється перехід на початок циклу (за адресою зі стека), а значення в $РА/ЛЦ$ зменшується на 1. Повторення циклу відбувається, поки виконується $РА/ЛЦ \neq 0$. Інакше ($РА/ЛЦ = 0$) здійснюється перехід до наступної мікрокоманди й виштовхування зі стека. Механізм виконання мікрокоманди {rfct;} такий, що декримент $РА/ЛЦ$ відбувається після аналізу вмісту $РА/ЛЦ$. Тому завжди тіло циклу буде виконане на один раз більш ніж указано у параметрі val, вслід чого необхідна кількість циклів має бути зменшена на одиницю.	<pre> graph TD N[N] -- "РА/ЛЦ = 0" --> N1[N+1] N1 -- "ПС-1" --> End1[] N -- "РА/ЛЦ ≠ 0" --> S[S] S -- "РА/ЛЦ-1" --> End2[] </pre>

Продовження табл. 2.27.

1	2	3	4	5
{rpct addr;}		$\overline{PE} = 0;$	У мікрокоманді перехід відбувається за адресою <i>addr</i> з поля <i>P</i> мікрокоманди, що надійшла на вхід <i>D</i> ФАМ через Буфер <i>P</i> ($\overline{PE} = 0$). У іншому механізм роботи МК співпадає із МК {rfct; }, що розглянута раніш.	
– повторити цикл (перехід за адресою <i>P</i>), якщо $РА/ЛЦ = 0$	1001	за $РА/ЛЦ \neq 0$, $A := D$, $РА/ЛЦ := РА/ЛЦ - 1$; за $РА/ЛЦ = 0$, $A := ЛМК$;		
{loop cond;}		$\overline{PE} = 0;$	Мікрокоманда {loop cond; } використовується для завершення тіла циклу, якщо цикл має заздалегідь невідому кількість повторень. Якщо умова <i>cond</i> = 0, то здійснюється перехід на початок циклу (за адресою збереженою у верхівці стеку), інакше здійснюється перехід до наступної мікрокоманди, що розташована наступною після МК завершення циклу і виштовхування зі стека адреси початку циклу.	
– вихід з циклу за умовою	1101	за $\alpha = 0$, $A := S$; за $\alpha = 1$, $A := ЛМК$, $ПС := ПС - 1$;		

Продовження табл. 2.27.

1	2	3	4	5
{twb cond, addr;}		$\overline{PE} = 0;$		
– розгалуження на три напрями	1111	<p>за $(\alpha = 0) \&$ $(РА/ЛЦ \neq 0),$ $A := \text{Стек},$ $РА/ЛЦ := РА/ЛЦ - 1;$</p> <p>за $(\alpha = 0) \&$ $(РА/ЛЦ = 0),$ $A := D,$ $ПС := ПС - I;$</p> <p>за $\alpha = 1,$ $A := \text{ЛМК}, ПС := ПС - I;$</p>	<p>Під час виконання мікрокоманди {twb cond, addr;} аналізуються дві умови: <i>cond</i>, що явно вказана в мікрокоманді, та вміст РА/ЛЦ, що порівнюється з нулем. Якщо виконуються дві умови $РА/ЛЦ \neq 0$ і $cond = 0$, то здійснюється перехід за адресою з верхівки стека $A := \text{Стек}$ і вміст РА/ЛЦ зменшується на одиницю. Якщо $РА/ЛЦ = 0$ і $cond = 0$, то здійснюється перехід за адресою <i>addr</i> з поля <i>P</i> мікрокоманди, що надійшла на вхід <i>D</i> ФАМ через Буфер <i>P</i> ($\overline{PE} = 0$), і виштовхування адреси повернення на початок циклу зі стека. Якщо виконується умова $cond = 1$, то здійснюється перехід до наступної після виконання циклу адреси $A := \text{ЛМК}$.</p>	<pre> graph TD Start(()) --> N[N] N -- "α = 1" --> N1[N+1] N1 --> PC1[PC-1] N -- "α = 0" --> S[S] S --> RA_LC1[РА/ЛЦ-1] N -- "α = 0" --> P[P] P --> PC1 style Start fill:none,stroke:none style PC1 fill:none,stroke:none </pre>

Продовження табл. 2.27.

1	2	3	4	5
Мікрокоманди для роботи з мікропідпрограмами				
{cjs cond, addr;} – виклик мікропрограми за умовою	0001	$\overline{PE} = 0;$ за $\alpha = 0, A := \text{ЛМК};$ за $\alpha = 1, A := D,$ $\text{ЛМК} \rightarrow \text{Стек};$	Мікрокоманда {cjs cond, addr;} (<i>Condition Jump to Subroutine</i>) забезпечує реалізацію механізму виклику мікропідпрограми за умовою, відбувається перехід на мікропідпрограму, що знаходиться в ПМК за адресою <i>addr</i> , якщо вказана умова виконується ($\text{cond} = 1$). При цьому в стек записується адреса повернення (тобто адреса наступної за cjs мікрокоманди), а на вихідну шину <i>A</i> ФАМ надходить <i>addr</i> (з поля <i>P</i> через Буфер <i>P</i>), інакше здійснюється перехід до наступної мікрокоманди.	
{crtm cond;} – умовне повернення з мікропідпрограми	1010	$\overline{PE} = 0;$ за $\alpha = 0, A := \text{ЛМК};$ за $\alpha = 1, A := \text{Стек},$ $\text{ПС} := \text{ПС} - 1;$	Мікрокоманда {crtm cond;} (<i>Condition ReTurn</i>) забезпечує повернення з мікропідпрограми. Якщо виконується умова $\text{cond} = 1$, то виконання мікропідпрограми припиняється, здійснюється повернення в основну мікропрограму за адресою з верхівки стеку ($A := \text{Стек}$) і виштовхування зі стеку. Інакше відбувається перехід до наступної мікрокоманди мікропідпрограми.	

Продовження табл. 2.27.

1	2	3	4	5
{jsrp cond, addr;} – перехід до однієї з двох мікропідпрограм	0101	$\overline{PE} = 0;$ за $\alpha = 0$, $A := PA/ЛЦ$, $ЛМК \rightarrow \text{Стек};$ за $\alpha = 1$, $A := D$;	Під час виконання мікрокоманди {jsrp cond, addr;} в стек записується адреса повернення з МПП (тобто адреса наступної за JSRP мікрокоманди) та здійснюється виклик однієї з двох мікропідпрограм – мікропідпрограми, що розміщена за адресою <i>addr</i> , якщо виконується умова <i>cond</i> = 1. (за цього значення <i>addr</i> з поля <i>P</i> МК через Буфер <i>P</i> надходить на вихід <i>A</i> ФАМ, під час встановлення сигналу $\overline{PE} = 0$), або мікропідпрограми, що розміщена за адресою з РА/ЛЦ (за цього $A := PA/ЛЦ$).	

Примітки: *N* – адреса виконуваної МК; *cond* – логічна умова; *addr* – адреса переходу; інші позначення відповідають рис. 2.27.

Приклад 2.6. Якщо результат обчислення суми у поточному такті від'ємний ($NO = 1$), перейти за адресою 0007H (за міткою *m1*), при цьому виконати виштовхування даних зі стеку, інакше продовжити обчислення.

Мікрокоманда

Адреса [16]	<i>D</i> [16]	БМУ_МІ [2]	<i>CCE</i>	<i>COM</i>	<i>CI</i>	\overline{RLD}	<i>MS</i>	СУСЗ_МІ $I_5 - I_0$	<i>P</i>	<i>CT</i>
0000	0007	1011	0	1	1	1	1	111110	0007	<i>IN</i>

Мнемонічний запис

```

link l1:ct;
000      {add r1,r1,r0;cjpp no, m1;}
...      {}
007  m1   {}

```

Приклад 2.7. Виконати безумовний перехід на мітку *m1*, інакше перейти за адресою з РА/ЛЦ.

Мікрокоманда

Адреса [16]	<i>D</i> [16]	БМУ_МІ [2]	<i>CCE</i>	<i>COM</i>	<i>CI</i>	\overline{RLD}	<i>MS</i>	СУСЗ_МІ $I_5 - I_0$	РА/ЛЦ	<i>P</i>	<i>CT</i>
0000	0007	0111	1	*	1	1	*	000000	000A	0007	*

Мнемонічний запис

```

0000      {jrp nz,m1;}
...      {}
0007  m1   {}
...
000A      {}

```

Приклад 2.8. Якщо результат виконання підсумовування не дорівнює нулю то перейти на мікроподпрограму за адресою, що надійшла з Буферу *V*.

Мікрокоманда

Адреса [16]	D [16]	БМУ_МІ [2]	CCE	COM	CI	\overline{RLD}	MS	СУС3_МІ $I_5 - I_0$	V	CT
0000	0007	0110	0	0	1	1	2	110100	0007	IZ

Мнемонічний запис

```

link l2:ct;
0000 {add r1,r1,z;cjv not zo;}

```

Приклад 2.9. Завантажити у РА/ЛЦ значення, якщо виконується умова $CT = 1$.

Мікрокоманда

Адреса [16]	D [16]	БМУ_МІ [2]	CCE	COM	CI	\overline{RLD}	MS	СУС3_МІ $I_5 - I_0$	P	CT
0000	0006	0100	1	0	1	0	1	000100	0006	NZ

Мнемонічний запис

```

link l1:ct;
0000 {push ct,6;} \ якщо  $CT=1$ , то РА/ЛЦ:=6

```

Приклад 2.10. Якщо отримано нульовий результат ($ZO = 1$), то перейти на мітку `m11`, інакше продовжити обчислення.

Мікрокоманда

Адреса [16]	D [16]	БМУ_МІ [2]	CCE	COM	CI	\overline{RLD}	MS	СУС3_МІ $I_5 - I_0$	P	CT
0000	0007	0011	0	1	1	1	1	110100	0007	IZ

Мнемонічний запис

```

link l1:ct;
0000 {cjp zo,m11;}
      {}
0007 m11 {}

```

Приклад 2.11. Якщо $NV = 1$, то здійснюється виклик мікроподпрограми за адресою SUB .

Мікрокоманда

Адреса [16]	D [16]	БМУ_МІ [2]	CCE	COM	CI	\overline{RLD}	MS	СУСЗ_МІ $I_5 - I_0$	P	CT
0000	0007	0001	0	1	1	1	4	000110	0007	NV

Мнемонічний запис

```

link l4:ct;
0000    {cjs rn_v, sub;}    \ перевірка логічної умови
                                \ якщо  $NV = 1$  – перехід на МПП

...
0007 sub {}                \ початок МПП  $SUB$ 
```

Приклад 2.12. Організувати цикл із заданою кількістю повторень можна в такий спосіб:

Мікрокоманда

Адреса [16]	D [16]	БМУ _{MI} [2]	CCE	COM	CI	\overline{RLD}	\overline{OECT}	MS	СУСЗ _{MI} $I_5 - I_0$	Коментарі		
										P	Стек	CT
0000	000C	0100	1	*	1	0	1	*	000000	000C	0001	*
0001	****	1110	*	*	1	1	1	*	000000	****	0001	*
...												*
0009	000F	1011	0	1	1	1	0	1	101010	000F	0000	MC
...												
000E	****	1000	*	*	1	1	1	*	000000	****	0001	*
000F	****	1110	*	*	1	1	1	*	000000	****	****	*

Мнемонічний запис

```

link l1:ct;
0000      {push nz,12;}      \ кількість повторень циклу
0001      {}                \ початок циклу
...
0009      {cjmp rm_c,mex}    \ перевірка логічної умови, якщо MC = 1 – вихід з циклу
...
000E      {rfct;}           \ кінець циклу
000F      mex {}            \ продовження мікропрограми

```

Приклад 2.13. Організувати цикл із виходом за умовою можна у такий спосіб:

Мікрокоманда

Адреса [16]	D [16]	БМУ_МІ [2]	CCE	COM	CI	\overline{RLD}	\overline{OECT}	MS	СУСЗ_МІ $I_5 - I_0$	Стек	CT
0000	****	0100	0	*	1	1	1	*	000000	0001	*
0001	****	1110	*	*	1	1	1	*	000000	0001	*
...											
000E	****	1101	0	0	1	1	0	3	101111	0001	MN
000F	****	1110	*	*	1	1	1	*	000000	****	*

Мнемонічний запис

	link 13:ct;	
0000	{push;}	\ завантаження в стек адреси початку циклу
0001	{ }	\ початок циклу
	...	
000E	{loop not rm_n;}	\ перевірка логічної умови: якщо $MN = 0$ – вихід з циклу,
		\ інакше – повернення на адресу із стеку
000F	{ }	\ продовження мікропрограми

Приклад 2.14. Розробити мікропрограму для обчислення заданої функції:

$$D = C/2 + 2A(B + 1)$$

Виконання завдання:

Алгоритм обчислення функції наведений на рис. 2.29. Цифрова діаграма стану регістрів наведена у табл. 2.28. Мікропрограма налагоджена у моделюючому комплексі COMPLEX зображена на рис. 2.30.

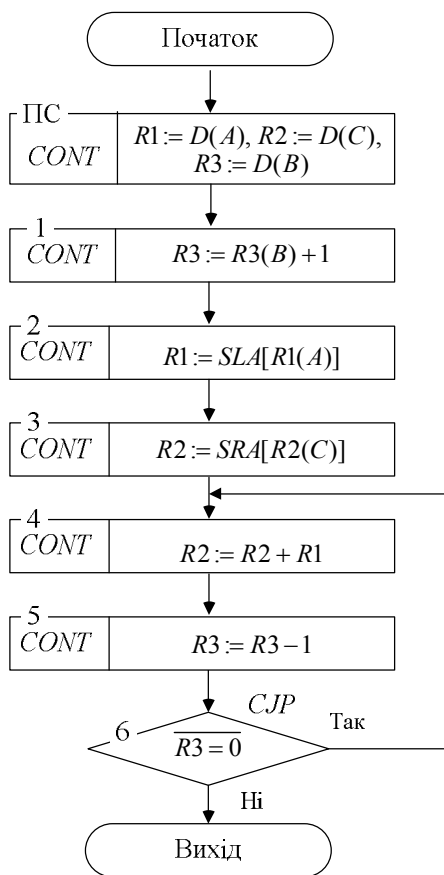


Рис. 2.29. Алгоритм обчислення функції

Таблиця 2.28. Цифрова діаграма стану регістрів

№ такту	R1 (A)	R2(C)	R3(B)	ZO	Коментарі
ПС	0005	000C	0005	0	Завантаження даних
1			0006		$R3 := R3 + 1 \ (B + 1)$
2	000A				$R2 := I[R2].0 \ (2A)$
3		0006			$R1 := 0.r[R1] \ (C/2)$
4(1) 5(1) 6(1)		0010	0005	0	$R2 := R2 + R1 \ (A+C)$ $R3 := R3 - 1 \ (B - 1)$ $ZO = 0 \ (B \neq 0)$
4(2) 5(2) 6(2)		001A	0004	0	$R2 := R2 + R1 \ (A+C)$ $R3 := R3 - 1 \ (B - 1)$ $ZO = 0 \ (B \neq 0)$
4(3) 5(3) 6(3)		0024	0003	0	$R2 := R2 + R1 \ (A+C)$ $R3 := R3 - 1 \ (B - 1)$ $ZO = 0 \ (B \neq 0)$
4(4) 5(4) 6(4)		002E	0002	0	$R2 := R2 + R1 \ (A+C)$ $R3 := R3 - 1 \ (B - 1)$ $ZO = 0 \ (B \neq 0)$
4(5) 5(5) 6(5)		0038	0001	0	$R2 := R2 + R1 \ (A+C)$ $R3 := R3 - 1 \ (B - 1)$ $ZO = 0 \ (B \neq 0)$
4(6) 5(6) 6(6)		0042	0000	1	$R2 := R2 + R1 \ (A+C)$ $R3 := R3 - 1 \ (B - 1)$ $ZO = 1 \ (B = 0)$
7					Кінець обчислення

COMPLEX

Файл

Редактор

Выполнить

Протокол

Помощь

Опции

Выход

Исходный файл :FUNC_1.PMK

Страница: 1.1

Адр	Конст.		Б О Д															
	БОД	ОЕД	В С 1						В Р 2									
			МІ ×	А	В	ОЕУ	МІ ×	С	З	Н	У	ОЕСТ	СЕМ	СЕМ	SE			
000	0005	0	011.011.101	1	1	1	00.00000.000000	0	0	0	0	0	1	1	1	1		
001	000С	0	011.011.101	2	2	1	00.00000.000000	0	0	0	0	0	1	1	1	1		
002	0005	0	011.011.101	3	3	1	00.00000.000000	0	0	0	0	0	1	1	1	1		
003	0000	1	011.000.100	3	3	1	01.00000.000000	0	0	0	0	0	1	1	1	1		
004	0000	1	111.000.100	1	1	1	00.10000.000000	0	0	0	0	0	1	1	1	0		
005	0000	1	101.000.100	2	2	1	00.00010.000000	0	0	0	0	0	1	1	1	0		
006	0000	1	011.000.001	1	2	1	00.00000.000000	0	0	0	0	0	1	1	1	1		
007	0000	1	011.001.100	3	3	1	00.00000.000000	0	0	0	0	0	1	1	1	1		
008	0006	1	011.000.100	3	3	1	00.00000.110100	0	0	0	0	0	0	1	1	1		
009	0000	1	001.000.000	0	0	1	00.00000.000000	0	0	0	0	0	1	1	1	1		

Исходный файл :FUNC_1.PMK

Страница: 1.2

Адр	БМУ и БПП												ОП, ВУ, РА								
	РАА, РАВ				ВУ4					M S	ВН1		ПАВ	ВУ				ОП		РА	
	MSA	MSB	EMA	EWB	МІ ×	CCE	COM	CI	RLD		МІ ×	EINS		EV	I	O	LCK	IA	R	W	EWB
000	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
001	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
002	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
003	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
004	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
005	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
006	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
007	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1
008	0	0	1	1	0011	0	0	1	1	1	0000	1	1	1	1	1	1	1	1	1	1
009	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1	1	1	1	1	1	1	1

Нажмите любую клавишу для окончания

Исходный файл :FUNC_1_A.ASM Страница: 1

```

/-----connection area-----
link l1:ct
/-----swap of registers area-----
accept R1:5h
accept R2:0Ch
accept R3:5h
/-----programs area-----
{add R3,R3,Z,NZ;}
{add SLL,R1,R1,Z;}
{add SRL,R2,R2,Z;}
l11 {add R2,R1,R2;}
    {sub R3,R3,Z,Z;}
    {add R3,R3,Z;cjp not Z0,l11;}
    {}

```

Рис. 2.30. Мікропрограма обчислення функції

Приклад 2.15. Розробити мікропрограму для блоку обробки даних, що реалізує заданий мікроалгоритм (рис. 2.31).

Виконання завдання:

На вихідному мікроалгоритмі (рис. 2.31) біля кожної вершини указані адреси розміщення відповідних мікрокоманд у ПМК, а також указані виконувані мікроінструкції для ІС K1804ВУ4, що забезпечують формування адрес мікрокоманд. Адреса першої вершини може бути довільною. Кодова карта з відповідними коментарями наведена у табл. 2.29. Кожній вершині мікроалгоритму відповідає рядок табл. 2.9, номер якої співпадає з номером вершини. Адреси та дані подані у шістнадцятирічній системі числення. Розряди мікрокоманди, значення яких можуть бути довільними, позначені символом «*».

У стовпці *LC* показані логічні умови, що впливають на формування адреси наступної мікрокоманди. Під час налагодження мікропрограми слід перевіряти правильність переходів, як за нульовим значенні умови, так і за одиничним.

На кодовій карті умовно не показані поля мікрокоманди, що не змінюються під час виконання мікропрограми.

Даний приклад ілюструє виконання кожної з шістнадцяти мікроінструкції, що реалізує БМУ.

Умовні інструкції у вершинах 4, 5, 6, 8, 10, 15, 24 мікроалгоритму відповідають безумовному переходу. Безумовний перехід забезпечується поданням одиничного сигналу на вхід $CCE = 1$ (табл. 2.29). Слід зазначити, що безумовний перехід можна реалізувати також шляхом підключення до входу *CC* виходу *L7* мультиплексора, що відповідає логічній одиниці. За цього у полях мікрокоманди треба встановити наступні сигнали: $MS = 7[16]$ – номер виходу мультиплексора, $CCE = 0$, $COM = 1$ (або $COM = 0$, якщо умова має бути інвертована) та $\overline{OECT} = 0$. Такий спосіб застосований під час кодування мікрокоманд на рис. 2.32.

Мікропрограма у машинних кодах та із застосуванням символічного мікроасемлера, налагоджена у моделюючому комплексі *COMPLEX*, зображена на рис. 2.32, *а, б* відповідно.

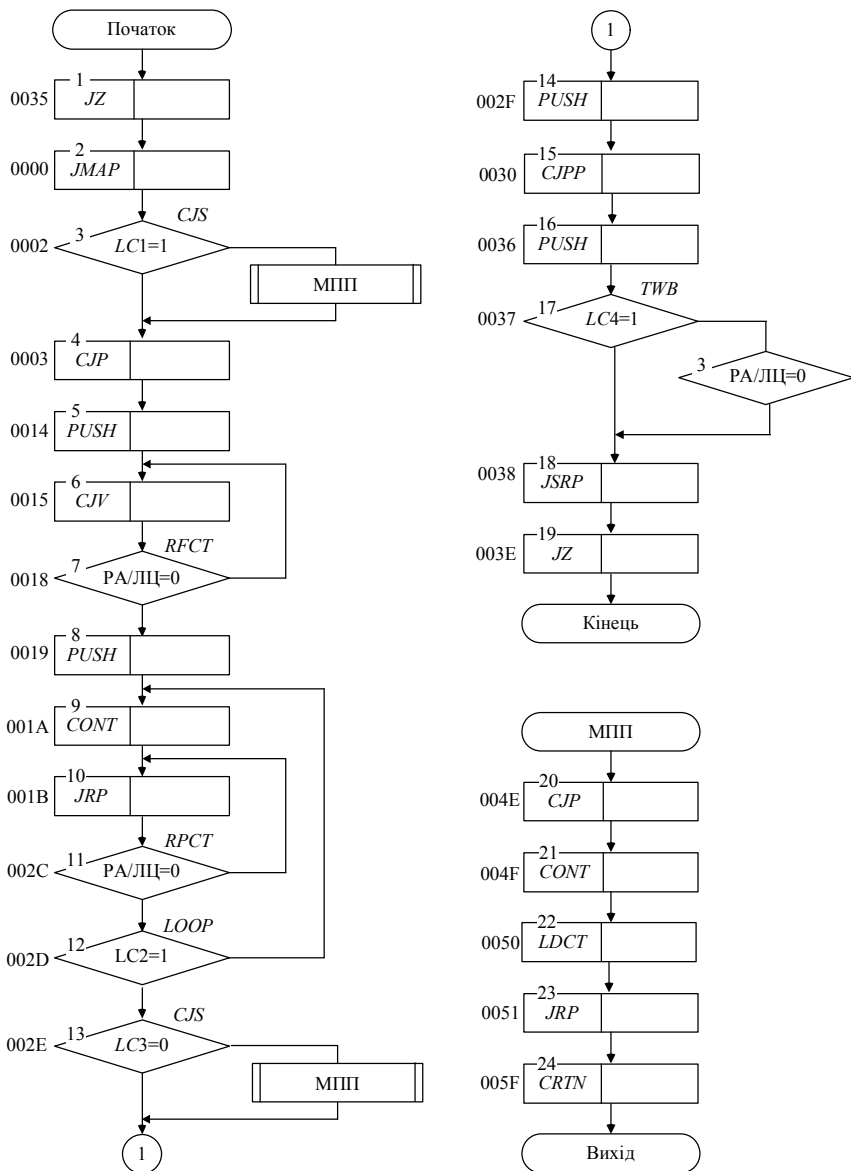


Рис. 2.31. Вихідний мікроалгоритм

Таблиця 2.29. Кодова карта

Мікрокоманда											Коментарі		
№ пп.	Мнс- моніка	Адреса [16]	Константа БОД		ВУ4 (БМУ)								
			D [16]	\overline{OED}	БМУ_МІ [2]	CCE	COM	CI	\overline{RLD}	MS	$P(M,V)$	ПС: СТЕК	LC
1	JS	035	0000	1	0000	*	*	1	1	0	*	0:*	*
2	JMAP	000	0000	1	0010	*	*	1	1	*	0002(M)		*
3	CJS	002	004E	1	0001	0	0	1	1	1	004E	1:003	$\overline{L1} = 0$
4	CJP	003	0014	1	0011	1	1	1	1	*	0014		1(NZ)
5	PUSH	014	0003	1	0100	1	1	1	0	*	0003	1:015	1(NZ)
6	CJV	015	0000	1	0110	1	1	1	1	*	0018(V)		1(NZ)
7	RFCT	018	0000	1	1000	*	1	1	1	*	*	0:*	PA/ЛЦ=0
8	PUSH	019	0002	1	0100	1	1	1	1	*	0002	1:01A	1(nz)
9	CONT	01A	0000	1	1110	*	1	1	1	*	*		*
10	JRP	01B	002C	1	0111	1	1	1	1	*	002C		1(NZ)
11	RPCT	02C	001B	1	1001	*	1	1	1	*	001B		PA/ЛЦ=0
12	LOOP	02D	0000	1	1101	0	1	1	1	2	*	0:*	L2=1

Продовження таблиці 2.29.

Мікрокоманда											Коментарі		
№ пп.	Мне- моніка	Адреса [16]	Константа БОД		ВУ4 (БМУ)								
			D [16]	\overline{OED}	БМУ_МІ [2]	CCE	COM	CI	\overline{RLD}	MS	$P(M,V)$	ПС: СТЕК	LC
13	CJS	02E	04E	1	0001	0	0	1	1	3	04E	1:02F	$\overline{L3} = 0$
14	PUSH	02F	0000	1	0100	*	1	1	0	*	0000	1:030	*
15	CJPP	030	0036	1	1011	1	1	1	1	*	0036	0:*	1(NZ)
16	PUSH	036	0005	1	0100	0	1	1	0	7	0005	1:037	$L7$
17	TWB	037	003E	1	1111	0	0	1	1	4	003E	0:*	$\overline{L4} = 0$
18	JSRP	038	003E	1	0101	0	1	1	1	7	003E	1:039	$L7$
19	JZ	038	0000	1	0000	*	*	1	1	0	*	0:*	*
20	CJP	04E	005F	1	0011	0	0	1	1	7	005F		$\overline{1(NZ)}$
21	CONT	04F	0000	1	1110	0	1	1	1	*	*		*
22	LDCT	050	005F	1	1100	1	1	1	0	*	005F		*
23	JRP	051	004F	1	0111	0	1	1	1	0	004F		0(Z)
24	CRTN	05F	0000	1	0111	1	1	1	1	*	*	0:*	1(NZ)

Адр	Конст.		Б О Д					БМУ						M	S
	БОД		В С 1					ВУ4							
	БМУ	ОЕД	MIκ	A	B	OEY	OECT	MIκ	CCE	COM	CI	RLD			
035	0000	1	001.000.000	0	0	1	1	0000	0	1	1	1	0		
000	0000	1	001.011.100	1	0	0	1	0010	0	1	1	1	0		
001	0000	1	001.000.000	0	0	1	1	1110	0	1	1	1	0		
002	004E	1	001.000.000	0	0	1	1	0001	0	0	1	1	1		
003	0014	1	001.000.000	0	0	1	0	0011	0	1	1	1	7		
014	0003	0	001.000.000	0	0	1	0	0100	0	1	1	1	7		
015	0000	1	001.000.000	0	0	1	0	0110	0	1	1	1	7		
018	0000	1	001.000.000	0	0	1	1	1000	0	1	1	1	0		
019	0002	0	001.000.000	0	0	1	0	0100	0	1	1	1	7		
01A	0000	1	001.000.000	0	0	1	1	1110	0	1	1	1	0		
01B	002C	1	001.000.000	0	0	1	0	0111	0	1	1	1	7		
02C	001B	1	001.000.000	0	0	1	1	1001	0	1	1	1	0		
02D	0000	1	001.000.000	0	0	1	1	1101	0	1	1	1	2		
02E	004E	1	001.000.000	0	0	1	1	0001	0	0	1	1	3		
02F	0000	1	001.000.000	0	0	1	1	0100	0	1	1	1	0		
030	0036	1	001.000.000	0	0	1	0	1011	0	1	1	1	7		
036	0005	0	001.000.000	0	0	1	0	0100	0	1	1	1	7		
037	003E	1	001.000.000	0	0	1	1	1111	0	0	1	1	4		
038	003E	1	001.000.000	0	0	1	0	0101	0	1	1	1	7		
03E	0000	1	001.000.000	0	0	1	1	0000	0	1	1	1	0		
04E	005F	1	001.000.000	0	0	1	0	0011	0	0	1	1	7		
04F	0000	1	001.000.000	0	0	1	1	1110	0	1	1	1	0		
050	005F	1	001.000.000	0	0	1	1	1100	0	1	1	1	0		
051	0000	1	001.000.000	0	0	1	0	0111	0	1	1	1	0		
05F	0000	1	001.000.000	0	0	1	0	1010	0	1	1	1	7		

Рис. 2.32. Результати моделювання: а – мікропрограма в машинних кодах, б – мікропрограма у мнемокодах.


```

Исходный файл :PROG1.ASM  Страница: 1

    link m:12,11,10,9,8,7,6,5,4,3,2,1
    link v:z,z,z,z,z,z,z,nz,nz,z,z,z
    link 12:nz
    accept r1:0005
    org 0035h
    <jz;>
    org 0000h
    <or nil,r1,z;oev;jmap;>
    <cont;>
    <cjs not 11,mpp1;>
    <cjp nz, 111;>
    111    org 0014h
           <push nz,0003;>
           <cjv nz;>
    112    org 0018h
           <rfct;>
           <push nz,0002;>
           <cont;>
    114    <jrp nz,113;>
           org 002ch
    113    <rpct 114;>
           <loop 12;>
           <cjs not 13,mpp1;>
           <push;>
           <cjpp nz,115;>
           org 0036h
    115    <push nz,0005h;>
           <twb not 14,003eh;>
           <jsrp nz,116;>
           org 003eh
    116    <jz;>
           org 004eh
    mpp1  <cjp not nz,117;>
           <cont;>
           <ldct 005fh;>
           <jrp z,0000;>
           org 005fh
    117    <crtm nz;>

```

Продовження рисунку 2.32

Приклад 2.16. Розробити мікропрограму для блоку обробки даних, що реалізує заданий мікроалгоритм (рис. 2.33).

Виконання завдання:

Мікропрограма у машинних кодах, налагоджена у моделюючому комплексі *COMPLEX*, зображена на рис. 2.34.

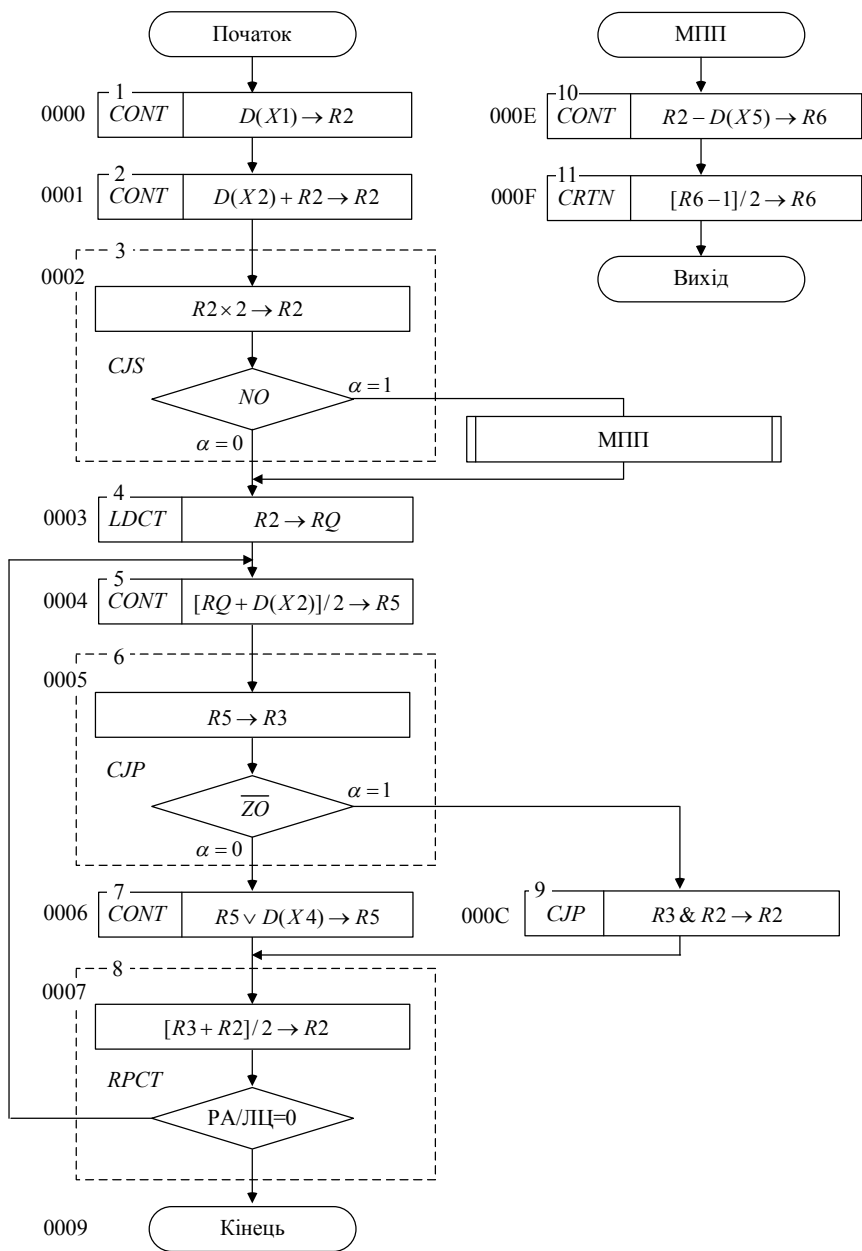


Рис. 2.33. Вихідний мікроалгоритм

```

/ мікропрограма із застосуванням символічного мікроасемблеру
/ область налагодження зв'язків
    link l2:ct;
/ область програми
0000      {or r2,r2,005h;}           /  $X1(005h) \rightarrow R2$ 
0001      {add r2,r2,0c0h,z;}       /  $X2(0C0h) + R2 \rightarrow R2$ 
0002 mll  {or sl.16 r2,r2,z;cjs no,mpp;} /  $R2 \times 2 \rightarrow R2$  ; перехід на мікропідпрограму,
                                         / якщо результат від'ємний
0003 lab3 {add rq,r2,z;ldct 6;}     /  $R2 \rightarrow RQ$  ; завантаження в РА/ЛЦ кількості
                                         / циклів
0004      {add sr.2 r5,rq,055h,z;}  /  $(RQ + X3(0C0h)) / 2 \rightarrow R5$ 
0005      {add r3,r5,z,z;cjp not zo,lab1;} / пересилання  $R5 \rightarrow R3$  ; умовний перехід на
                                         / мітку, якщо результат не дорівнює нулю
0006      {or r5,r5,0f00fh;}        /  $R5 \vee X4(0F00Fh) \rightarrow R5$ 
0007 lab2 {add sr.2 r2,r3,r2,z;rpct,lab3;} /  $[R3 + R2] / 2 \rightarrow R2$  ; інкремент вмісту РА/ЛЦ
                                         / та перевірка, якщо РА/ЛЦ не дорівнює нулю,
                                         / повернення на початок циклу
0009      {cjp nz, lend}           / безумовний перехід на кінець мікропрограми
000C lab1 {and r2,r2,r3;cjp nz,lab2;} /  $R3 \& R2 \rightarrow R2$  , безумовний перехід на мітку
000E mpp  {sub r6,r2,0013h}        /  $R2 - X5(0013h) \rightarrow R6$ 
000F      {sub r6,r6,z,nz;crtn;}    /  $[R6 - 1] / 2 \rightarrow R6$  , безумовний вихід із
                                         / мікропідпрограми
0010 lend {}                       / кінець

```

COMPLEX														
Файл Редактор Выполнить Протокол Помощь Опции Выход														
Исходный файл :UUUUAU.PMK Страница: 1.1														
Адр	Конст.			Б О Д										
	БОД		ОЕД	В С 1					В Р 2					
	БМУ			МІ*	А	В	ОЕУ		МІ*	С	З	Н	У	Е
000	0005	0		011.000.111	0	2	1		00.00000.000000	0	0	0	0	1
001	02C0	0		101.000.101	2	2	1		00.00000.000000	0	0	0	0	1
002	000E	0		111.000.001	2	2	1		00.10000.111110	1	0	0	0	0
003	0006	0		000.000.100	2	0	1		00.00000.000000	0	0	0	0	1
004	0055	0		101.000.110	0	5	1		00.00010.000000	0	0	0	0	1
005	000C	0		011.000.100	5	3	1		00.00000.110101	0	0	0	0	1
006	F00F	0		011.011.101	5	5	1		00.00000.000000	0	0	0	0	1
007	0000	1		101.000.001	3	2	1		00.00010.000000	0	0	0	0	1
008	0004	1		001.000.011	0	0	1		00.00000.000000	0	0	0	0	1
009	0000	1		001.000.000	0	0	1		00.00000.000000	0	0	0	0	1
00A	0000	1		001.000.000	0	0	1		00.00000.000000	0	0	0	0	1
00B	0000	1		001.000.000	0	0	1		00.00000.000000	0	0	0	0	1
00C	0007	0		011.100.000	5	5	1		00.00000.000000	0	0	0	0	1
00D	0000	1		001.000.000	0	0	1		00.00000.000000	0	0	0	0	1
00E	0013	0		011.001.101	2	6	1		01.00000.000000	0	0	0	0	1
00F	0000	1		101.001.000	0	0	1		00.00010.000000	0	0	0	0	1
010	0000	1		001.000.000	0	0	1		00.00000.000000	0	0	0	0	1
Исходный файл :UUUUAU.PMK Страница: 1.2														
Адр	БМУ и БП													
	РАА, РАВ				ВУ4					ВН1		ПАВ		
	MSA	MSB	EWB	EWB	MI*	CCE	COM	CI	RLD	M	MI*	EINS	EU	PrA
000	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1
001	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1
002	0	0	1	1	0001	0	0	1	1	2	0000	1	1	1
003	0	0	1	1	1100	1	1	1	0	0	0000	1	1	1
004	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1
005	0	0	1	1	0011	0	0	1	1	2	0000	1	1	1
006	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1
007	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1
008	0	0	1	1	1001	1	1	1	1	0	0000	1	1	1
009	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1
00A	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1
00B	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1
00C	0	0	1	1	0011	1	1	1	1	0	0000	1	1	1
00D	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1
00E	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1
00F	0	0	1	1	1010	1	1	1	1	0	0000	1	1	1
010	0	0	1	1	1110	1	1	1	1	0	0000	1	1	1
Нажмите любую клавишу для окончания														

Рис. 2.34. Мікропрограма в машинних кодах

ВКАЗІВКИ ДО ВИКОРИСТАННЯ МІКРОАСЕМБЛЕРУ

А.1. Трансляція мікропрограми

Мнемонічний двопрохідний мікроасемблер призначений для розробки мікропрограм.

Результатом роботи мікроасемблера є файл даних з розширенням «*.pmk», який є вихідним для програмного емулятора системи на рис. А.1.

Процес трансляції вихідного файлу здійснюється за два проходи. Під час першого проходу відбувається визначення обсягу вихідного файлу, формуються таблиці міток і відповідностей, а також проводиться попередній синтаксичний аналіз. Під час другого проходу безпосередньо формуються коди мікрокоманд. У випадку виявлення синтаксичної або семантичної помилки в вихідному тексті процес трансляції припиняється з видачею повідомлення про характер помилки і рядка тексту, на якому вона була виявлена.

Вихідним файлом для мікроасемблера є текстовий файл в кодах ASCII з розширенням «*.asm». Розходження між заголовними і малими літерами мікроасемблером не сприймаються. Між окремими мнемоніками може бути будь-яке число службових символів, наприклад, пробіл, табуляція, повернення каретки, переклад рядка і таке інше.

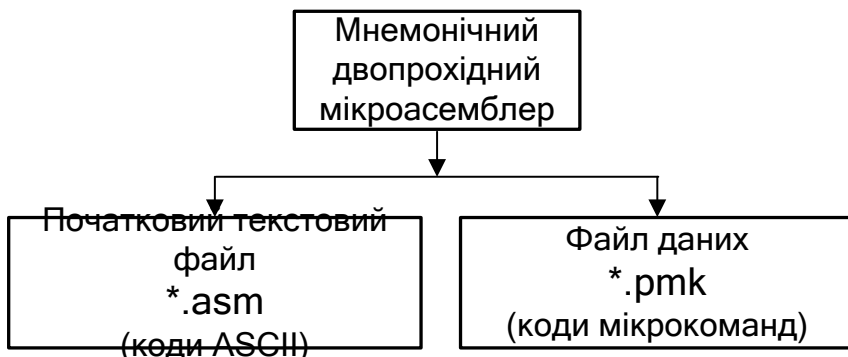


Рис. А.1. Процес трансляції мікропрограми

Строге дотримання правил написання мікропрограми, акуратність в наборі тексту прискорюють трансляцію і налагодження. Більшість помилок виникає насамперед через небалий стиль написання і неточне знання самого об'єкта розробки.

Приклад текстового файлу (в кодах ASCII), який є початковим для мнемонічного двопрорідного мікроасемблера:

link l1:ct	\ завдання зв'язків
accept r1:00ffh	\ завантаження регістрів
org 100h	\ розміщення МП у ПМК
macro inc reg:{add reg,reg,z,nz;}	\ створення макрокоманди
	\ інкременту
accept ra:12	\ завантаження номеру
	\ регістру АЛП у R4
{add r10,ra;}	\ R10:=R10 + R12
inc r1	\ інкремент регістру R1
equ znach:125	\ параметру znach
	\ присвоюється значення
	\ 125
{sub r1,znach,nz;}	\ R1:=R1 – ZNACH

Коментарі

Коментарі використовуються для пояснень. Ознакою початку коментарю є символ «\ ». Далі мікроасемблер ігнорує всі символи, які зустрічаються, до наступного «\» або до кінця рядка.

Наприклад:

```
{add r11,r11,r10,z;} \додати до вмісту R11 вміст R10
```

Числові константи

Числові константи застосовуються під час завдання значень операндів і адрес. Ознакою константи є цифра на початку мнемоніки.

Наприклад:

\способи завдання констант у різних системах числення	
65535	\десятькова константа
0FFFFh	\шістнадцятирічна константа
177777o	\вісімкова константа
1111111111111111%	\двійкова константа

Мітки

Мітки включають до 10 символів (букв, цифр та символів «_»), причому першим символом не повинна бути цифра. Ознакою кінця мітки служить будь-який зазначений далі розділювач: пробіл, повернення каретки, переведення рядка, табуляція. Мітка не повинна збігатися з зарезервованою мнемонікою. Приклади написання міток:

Наприклад:

\застосування міток

```
l11 {or sll r1,r1,0;}   \логічний зсув вмісту регістру вліво
    {cjp not rm_z,l11} \якщо вміст регістру не дорівнює
                        \повторюємо зсув
    {add r1,r1,r3,z}   \додавання
    {cgp nz, l11}      \безумовний перехід на мітку
```

Мнемонічний запис мікрокоманд

Арифметичні мікрокоманди, що виконуються в АЛП, записуються в вигляді

```
<мнемоніка>(<оператор_зсуву>,)(<приймач_результату>,)
    <джерело_1>,<джерело_2>,<вхідний_перенос>
```

Тут і далі в круглих дужках вказуються необов'язкові елементи конструкцій.

Запис логічних мікрокоманд відрізняється від арифметичних відсутністю операнду вхідного переносу, тому що перенос не бере участь в логічних мікроопераціях.

Мнемоніка арифметичних і логічних мікрокоманд зазначена в табл. А.1. Як приймач результату може бути зазначений кожний з регістрів R0 – R15, а також nil, коли результат в НОЗП не записується, але може бути виданий на локальну шину через БУ. Якщо приймач результату не вказується, то результат міститься на місці першого джерела операндів. Джерелами операндів можуть бути два регістри НОЗП, а також один регістр (він вказується як перше джерело операндів) в комбінації з константою, bus_d або нулем. Нуль в полі джерела операнда позначається буквою z. Регістри НОЗП можуть адресуватися непрямо. Якщо в якості джерел операндів зазначені RA та/або RB, то операнди вибираються з регістрів, коди яких записані в RA і RB. Вхідний перенос може

приймати значення 0, 1 (записується відповідно через *z i nz*), а також *rm_c i not rm_c*. Мнемоніки операторів зсуву зазначені в табл. А.2.

Таблиця А.1. Мнемоніка мікрооперацій в АЛБ

Мнемоніка	Мікрооперація в АЛБ
add	$R + S + CI$
sub	$R - S - 1 + CI$
or	$R \text{ or } S$
and	$R \text{ and } S$
nand	$\text{not}(R \text{ and } S)$
xor	$R \text{ xor } S$
nxor	$\text{not}(R \text{ xor } S)$

Таблиця А.2. Мнемоніка операторів зсуву

Мнемоніка	Найменування зсуву	Розряди МК $I_{10} - I_6$	Схема зсуву
sra	Зсув вправо арифметичний	00010	
srl	Зсув вправо логічний		
sr.9	Зсув вправо з переносом	01001	
sla	Зсув вліво арифметичний	10010	
sll	Зсув вліво логічний	10000	
sl.25	Зсув вліво з переносом	11001	

Приклади мікрокоманд:

```
{add slr,r10,r2,z;} \ R10:=0.r[R1+R10]
{xor r5,r5,r5;}      \ встановлення в нуль вмісту регістру R5
{SUB r7,r7,bus_d,nz;} \ віднімання з вмісту регістру R7
                        \ даних, що надходять з ЛПШ
{add r9,z,nz;}        \ інкремент регістру R9
```


А.2. Директиви мікроасемблера для блока обробки даних

Директиви мікроасемблера – це службові мнемоніки, які не транслюються в мікрокоманди мікропрограми.

Вони служать для задання початкових значень в регістрах, початкової адреси мікропрограми в пам'яті мікрокоманд, для налаштування окремих вузлів обчислювальної системи тощо.

асепт – директива занесення інформації в регістри БОД

Загальний вигляд директиви:

```
асепт <регiстр>: <значення>
```

де <регiстр>: R0, R1,..., R15, RQ,
POH,
RM, RN,
RA, RB.

Таким чином директива дозволяє задати <значення>:

- в будь-якому з регістрів АЛП,
- одночасно у всіх регістрах АЛП,
- значення ознак в регістрах RM та RN СУСЗ,
- значення в регістрах RA, RB обрамлення БОД.

Наприклад,

```
асепт r1:0affh      \ R1:=AFFH
асепт r1:0affh      \ R1:=AFFH
асепт rq:12o        \ RQ:=12o
асепт rm:1001%      \ RM:=1001%
асепт rb:10         \ RB:=10
```

Директива

```
асепт poh:<16 значень>
```

дає можливість задати значення в регістрах загального призначення АЛП. Після poh необхідно задати 16 значень, перше з яких завантажувється в регістр R0, останнє – в регістр R15 АЛП.

Наприклад:

```
асепт poh: 0,1,2,3,4,5,6,7,8,9,0ah,0bh,0ch,0dh,0eh,0fh
```

Наведемо приклади фрагментів мікропрограм із застосуванням директиви `accept`:

Приклад:

```
accept r2:123      \ завантажити значення 123 в R2
accept r10:375o    \ завантажити 8-кове значення 375 в R10
{sub r2,r10,nz;}    \ R2:=R2 – R10
```

Приклад:

```
accept rb:9        \ завантажити значення 9 в RB
{add rb, bus_d;}    \ виконати додавання вмісту регістру,
                   \ номер якого завантажений у RB із
                   \ значенням з локальної шини
                   \ R9:=R9+bus_d
```

link – директива приєднання регістрів RA, RB до ЛШ

Загальний вигляд директиви:

```
link <регістр>: <4 номери розрядів ЛШ>
```

де <регістр> – регістри RA або RB (рис. А.2).

Директива `link` вказує, номери розрядів 16-розрядної локальної шини які мають бути приєднані до виводів 4-розрядних регістрів RA, RB об'ємлення БОД.

Наприклад, наступні директиви приєднують виводи RA до розрядів молодшої тетради локальної шини, а RB до розрядів третьою тетради ЛШ:

```
link rb:3,2,1,0
link ra:11,10,9,8
```

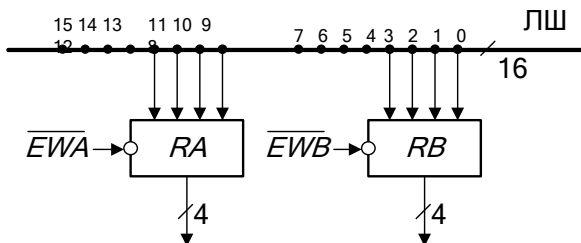


Рис. А.2. Приєднання регістрів RA, RB БОД до локальної шини

load – директива завантаження регістрів RA, RB з ЛШ

Загальний вигляд директиви:

```
load <регістр>
```

де <регістр> регістри RA або RB.

Директива load означає, що в регістр (RA або RB) буде завантажено значення, яке в даний час перебуває на ЛШ.

Приклади застосування директиви load:

```
load ra
load rb
```

Приклад:

Фрагмент мікропрограми з використанням директив accept, link, load:

accept r3:0a2d4h	\ R3:=A2D4H
link rb:7,6,5,4	\ виводи регістру RB будуть
	\ поєднані з розрядами другої
	\ тетради ЛШ
{and nil,r3,00f0h;oeu;}	\ ЛШ:=00D0 A3C4
	\ виконується мікрооперація
	\ підсумовування вмісту
	\ регістру R3 із константою,
	\ результат не фіксується, але
	\ видатись на локальну шину (oeu).
load rb	\ RB:=D C
	\ в регістр RB завантажується номер
	\ регістра АЛП – R13
{sub rb,r10, z;}	\ R13:=R13 – R10 – 1
	\ виконується мікрооперація
	\ віднімання. Регістр RB адресує
	\ перший з операндів. Місце
	\ розташування результату задано
	\ неявно. Результат записується за
	\ місцем розташування першого
	\ операнду.

Мікрокоманди управління регістрами RM та RN

Для завантаження регістрів RM, RN СУСЗ застосовуються наступні мікрокоманди

```
{load rm, z;}      \ встановлення всіх розрядів RM в нуль
{load rm, nz;}     \ встановлення всіх розрядів RM в одиницю
{load rm, flags;}  \ завантаження всіх ознак сформованих
                  \ під час виконання мікро операції в АЛП
{load rn, flags;}  \ завантаження ознак у регістр RN
```

Одночасно з зазначеними мікрокомандами застосовуються мікрокоманди заборони запису у відповідні розряди RM, а саме `sem_c`, `sem_z`, `sem_n`, `sem_v`.

Приклад:

```
{load rm, flags; sem_v; sem_n;} \ завантаження тільки
                                \ розрядів gm_c та gm_z.
{load rm, flags; sem_c;}       \ завантаження всіх ознак
                                \ окрім gm_c.
```

org – директива розміщення виконуваного коду мікропрограми в пам'яті мікрокоманд

Загальний вигляд директиви:

```
org <мітка>
org <адреса>
```

Директива `org` розміщує виконуваний код мікропрограми в ПМК за вказаною адресою.

Приклади застосування директиви org:

```
org 20h
org start
```

Якщо мікропрограма не містить директиви `org`, вона розміщується в ПМК за адресою 000.

equ – директива задання відповідності

Загальний вигляд директиви:

```
equ <ім'я>:<значення>
```

Директива `equ` використовується для присвоєння символічним іменам, що використовуються у мікропрограмі, конкретних числових значень.

Приклади застосування директиви `equ`:

```
equ start:100
equ op1:2150
equ op2:0afh
```

macro – директива створення макрокоманд

Загальний вигляд директиви:

```
macro<ім'я><формальні_параметри>:{<мікрокоманда>;}
```

Директива `macro` дозволяє конструювати власні мнемоніки операцій (макрокоманди) і користуватися ними надалі як стандартними.

Приклади застосування директиви `macro`:

```
macro inc reg:{add reg,reg,z,nz;}
macro dec reg:{sub reg,reg,z,z;}
macro mov reg1,reg2:{or reg1,reg2;}
```

Приклад:

```
inc r2           \R2:=R2+1
mov r10,r6       \R10:=R6
dec rq           \RQ:=RQ-1
```

Ім'я макрокоманди надалі стає для транслятора звичайною стандартною мнемонікою. В якості імен формальних параметрів макроса не можуть бути застосовані зарезервовані мнемоніки. У мікропрограмі макрокоманда задається своїм власним ім'ям (мнемонікою) та реальними операндами, в тому ж самому порядку, в якому вони вказані в макросі.

include – директива приєднання файлу

Загальний вигляд директиви:

```
include <имя_файла>
```

Файл, що указується в директиві повинен знаходитись в одному і тому самому каталозі, що транслюємий.

Приклад:

```
include macro.lib
include routine
```

А.3. Директиви мікроасемблера для блока мікропрограмного управління

link – директива встановлення відповідності між входами МУ та логічними умовами

Якщо в системі мікрокоманд схеми формування адреси мікрокоманди ФАМ, як умови використовуються сигнали на входах мультіплексора умов МУ – L1, L2, ..., L6 (або not L1, not L2, ..., not L6), то сигнал умови необхідно зв'язати з одним із входів зазначеного мультіплексора (рис. 1) за допомогою директиви link. Під час аналізу Li (де i = 1, 6) як логічної умови в структурі мікрокоманди у частині призначеній для управління ФАМ поле MS буде містити двійковий код відповідний до номеру входу умови – i.

Загальний вигляд директиви:

```
link <ім'я_входу>:<умова>
```

де <ім'я входу> відповідає L1, ..., L6.

Приклади застосування директиви link:

```
link l2:rdm
link l3:no
link l4:ct
```

До входів L1, ..., L6 МУ можна приєднувати наступні управляючі сигнали:

```
zo, co, no, vo
rm_z, rm_c, rm_n, rm_v
rn_z, rn_c, rn_n, rn_v
ct
rdm, rdd
int
irq0, ..., irq7
```

Завдання відповідності між входами МУ та умовами за допомогою директиви `link` може здійснюватись як окремо для кожного входу МУ, так і для всіх входів одночасно за допомогою наступної директиви

```
link l:<шість_умов>)
```

Приклад:

```
link l:ct,int,irq0,irq2,irq5,irq7
```

За цього до входів мультиплексора умов приєднуються відповідні управляючі сигнали L1:=CT, L2:=INT, L3:=IRQ0, L3:=IRQ2, L3:=IRQ5, L6:=IRQ7.

accept – директива встановлення схеми ФАМ в початковий стан

Директива `accept` дозволяє встановити в початковий стан наступні вузли ФАМ:

<code>accept sp:<значення></code>	– покажчик стека
<code>accept stack:< значення ></code>	– комірки стека
<code>accept rac:< значення ></code>	– регістр адреси / лічильник циклів (РА/ЛЦ)
<code>accept pcmk:< значення ></code>	– лічильник мікрокоманд

Приклад:

<code>accept sp:003</code>	\встановлення покажчика стека у значення 3
<code>accept stack[2]:0afdh</code>	\записати в комірку 2 стека значення AFDH
<code>accept stack:0a00h,0b00h,0c00h,2d0h,0d00h</code>	\5 значень заносяться в комірки стека
	\комірка за коміркою розпочинаючи з верхівки стека
<code>accept rac:5</code>	\в РА/ЛЦ записати значення 5
<code>accept rcmk:10</code>	\початковий стан ЛМК – 10.

link m – директива присднання Буфера М до ЛШ

Дванадцять розрядний Буфер *М* поєднує 16-розрядну локальну шину та 12-розрядну шину адреси розгалуження (рис. А.3). Інформацію з ЛШ на ШАР можна подавати у довільному порядку послідовності бітів.

Директива *link m* задає відповідність між розрядами ЛШ та розрядами Буфера *М*.

Наприклад наступна директива

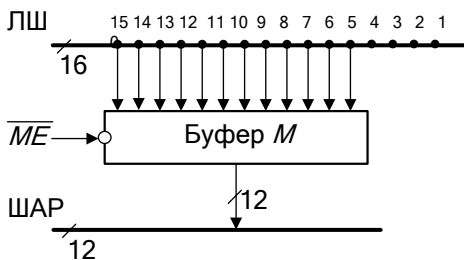
link m:15,14,13,12,11,10,9,8,7,6,5,4

з'єднує дванадцять старших розрядів ЛШ із входами Буфера *М* (рис. А.3, а).

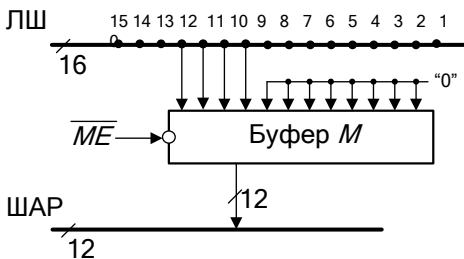
Директива

link m: 12,11,10,9,z,z,z,z,z,z,z,z,z

з'єднує 12, 11, 10, 9 розряди ЛШ із старшими чотирма входами Буфера *М*, інші розряди Буфера *М* завантажаться нулями (рис. А.3, б).



а



б

Рис. А.3. Поєднання локальної шини з Буфером *М*:
а, б – різні варіанти поєднання

link v – директива налагоджування Буфера V

Директива `link v` задає відповідність між управляючими сигналами, що поступають на вхід Буфера V та розрядами Буфера V.

На основі цих сигналів під час виконання мікрокоманди

```
{c jv cond;}
```

формується початкова адреса мікропрограми в ПМК, яка саме через Буфер V надходить в ФАМ.

На вхід Буфера V (рис. А.4) подаються наступні сигнали

<code>irq0, ..., irq7</code>	– запити на переривання від зовнішніх пристроїв,
<code>ct</code>	– сигнал логічної умови,
<code>int</code>	– сигнал вимоги загального переривання,
<code>rdm, rdd</code>	– сигнали готовності пам'яті та зовнішнього пристрою відповідно (у вигляді мнемонік ці сигнали позначаються без інверсії),
<code>z, nz</code>	– сигнали “0” та “1”.

Наприклад наступна директива

`link v:z,nz,irq0,irq1,irq2,irq3,irq4,irq5,irq6,irq7,ct,int`
під'єднає зазначені управляючі сигнали до дванадцяти входів Буфера V (рис А.4).

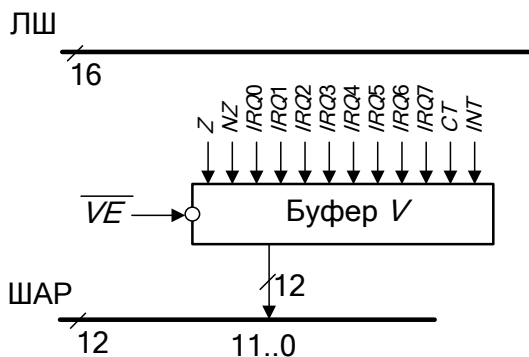


Рис А.4. Приєднання Буфера V до шини адреси розгалуження

А.4. Директиви роботи із пам'яттю та зовнішніми пристроями

dw - директива завдання значень комірок пам'яті має вигляд

dw <адреса>:<значення>

Приклад:

dw 12:0ffh

dw 03Fh:15

Мікрокоманди управління зовнішніми пристроями

Мнемоніка мікрокоманд управління зовнішніми пристроями, пам'яттю, регістром адреси і буфером БУ збігається з найменуванням відповідного управляючого сигналу:

r – мікрокоманда читання з пам'яті;

w – мікрокоманда запису в пам'ять;

i – мікрокоманда вводу даних із зовнішнього пристрою;

o – мікрокоманда виводу в зовнішній пристрій;

ewh, ewl – мікрокоманди запису відповідно в старші і молодші розряди регістра адреси;

oeu – мікрокоманда видачі результату Y з АЛБ на локальну шину.

Можна задати такі характеристики зовнішніх пристроїв (dev):

– тип пристрою (in - введення, out – виведення);

– адреса регістра стану (РС) в межах 64К (max 0FFFFh);

– адреса регістра даних (РД) в межах 64К (max 0FFFFh);

– затримка в тактах формування сигналу rdd (max 0FFFFh);

– затримка в тактах установки біта «Готовність» в регістрі стану (max 0FFFFh).

Приклад:

accept dev[2]:in, 30h, 32h, 3, 114

де in – пристрій вводу даних;

30h - адреса РС;

32h – адреса РД;

3 – затримка сигналу RDM в тактах;

114 – затримка установки біта готовності в РС після звертання до РД;

Для пристроїв введення можна задавати внутрішній буфер даних `dev_buf`, обсягом до 16 слів.

Приклад:

```
accept dev_buf[2]:1234h,5678h,89abh,0eeeh
```

Зазначені після символу «:» дані вводяться в процесор один за одним під час кожного звертання до РД даного пристрою введення.

Директива `accept` дозволяє задати швидкодію пам'яті за допомогою змінної `rdm_delay`.

Приклад:

```
accept rdm_delay: 3
```

В даному випадку сигнал RDM буде формуватися з затримкою на 3 такти після видачі на шину керування сигналу R або W.

Для опису конфігурації зв'язків між компонентами системи використовується директива `link`.

Для установки відповідності входів L1, L2, L3 БМК і логічних умов використовується директива

```
LINK <ім'я_входу>:<умова>
```

Приклад:

Для забезпечення зв'язків БМУ з пам'яттю, пристроями вводу/виводу та блоком обробки даних необхідно записати:

```
link L1:rdm
link L2:rdd
link L3:ct
```

Підключення двадцятирозрядного регістру адреси РАД до шістнадцятирозрядної ЛШ описується директивою:

```
link ewh : <номер_розряду>
```

Номер розряду РАД[19..0], зазначений у директиві, розділяє регістр на дві частини. Старша частина, включаючи зазначений розряд, управляється сигналом EWH, а молодша – EWL.

Приклад:

```
link ewh : 16
```

Директива набуває зв'язки так, що за сигналом EWH чотири молодших розряди з ЛШ записуються в поле РАД[19..16]. За сигналом EWL всі шістнадцять розрядів з ЛШ записуються в РАД[15..0], тобто в молодшу частину регістра.

В один момент часу в різних вузлах системи можуть виконуватися різні мікрооперації. Всі мікрокоманди, що управляють мікроопераціями, які виконуються в одному такті, записуються в операторних дужках { }, утворюючи повну мікрокоманду для даного такту роботи ЕОМ. Окремі мікрокоманди розділяються символом «;», окрім того, під час запису мікрокоманди можуть використовуватися роздільники типу «пробіл», «повернення каретки», «переведення рядка». Повна мікрокоманда може займати кілька рядків в тексті мікропрограми. За необхідності мітка записується зліва перед операторною дужкою, що відкривається.

А.5. Приклади розробки мікропрограм

Приклад:

Встановити нулі в чотирьох старших розрядах РАД і записати в молодші розряди цього регістра адресу з R7. Прийняти слово з ОП в регістр R15. Сигнал готовності RDM формується рівнем логічного нуля.

```
\Область налагодження зв'язків
    link l1:rdm
    link ewh:16          \установка зв'язків між РАД і ЛШ
\Визначення затримки формування RDM
    accept rdm_delay:2
\-----
\Область завантаження регістрів
\вихідна установка r7 (для налагодження)
    accept r7:1234h
\-----
\Область завантаження пам'яті
\Завантаження даних в ОП за адресою 1234h
    dw 1234h:070fh
\-----
\Область програми
```

```

    {cont;xor nil,r0,r0;oeu;ewh;}      \РАД[19...16]:=0
    {cont;or nil,r7,z;oeu;ewl;}      \РАД[15...0]:=r7
\Завантаження даних з ОП у регістр R15:=070fh
111 {cjp rdm,111;r;or r15,bus_d,z;}
    {}                                \кінець мікропрограми

```

Мікроінструкція БМУ cont, під час виконання якої відбувається формування адреси наступної мікрокоманди за інкрементом лічильника мікрокоманд (тобто відбувається лінійний перехід до наступної адреси), може бути безпосередньо не указана у мікрокоманді. При цьому формування наступної адреси відбувається за замовчуванням (див. розділ 2.5, табл. 2.27).

Приклад:

Підсумувати коди в регістрах R1,R2 і R15. Записати подвоєний результат в пам'ять за адресою, яка записана в регістрі, зазначеному в RA.

```

\Область налагодження зв'язків
    link l1:rdm
    link ewh:16
\Визначення затримки формування RDM
    accept rdm_delay:3
\-----
\Область завантаження регістрів
    accept ra:3
    accept r3:0004h
    accept r1:4
    accept r2:16
    accept r15:32
\-----
\Область програми
    {xor nil,r0,r0;oeu;ewh;}      \РАД[19...16]:=0
    {add r1,r1,r2,z;}              \ R1:=R1+R2
    {add sla,r1,r1,r15,z;}        \R1:=1(R1+R15).0
    {or nil,ra,z;oeu;ewl;}      \запис адреси в РАД
\запис результату в пам'ять
112 {cjp rdm,112;w;or nil,r1,z;oeu;}
    {}                                \кінець мікропрограми

```