

ЗМІСТ

1. Програмний комплекс для виконання лабораторних робіт.
2. Лабораторна робота №1. Синтез арифметико-логічних пристроїв з розподіленою логікою.
3. Лабораторна робота №2. Синтез блоків мікропрограмного управління.
4. Загальні вказівки до виконання лабораторних робіт №3 і №4.
5. Лабораторна робота №3. Перетворення даних в ЕОМ з мікропрограмним управлінням.
6. Лабораторна робота №4. Обробка інформації в ЕОМ на програмному і мікропрограмному рівнях.

1. ПРОГРАМНИЙ КОМПЛЕКС ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ 1,2

Програмний комплекс ПРОГМОЛС 2.0 (ПРОГрама МОделювання Логічних Схем) призначений для моделювання процесів у комбінаційних і послідовнісних схемах. Він дозволяє створювати і редагувати логічні схеми, здійснювати моделювання у синхронному (без урахування затримок сигналів в елементах схеми) і в асинхронному (з урахуванням затримок) режимах, а також зберігати отримані моделі. На базі цього комплексу виконуються лабораторні роботи 1 і 2. Вигляд моделі показано на рис. 1.1.

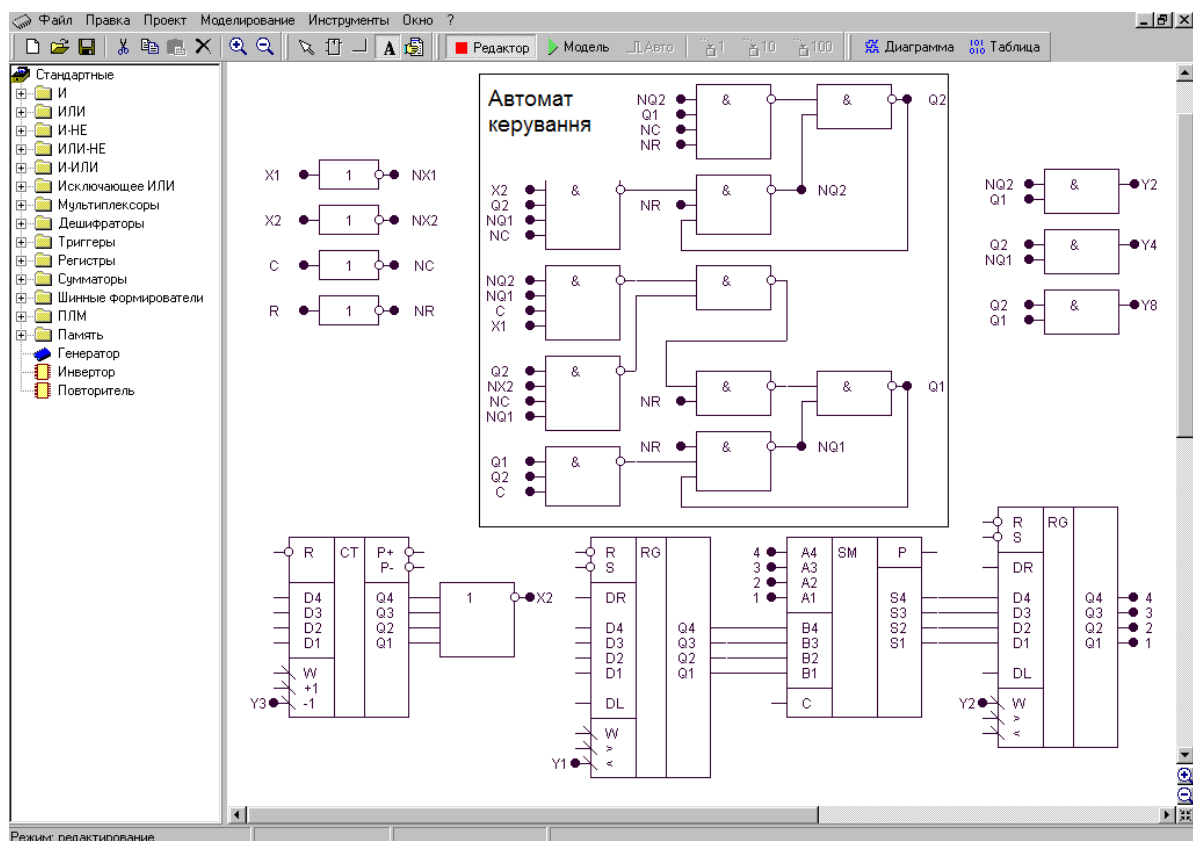


Рис. 1.1. Зображення моделі операційного пристрою з автоматом керування

Для роботи з програмою використовують систему ієрархічних меню, що містить наступні розділи:

- *файл;*
- *виправлення;*
- *проект;*
- *моделювання;*
- *інструменти;*
- *вікно;*
- *допомога.*

Комплекс включає систему підказок, що полегшує роботу в різних режимах моделювання. Побудова та елементи керування для кожного розділу пояснюються нижче.

Для дослідження схем у загальному випадку необхідно виконати послідовність дій.

1. Створити за допомогою редактора логічну схему на екрані дисплея.
2. Позначити на схемі вхідні і вихідні змінні.
3. Створити заголовок (перелічити вхідні і вихідні змінні) і сформулювати послідовність вхідних наборів в таблиці істинності.
4. Задати необхідні величини затримок сигналів для елементів схеми.
5. Встановити початковий стан схеми.
6. Перейти до режиму моделювання схеми.

2. ЛАБОРАТОРНА РОБОТА №1

СИНТЕЗ АРИФМЕТИКО-ЛОГІЧНИХ ПРИСТРОЇВ З РОЗПОДІЛЕНОЮ ЛОГІКОЮ

Мета роботи: одержати навички в проектуванні арифметико-логічних пристроїв з розподіленою логікою і автоматів управління з жорсткою логікою.

Теоретичні відомості

За структурою розрізняють АЛП з розподіленою та зосередженою логікою. Інакше їх називають відповідно АЛП із закріпленими та загальними мікроопераціями.

В АЛП першого типу апаратура для реалізації мікрооперацій розподілена між регістрами та закріплена за ними, тобто кожен регістр використовує власну логіку для виконання мікрооперацій. У пристроях другого типу всі логічні ланцюги об'єднані в арифметико-логічному блоці, а всі регістри реалізовані у вигляді надоперативного запам'ятовуючого пристрою.

АЛП з розподіленою логікою складаються з двох функціональних ча-

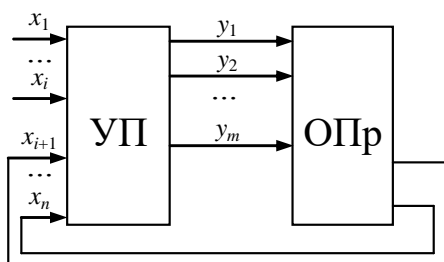


Рис. 2.1. Загальна структура АЛП

стин (рис. 2.1): управляючий пристрій (УП), що забезпечує формування всіх управляючих сигналів; операційний пристрій (ОПр), що забезпечує перетворення інформації та виконує мікрооперації над машинними

словами.

Побудова таких АЛП відбувається за наступними етапами:

1. Для кожної операції будується операційна схема та функціональний мікроалгоритм (Ф-мікроалгоритм). Рекомендується обирати такі мікроал-

горитми виконання операцій, що краще об'єднуються, тобто вимагають однакового напрямку зсувів в регістрах, однакову схему з'єднання регістрів і суматорів і таке інше.

2. Обирається розрядність регістрів, лічильників. Виконується логічне моделювання роботи ОПр, наприклад, із застосуванням діаграми стану регістрів при виконанні мікрооперацій з критичними значеннями операндів.

3. Розробляється функціональна та принципова схеми ОПр із зазначенням управляючих сигналів для кожного вузла пристрою.

4. Складається структурний мікроалгоритм (С-мікроалгоритм) виконання заданих операцій, що враховує спосіб управління мікроопераціями на вузлах ОПр.

5. Виконується синтез управляючого пристрою.

6. Складається функціональна та принципова схеми всього АЛП.

При синтезі управляючого автомата необхідно враховувати можливість формування просічек, що може привести до невірному результату обчислень в операційному пристрої.

Крім цього, необхідно враховувати можливість перетинання управляючих сигналів в часі (наступний сигнал встановлюється раніше ніж знімається попередній сигнал). Якщо сигнали подаються на один регістр, то можливий збій виконання мікрооперацій.

Для правильної роботи операційного пристрою необхідно корегування структурного мікроалгоритму роботи автомата або графа наступним чином.

1. Якщо дві сусідні операторні вершини містять управляючі сигнали для одного регістра (лічильника), то між вершинами треба включити пусту операторну вершину. Це виключить накладення сигналів.

2. Перевірка умови, що змінюється в даному такті, може перевірятися тільки в наступному такті (наприклад, змінюється стан лічильника в даному такті – перевірка тільки в наступному такті).

3. Якщо операторна вершина охоплюється петлею для циклічного виконання мікрооперації, то необхідно забезпечити перепад сигналів, що подаються на регістри (лічильники), інакше мікрооперації виконуватись не будуть. У випадку, коли це неможливо забезпечити перестановкою вершин в циклі, необхідно додати порожню вершину.

Приклад. Побудувати схему АЛП для реалізації операції множення чисел без знаків за першим способом. Формат операндів для обчислення добутку $Z=Y \times X$ є правильні дробові числа $Y = 0, y_1, y_2 \dots y_n$ та $X = 0, x_1, x_2 \dots x_n$.

Виконання завдання

Операційна схема, що реалізує перший спосіб множення, подана на рис. 2.2, де $RG1$ – регістр накопичення суми часткових добутоків, $RG2$ – регістр множника, $RG3$ – регістр множеного, $RG4$ (CT) – лічильник циклів, SM – комбінаційний суматор. DR – вхід заповнення старшого розряду при зсуві вправо. Регістри $RG1$ та $RG2$ реалізують мікрооперації зсуву, лічильник $RG4$ дозволяє формувати ознаку нуля. За нульовим вмістом регістру $RG4$ результат обчислення є сформованим в регістрах $RG1$ та $RG2$.

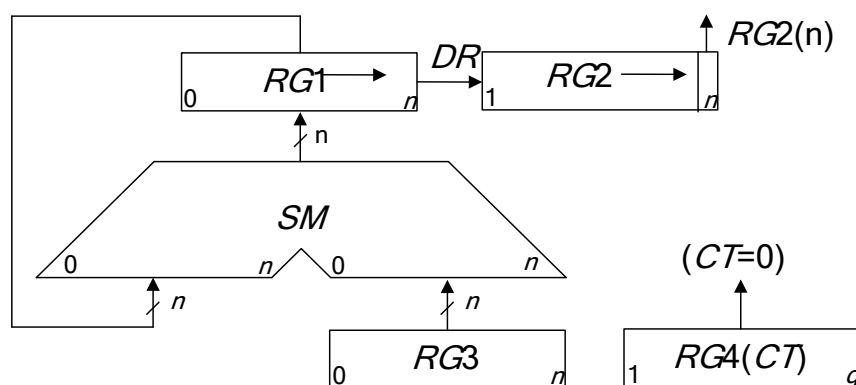


Рис. 2.2. Операційна схема множення

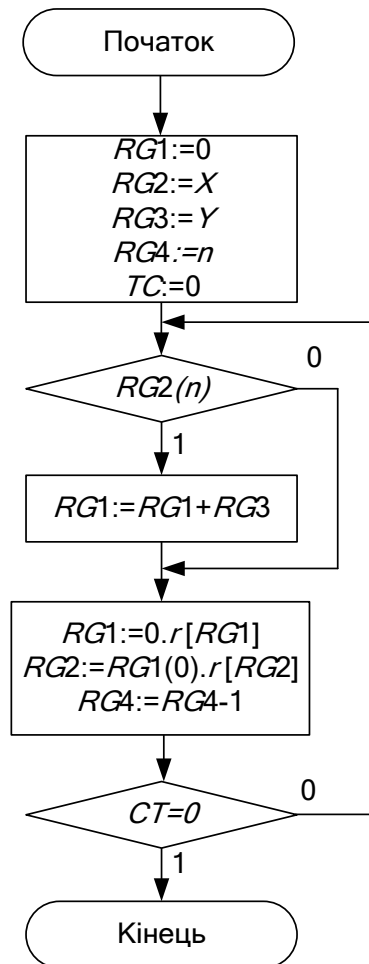


Рис. 2.3. Ф-мікроалгоритм множення чисел

Для розробленої операційної схеми побудуємо Ф-мікроалгоритм (рис. 2.3), де $RG2(n)$ – значення молодшого розряду регістру $RG2$.

Логічне моделювання потактової роботи ОПр приведене в табл. 2.1

Значення операндів:

$$Y = 5_{10} = 101_2;$$

$$X = 7_{10} = 111_2;$$

$$Z = 35_{10} = 100011_2.$$

Розрядність: $n = 3, q=2$.

Таблиця 2.1. Логічне моделювання роботи ОПр

№ та- кту	RG1	RG2	RG2(n)	RG3	RG4	МО
ПС	0000	101	1	0111	11	Початковий стан
1	$\begin{array}{r} 0000 \\ +0111 \\ \hline 0111 \\ 0011 \end{array}$	110	0	0111	10	$RG1 + RG3$ $RG1 \rightarrow, RG2 \rightarrow,$ $RG4 - 1; CT \neq 0$
2	0001	111	1	0111	01	$RG1 \rightarrow, RG2 \rightarrow,$ $RG4 - 1; CT \neq 0$
3	$\begin{array}{r} 0001 \\ +0111 \\ \hline 1000 \\ 0100 \end{array}$	011	1	0111	00	$RG1 + RG3$ $RG1 \rightarrow, RG2 \rightarrow,$ $RG4 - 1; CT = 0$

На підставі операційної схеми множення та Ф-мікроалгоритму складемо перелік управляючих сигналів для всіх функціональних вузлів ОПр та побудуємо функціональну схему.

Функціональна схема ОПр зображена на рис. 2.4. Перелік управляючих сигналів наведений в табл. 2.2.

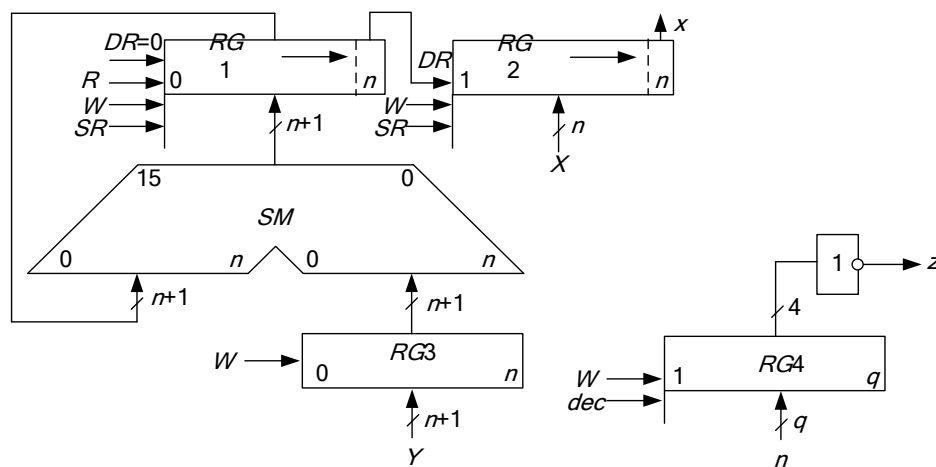


Рис. 2.4. Функціональна схема операційного пристрою

Таблиця 2.2. Таблиця управляючих сигналів

Вузол	Мікрооперація	Управляючий сигнал
RG1	Скидання в нуль	R
	Запис	W
	Зсув вправо	SR
RG2	Запис	W
	Зсув вправо	SR
	Старший розряд при зсуві вправо	DR
RG3	Запис	W
RG4	Запис	W
	Декремент лічильника	Dec

За функціональною схемою будуємо структурний мікроалгоритм (С-мікроалгоритм), що зображений на рис. 2.5. Індекс указує до якої з функціональних частин пристрою належить управляючий сигнал. Кодування сигналів управління та логічних умов наведене в табл. 2.3 – 2.4.

Закодований С-мікроалгоритм зображений на рис. 2.6, де управляючі сигнали та сигнали логічних умов відповідають рис. 2.5 та табл. 2.2 – 2.4. Сигнали, що завжди формуються разом, кодуються одним символом. Цьому символу відповідає один вихід пристрою управління.

Для управління роботою ОПр застосуємо *пристрій управління з жорсткою логікою*, який реалізуємо у вигляді цифрового автомата Мура.

Розмітка С-микроалгоритма для автомата Мура наведена на рис. 2.6. Стани автомата позначені символами a_i . Порожня вершина a_4 введена для запобігання перетинання у часі сигналів W_1 і SR_1 , що подаються на RG1. Це необхідно для правильного виконання послідовності мікрооперацій на одному регістрі. Для забезпечення перепаду сигналів управління SR_1 , SR_2 , Dec (вершину з цими сигналами охоплює петля) необхідно ввести порожню додаткову вершину a_6 . Ця вершина забезпечує також вірну оцінку ста-

ну лічильника після декременту в наступному такті, коли новий стан лічильника вже буде встановлено.

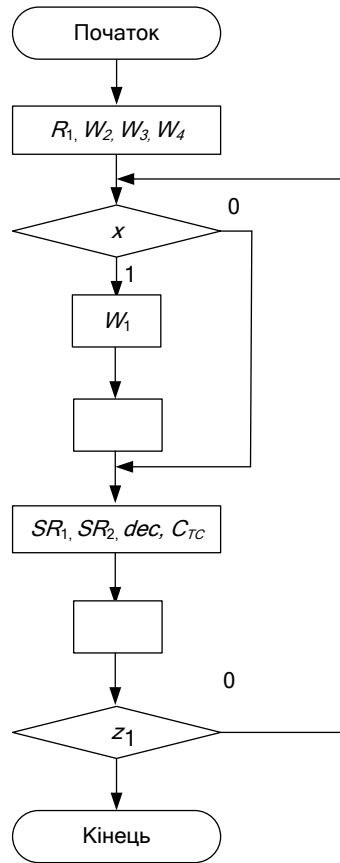


Рис. 2.5. Структурний мікроалгоритм

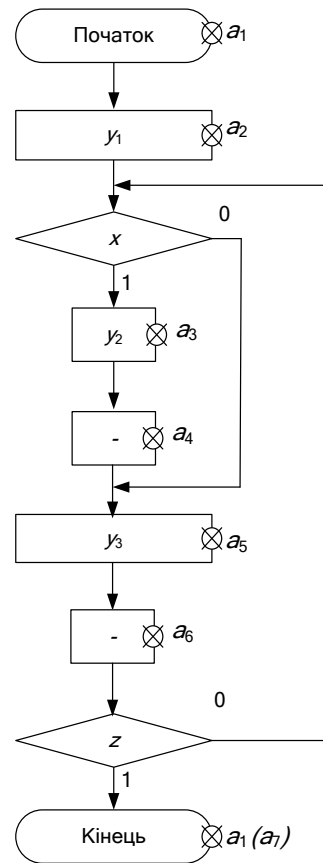


Рис. 2.6. Закодований структурний мікроалгоритм

Таблиця 2.3. Кодування сигналів управління

Управляючі сигнали	Код
R_1	y_1
W_2	
W_3	
W_4	
R_{TC}	y_2
W_1	
SR_1	y_3
SR_2	
C_{TC}	
dec	

Таблиця 2.4. Кодування логічних умов

Логічні умови	Код
Значення молодшого розряду множника	x
Нульовий вміст лічильника	z

Одержаний закодований С-мікроалгоритм є вихідним для здійснення синтезу управляючого пристрою (автомата з жорсткою логікою). На рис. 2.7 зображена узагальнена структурна схема АЛП. Управляючі сигнали з виходів пристрою управління підключаються до входів відповідних функціональних вузлів ОПр.

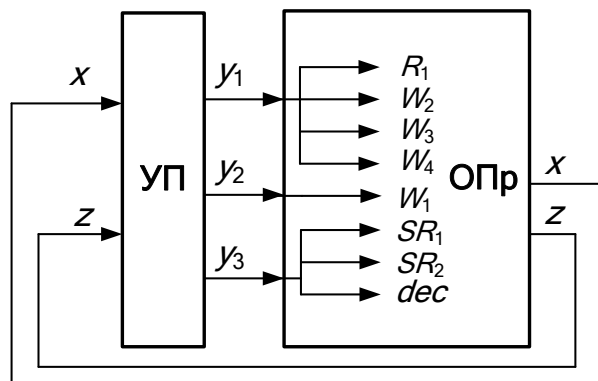


Рис. 2.7. Узагальнена структурна схема АЛП

Підготовка до лабораторного заняття

1. Варіанти завдання визначаються молодшими розрядами a_7, \dots, a_1 двійкового номера залікової книжки студента відповідно з табл. 2.7-2.9. Множення 1-м способом виконати в доповняльному коді (ДК) для чисел із знаками. Для інших функцій операнди є додатні числа.
2. Розробити операційну схему пристрою та змістовний мікроалгоритм обчислення функції (виконання операції множення) відповідно з табл. 2.7.
3. Розробити функціональну схему операційного пристрою. Для побудови схеми можна використовувати комбінаційний суматор, регістр-лічильник циклів та асинхронні регістри, що мають входи управління зсувами і занесенням інформації. На схемі повинні бути зазначені розрядність регістрів та шин.
4. Виконати логічне моделювання роботи операційного пристрою за допомогою таблиці станів регістрів та лічильника. Задати самостійно значення операндів (3-4 розряди).
5. Здійснити синтез пристрою управління, тип тригерів і управляючого автомату обрати із табл. 2.8 та 2.9. Ураховувати, що мікрооперації на регістрах виконуються за перепадом управляючих сигналів з 1 в 0.

Таблиця 2.7. Варіанти завдань

a_7	a_6	a_5	a_4	Спосіб множення
0	0	0	0	1-й, операнди зі знаками в ДК
0	0	0	1	2-й
0	0	1	0	3-й
0	0	1	1	4-й
a_7	a_6	a_5	a_4	Спосіб ділення
0	1	0	0	1-й
0	1	0	1	2-й
a_7	a_6	a_5	a_4	Функція

0	1	1	0	$F = \sqrt{X}$
0	1	1	1	$F = X^2$
1	0	0	0	$D = 2A(B+1) + 2C$
1	0	0	1	$D = A(B+1) + 2C$
1	0	1	0	$D = C + 2AB$
1	0	1	1	$D = AB + 0,5C$
1	1	0	0	$D = 2A(B+1) + C$
1	1	0	1	$D = A(B-1) + 2C$
1	1	1	0	$D = A(B+1) + 0,5C$
1	1	1	1	$D = 2A(B-1) + C$

Таблиця 2.8. Варіанти завдання

a_3	a_2	Тип тригера
0	0	JK
0	1	T
1	0	RS
1	1	D

Таблиця 2.9. Варіанти завдання

a_1	Тип автомата
0	Мили
1	Мура

Методи виконання арифметичних операцій та синтезу автоматів подано в літературі [1,2,3].

Для функцій D (табл. 2.7) операцію множення AB можна замінити додаванням A до самого себе B разів. Такий підхід дозволяє спростити схему пристрою, але з втратою його швидкодії.

Наприклад, функцію $D = AB/2 + 2C$ можна реалізувати за допомогою операційної схеми на рис. 1 за змістовним мікроалгоритмом на рис. 2.

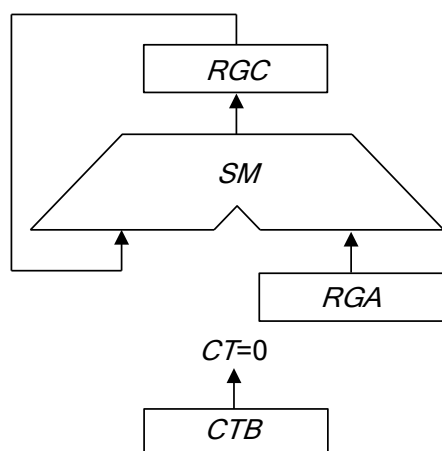


Рис. 2.8. Операційна схема пристрою для обчислення функції

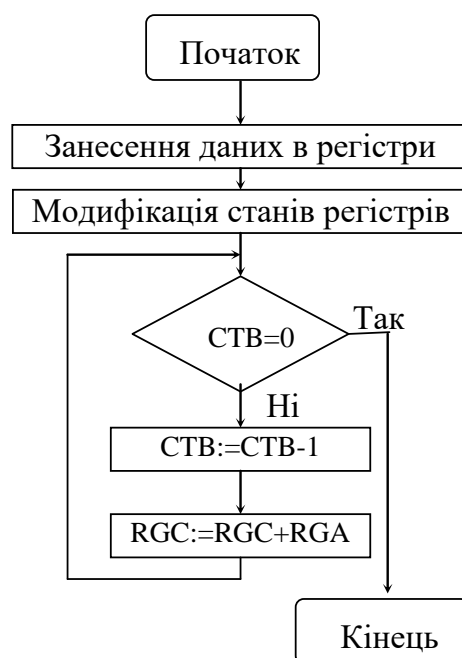


Рис. 2.9. Змістовний мікроалгоритм

У вихідному стані операнд C записаний в регістр RGC , операнд A – в регістр RGA і операнд B – у лічильник CTB . Модифікація вмісту регістрів виконується зсувами. Далі до вмісту RGC додається B разів слово, записане в RGA . Після кожного такту вміст CTB зменшується на 1. Обчислення закінчуються при виконанні умови $CTB=0$. Результат операції формується в регістрі RGC .

Порядок виконання роботи

1. В моделюючій програмі ПРОГМОЛС 2.0 (AFDK) побудувати схему операційного пристрою для множення чисел або обчислення функції та доповнити її схемою управляючого автомата. На першому етапі виходити автомата до входів операційного пристрою не підключати. Налаштувати окремо схему операційного пристрою та схему управляючого автомата. Опис програмного комплексу ПРОГМОЛС 2.0 наведений у додатку М.

2. Підключити до управляючих входів операційного пристрою виходи автомата. Зробити комплексне налагодження схеми і переконатися в правильності одержання результату.

3. Перейти до асинхронного моделювання. Дослідити зазначені викладачем часові параметри схеми.

Зміст звіту

Звіт з лабораторної роботи повинен включати короткі теоретичні відомості, необхідні для виконання лабораторної роботи; структурні та функціональні схеми; таблиці та діаграми, отримані при виконанні теоретичного завдання, а також у процесі моделювання схем; висновки за роботою.

Контрольні питання

1. Охарактеризуйте чотири основних методи множення чисел.
2. Як розрахувати розрядність вузлів операційного пристрою?
3. Визначить поняття: операція, мікроалгоритм, мікрооперація.
4. Наведіть типи арифметико-логічних пристроїв, та їх основні відмінності.

5. Охарактеризуйте основні етапи проектування арифметико-логічного пристрою з розподіленою логікою.

6. Що відображує операційна схема виконання операції?
7. Що відображує функціональна схема пристрою?
8. В чому відмінність функціонального та структурного мікроалгоритмів?

9. Напишіть вирази, що визначають закони функціонування автоматів Милі та Мура.

10. Нарисуйте узагальнену структурну схему управляючого автомата.
11. Охарактеризуйте основні етапи проектування управляючого автомата.
12. Як перейти від змістовного мікроалгоритму до закодованого мікроалгоритму?
13. Як побудувати граф автомата?
14. Як здійснюється оцінка станів автомата?
15. Як визначити необхідну тривалість управляючих сигналів?
16. Від чого залежить кількість тригерів, необхідних для побудови пам'яті автомата?
17. Як скласти структурну таблицю автомата?
18. Складіть таблицю переходів для *JK*-, *RS*-, *T*- і *D*-тригерів. Наведіть їх умовне графічне позначення.
19. Коли можливе виникнення помилкових управляючих сигналів (що непередбачені графом автомата) і чим визначається їх тривалість?
20. Наведіть способи усунення короточасних помилкових управляючих сигналів.
21. Як забезпечити перепад управляючого сигналу у випадку, коли операторну вершину з цим сигналом охоплює «петля»?
22. Як визначити час переходу автомата з одного стану в інший?

Список літератури

1. Арифметичні та управляючі пристрої цифрових ЕОМ: Навчальний посібник / В.І.Жабін, І.А.Жуков, І.А.Клименко, Стиренко С.Г. – К.: БЕК +, 2008. – 176 с.

2. *Прикладана теорія цифрових автоматів: Навчальний посібник* / В.І.Жабін, І.А.Жуков, І.А.Клименко, В.В.Ткаченко. – К.: Книжкове видавництво НАУ, 2007. – 364 с.

3. *Цифровые ЭВМ. Практикум* / К.Г.Самофалов, В.И. Корнейчук, В.П. Тарасенко, В.И.Жабин – К.: Высш.шк. 1989. – 124 с.

3. ЛАБОРАТОРНА РОБОТА №2

СИНТЕЗ БЛОКІВ МІКРОПРОГРАМНОГО УПРАВЛІННЯ

Мета роботи: Дослідити засоби побудови блоків мікропрограмного управління. Одержати навички в проектуванні й налагодженні схем пристроїв управління з мікропрограмним управлінням.

Теоретичні відомості

БМУ функціонує у відповідності з *принципом мікропрограмного управління*, що полягає в наступному.

Спрощена структурна схема БМУ наведена на рис. 3.1.

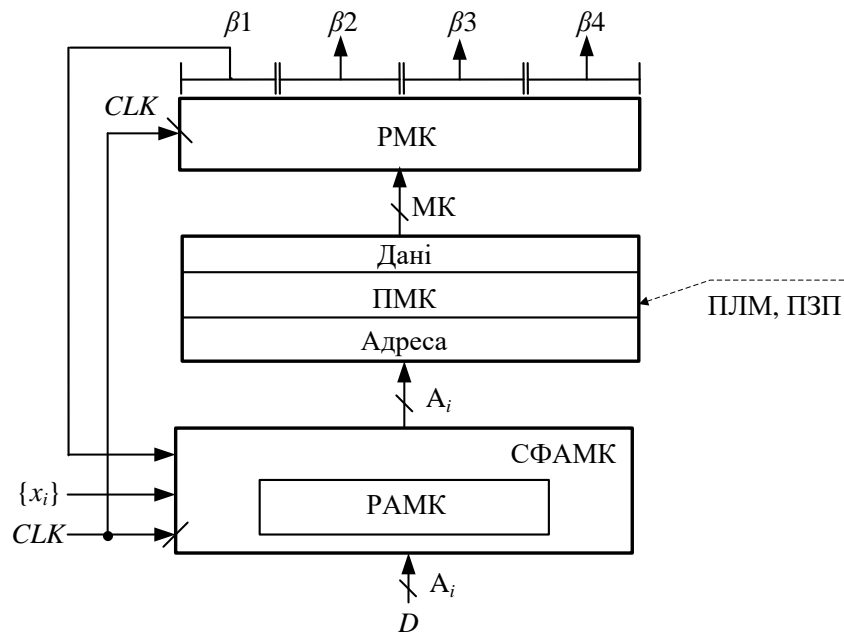


Рис. 3.1. Структурна схема БМУ

Основні функціональні частини БМУ:

- РАМК – регістр адреси МК;
- СФАМК – схема формування адреси МК;
- ПМК – пам'ять МК;
- РМК – регістр МК;

- A_i – адреса МК;
- CLK – синхросигнал;
- $\{x_i\}$ – логічні умови;
- D – вхід завдання початкової адреси мікропрограми.

МК розміщуються у пам'яті мікрокоманд. На рис. 3.2 наведений формат мікрокоманди.

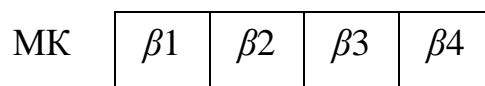


Рис. 3.2. Формат мікрокоманди

Сигнали зони $\beta 2$ управляють вузлами комп'ютера, зони $\beta 3$ – визначають тривалість цих сигналів, сигнали зони $\beta 1$ разом із логічними умовами $\{x_i\}$ поступають на вхід СФАМК і формують адресу наступної МК. За черговим сигналом CLK адреса наступної МК буде сформована у РАМК. Зона $\beta 4$ використовується для виконання допоміжних функцій, наприклад контролю апаратури.

Структурна схема БМУ з урахуванням зони затримки управляючих сигналів зображена на рис. 3.3.

У обчислювальних системах зона $\beta 4$ може складатися із сотні розрядів. Найчастіше цю зону використовують для контролю апаратури.

Схема контролю має вигляд зображений на рис. 3.4. Для контролю використовують операцію згортки (суму за модулем 2). У цьому випадку зона $\beta 4$ має довжину 1 розряд, вміст цього розряду доповнює кількість 1 у слові мікрокоманди до парної (або непарної, при контролі слова МК на непарність).

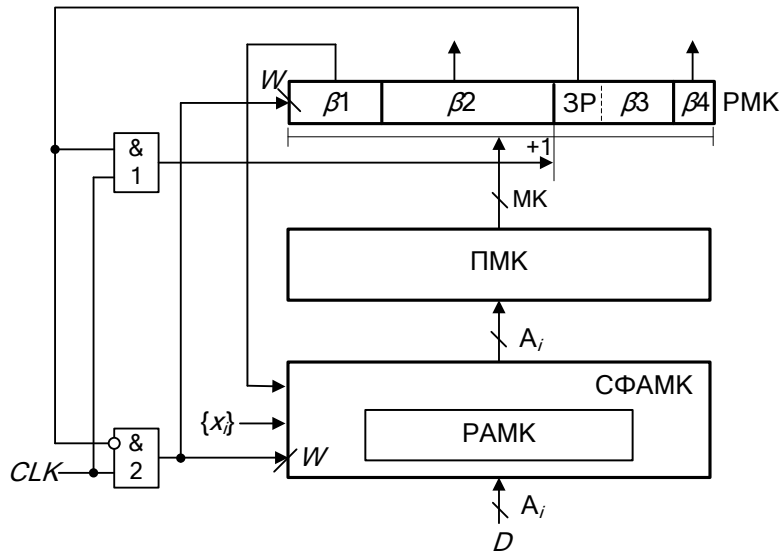


Рис. 3.3. Формування тривалості управляючих сигнала-

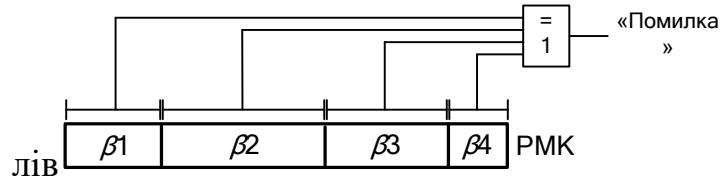


Рис. 3.4. Схема контролю слова МК на парність

БМУ з примусовою адресацією

За примусової адресації зона $\beta 1$ має наступний формат:

$$\beta 1 \begin{array}{|c|c|} \hline \overbrace{M}^q & \overbrace{K}^{n-1} \\ \hline \end{array},$$

- де M – поле управління мультиплексором;
- q – довжина поля управління мультиплексором;
- K – константа, що визначає адресу наступної мікрокоманди;
- n – розрядність адреси мікрокоманди.

Довжина поля управління мультиплексором визначається за формулою:

$$q = \lceil \log_2(k+2) \rceil$$

де k – кількість зовнішніх умов.

Поле константи K являє собою $(n-1)$ старших розрядів адреси мікрокоманди.

Формат адреси мікрокоманди має наступний вигляд:

$$A_i \begin{array}{|c|c|} \hline & n \\ \hline A_i(n-1) & \alpha \\ \hline & n-1 \\ \hline \end{array}$$

де α – визначає умову переходу, яка формується на виході мультиплексора в залежності від логічних умов X_i .

Спрощена структурна схема БМУ з примусовою адресацією зображена на рис. 3.5. На цій та подальших схемах БМУ входи для занесення початкової адреси D в РАМК умовно не показані.

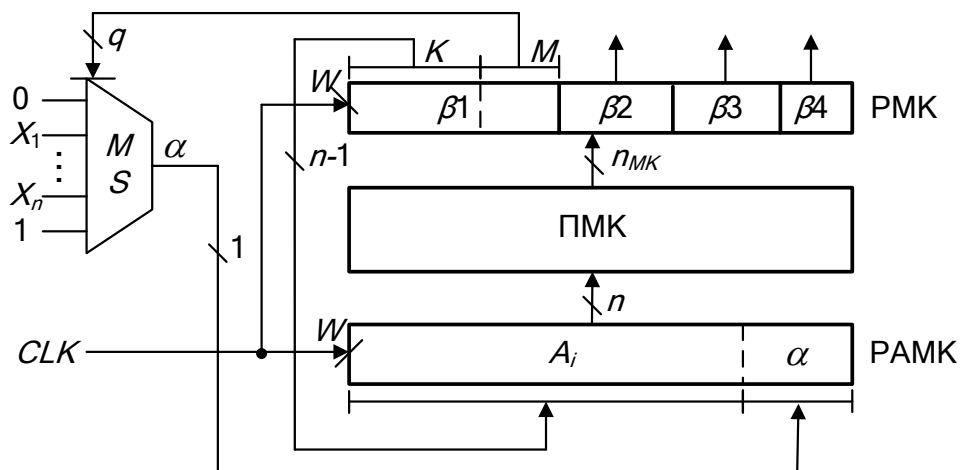


Рис. 3.5. Структурна схема БМУ з примусовою адресацією

БМУ з відотною адресацією

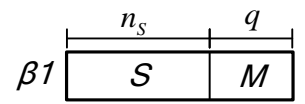
ЛОЮ:

$$A_{i+1} = A_i + S + \alpha,$$

де S – приріст адреси МК;

α — сигнал на виході мультиплексора, що залежить від логічних умов X_i .

Формат зони $\beta 1$ у загальному вигляді:



Довжину поля S визначають за виразом:

$$n_S = \lceil \log_2 N \rceil + 1,$$

де N – максимальний приріст, додатковий знаковий розряд додається для визначення напрямку переходу (зменшення або збільшення адреси).

Структурна схема БМУ наведена на рис. 3.6.

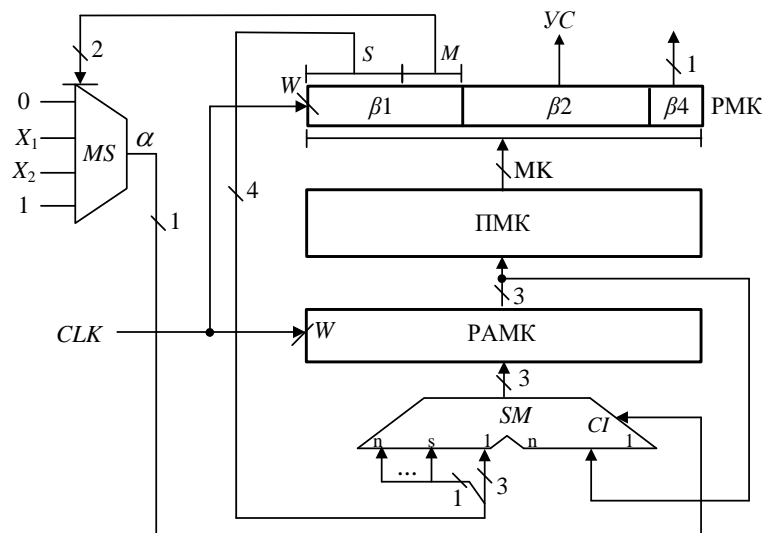


Рис. 3.6. Структурна схема БМУ з відносною адресацією

Приклад . Побудувати структурну схему БМУ і карту пам'яті мікропрограм для мікроалгоритму виконання операції множення. Мікроалгоритм по-

винен забезпечувати управління арифметико-логічним пристроєм із розподіленою логікою.

Вихідні дані:

- Спосіб адресації мікрокоманд – примусовий;
- Структура ПМК – лінійна;
- Ємність ПМК – 16 слів;
- Тривалість мікрооперації підсумовування – 4 такти;
- Початкова адреса мікропрограми – 0007h;
- Виконати перевірку слова МК на непарність;
- Розрядність операндів – 16 розрядів;
- Розрядність регістрів та суматорів – 8 розрядів.

Виконання завдання

Структурна схема пристрою для виконання операції множення першим способом з урахуванням елементної бази наведена на рис. 3.7. Мікроалгоритм управління роботою пристрою наведений на рис. 3.8. Змістовний МА наведений на рис. 3.9.

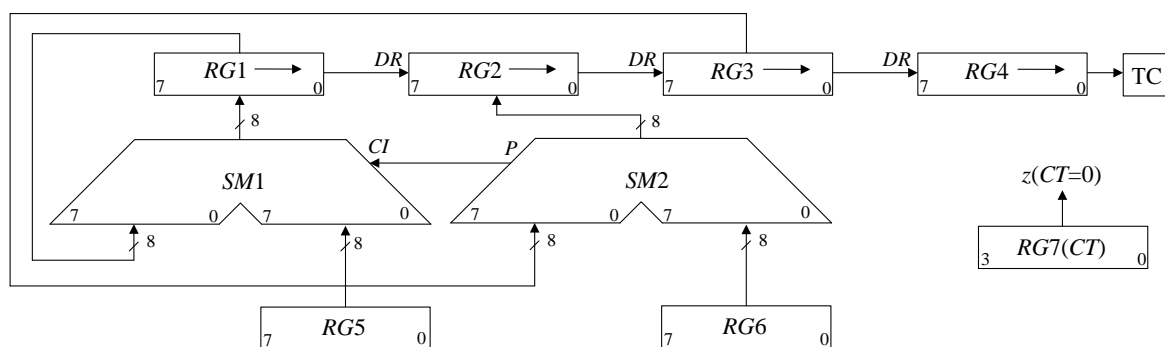


Рис. 3.7. Структурна схема пристрою множення

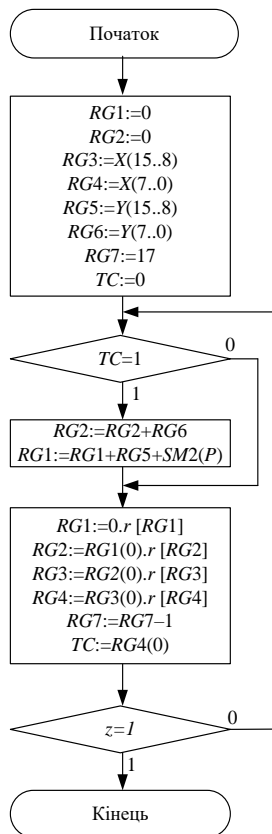


Рис. 3.8. Змістовний мікроалгоритм

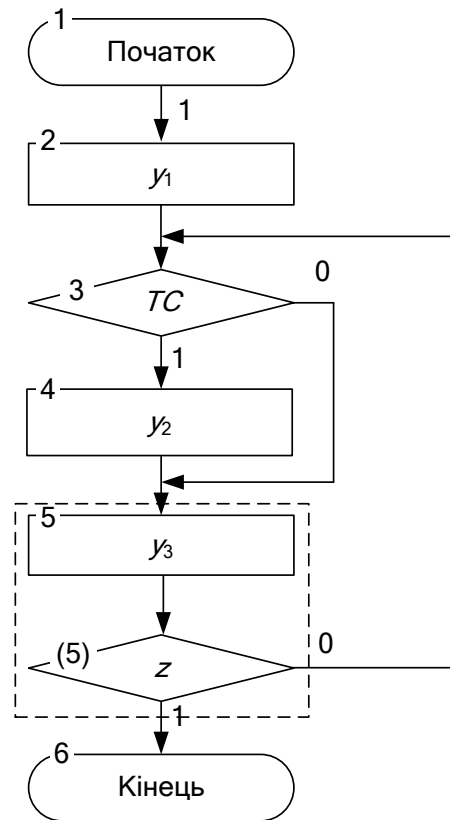


Рис. 3.9. Закодований алгоритм управління пристроєм множення

Визначимо формат зони β_1 :

$$n_a = \lceil \log_2 16 \rceil = 4; \quad n_K = 3;$$

$$n_M = \lceil \log_2 4 \rceil = 2; \quad n_{\beta_1} = 5.$$

Визначимо спосіб управління мультиплексором (табл. 3.11).

Таблиця 3.1. Кодування поля M

$m_2 \ m_1$	УС
00	0
01	TC
10	z
11	1

Визначимо формат зони $\beta 2$. Для максимального способу кодування управляючих сигналів розрахуємо розрядність коду дешифратора за виразом (3.2):

$$n_{\beta 2} = \lceil \log_2 4 \rceil = 2.$$

Наведемо кодування сигналів у зоні $\beta 2$ (табл. 3.12).

Таблиця 3.2. Кодування сигналів

$\alpha_2 \alpha_1$	УС
00	—
01	y_1
10	y_2
11	y_3

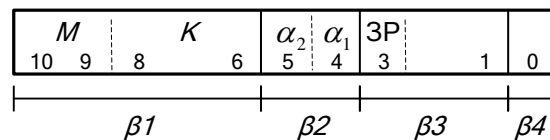
За виразом (3.3) розрахуємо довжину зони $\beta 3$:

$$\Delta t_{\max} = 3;$$

$$n_{\beta 3} = \lceil \log_2 3 \rceil + 1 = 3.$$

Для перевірки на парність у зоні $\beta 4$ необхідно виділити один розряд.

Отримаємо наступний формат мікрокоманди ($n_{\text{МК}} = 10$):



Розміщуємо мікрокоманди в пам'яті мікрокоманд (рис. 3.10).

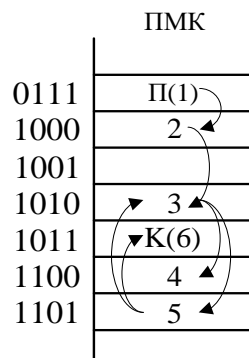


Рис. 3.10. Розміщення мікрокоманд в ПМК

Карта програмування БМУ наведена у табл. 3.3.

Таблиця 3.3. Карта програмування БМУ

№ МК	Адреса	$\beta 1$		$\beta 2$	$\beta 3$		$\beta 4$
		K	M	$\alpha_2 \alpha_1$	ЗР		
П(1)	0111	100	00	00	0	00	0
2	1000	101	00	01	0	00	0
3	1010	110	01	00	0	00	1
4	1100	110	11	10	1	01	1
5	1101	101	10	11	0	00	0
К(6)	1011	101	11	00	0	00	1

Структурна схема БМУ із лінійною ПМК та примусовим способом адресації мікрокоманд зображена на рис. 3.11.

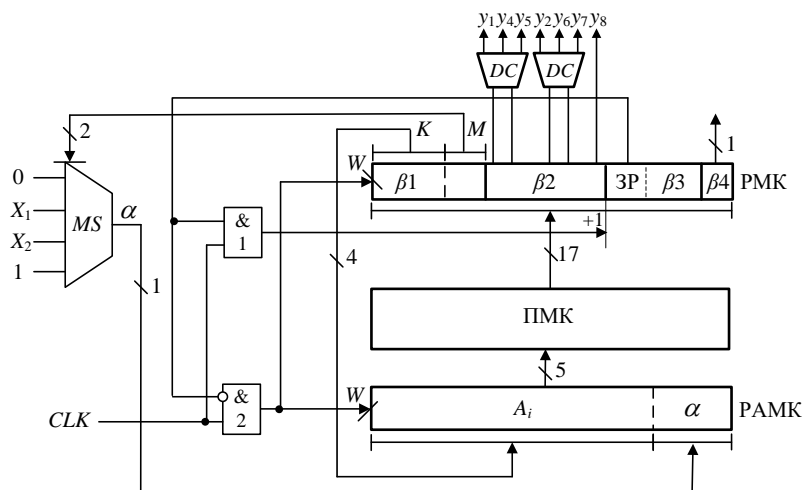


Рис. 3.11. Схема БМУ з примусовою адресацією

Підготовка до роботи

6. Варіанти завдання визначаються молодшими розрядами a_7, \dots, a_1 двійкового номера залікової книжки.

7. Розробити структурну схему операційного пристрою та змістовний мікроалгоритм обробки додатних чисел відповідно до завдання на-

веденого у табл. 3.4. Для побудови схеми використати комбінаційний суматор, регістр-лічильник циклів та асинхронні регістри, що мають входи управління зсувами і занесенням інформації. На структурній схемі повинні бути зазначені розрядність регістрів та шин.

8. Розробити функціональну схему операційного пристрою.

9. Виконати логічне моделювання роботи операційного пристрою за допомогою цифрової діаграми для вибраних значень операндів і їх розрядності.

Таблиця 3.4. Варіанти завдання

a_6	a_5	a_4	Функція	Розрядність операндів (без знаку)
0	0	0	1-й спосіб ділення	7
0	0	1	2-й спосіб ділення	4
0	1	0	$F = \sqrt{X}$	8
0	1	1	$F = X^2$	4
1	0	0	1-й спосіб множення	6
1	0	1	2-й спосіб множення	4
1	1	0	3-й спосіб множення	6
1	1	1	4-й спосіб множення	4

10. Побудувати структурну і функціональну схему БМУ, а також карту програмування ПМК для мікроалгоритму виконання заданої операції.

11. Використувати горизонтальне програмування зони управляючих сигналів. Врахувати дані, наведені у табл. 3.5 і 3.6.

Таблиця 3.5. Варіанти завдання

a_4	a_2	Спосіб адресації мікрокоманд	Ємність ПМК, слова	Використати зону β_4 для перевірки слова МК
0	0	примусовий	32	на непарність
0	1			на парність
1	0	відносний	16	на непарність
1	1			на парність

Таблиця 3.6. Варіанти завдання

a_6	a_5	a_4	Тривалість мікрооперації підсумовування, такти
0	0	0	7
0	0	1	4
0	1	0	3
0	1	1	6
1	0	0	11
1	0	1	4
1	1	0	5
1	1	1	2
Інші мікрооперації виконуються за один такт			

Виконання роботи

1. Використовуючи моделюючу систему AFDK (ПРОГМОЛС 2.0) побудувати і налагодити пристрій для виконання заданої операції. Опис програмного комплексу ПРОГМОЛС 2.0 наведений у додатку.

2. На спроектованому пристрої виконати числовий приклад і порівняти результат з одержаним при логічному моделюванні.

Зміст звіту

Звіт з лабораторної роботи повинен включати короткі теоретичні відомості, необхідні для виконання лабораторної роботи; структурні та функціональні схеми; таблиці та діаграми, отримані при виконанні теоретичного завдання, а також у процесі моделювання схем; висновки.

Контрольні питання

1. Наведіть загальну конструктивно-функціональну структуру ЕОМ, пояснити загальне призначення БМУ та АЛП.
2. Наведіть порівняльну характеристику АЛП з розподіленою та зосередженою логікою.
3. Приведіть етапи побудови АЛП із розподіленою логікою.
4. Визначіть призначення АЛП у ЕОМ. Наведіть класифікацію АЛП.
5. Визначіть призначення БМУ у ЕОМ, наведіть класифікації БМУ.
6. Поясніть, що розуміють під принципом мікропрограмного управління?
7. Наведіть класифікацію БМУ з точки зору забезпечення тривалості виконання мікрооперацій. Наведіть недоліки і переваги кожного із способів.
8. Як забезпечується тривалість виконання мікрооперацій при асинхронному способі управління виконанням МК у БМУ?
9. Наведіть формат слова мікрокоманди і поясніть призначення кожної із зон.
10. Як визначити довжину зони β_2 при горизонтальному способі мікропрограмування?

11. Назвіть способи формування структури зони β_2 , переваги та недоліки кожного із способів.

12. Як визначити довжину зони β_3 формування управляючих сигналів БМУ при асинхронному способі управління виконанням МК?

13. Назвіть способи формування структури зони β_1 , переваги та недоліки кожного із способів.

14. Як скоротити довжину зони β_1 при застосуванні примусової адресації МК?

15. Наведіть приклад застосування зони β_4 .

Список літератури

4. Арифметичні та управляючі пристрої цифрових ЕОМ: Навчальний посібник / В.І.Жабін, І.А.Жуков, І.А.Клименко, Стиренко С.Г. – К.: ВЕК +, 2008. – 176 с.

5. Жабін В.И., Жуков І.А., Ткаченко В.В., Клименко І.А. Мікропроцесорні системи: Навчальний посібник. – К. Видавництво «СПД Гуральник», 2009. – 492 с.

6. *Прикладана* теорія цифрових автоматів: Навчальний посібник / В.І.Жабін, І.А.Жуков, І.А.Клименко, В.В.Ткаченко. – К.: Книжкове видавництво НАУ, 2007. – 364 с.

7. *Цифровые ЭВМ. Практикум* / К.Г.Самофалов, В.И. Корнейчук, В.П. Тарасенко, В.И.Жабін – К.: Выш.шк. 1989. – 124 с.

4. ЗАГАЛЬНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ 3,4

При виконанні лабораторних робіт дослідження архітектури, системи команд і мікропрограм може бути використаний програмний моделюючий комплекс.

Комплекс включає програмний емулятор ЕОМ з мікропрограмним керуванням, мікроасемблер, текстовий редактор та інші програми, об'єднані в інтегроване середовище. Комплекс дозволяє розробляти та відлагоджувати мікропрограми для будь-якої системи команд у рамках заданої структури ЕОМ, системи мікрооперацій, розрядності слів даних і адреси.

Структура ЕОМ показана на рис. 4.1.

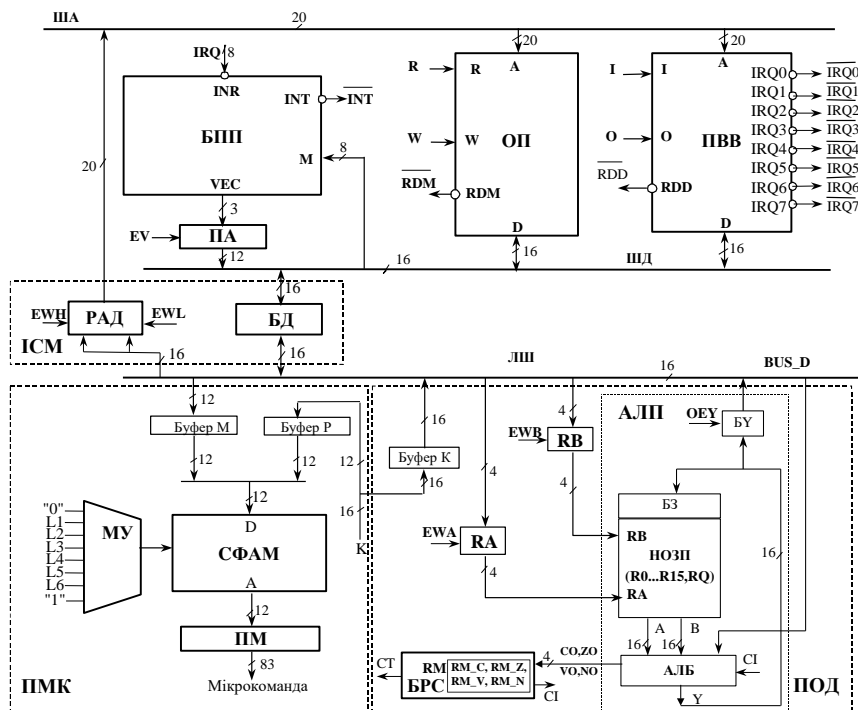


Рис. 4.1. Структура ЕОМ

До складу ЕОМ входить процесор, основна пам'ять (ОП), пристрій вводу-виводу (ПВВ), блок пріоритетних переривань (БПП).

Процесор складається з пристрою обробки даних (ПОД), пристрою мікропрограмного керування (ПМК) і інтерфейсу системної магістралі (ІСМ). До складу ПОД входять арифметико-логічний пристрій (АЛП) і блок реєстра стану (БРС). ПМК містить мультиплексор умов (МУ), схему формування адреси мікрокоманди (СФАМ), пам'ять мікрокоманд. Інтерфейс ІСМ містить реєстр адреси (РАД) і буфер даних (БД).

Програмний емулятор настроєний на 16-розрядні реєстри процесора. Шина адреси (ША) і шина даних (ШД) мають відповідно 20 і 16 розрядів. Мікропрограми можуть бути написані на мікроасемблері, а також в кодах мікрокоманд.

Для запису слова в ОП необхідно завантажити адресу комірки пам'яті в РАД, а потім подати сигнал R (Read). Через визначений проміжок часу на ШД встановлюється вміст комірки пам'яті, до якого здійснювалося звертання. При цьому ОП виробляє сигнал готовності RDM (ReaDy Memory). Через БД зчитане з пам'яті слово може бути записане у визначений реєстр процесора. При записі інформації в ОП після завантаження адреси в РАД через БД виставляється слово, яке повинне бути записане в пам'ять, а потім видається керуючий сигнал W (Write). По завершенні запису в ОП формується сигнал RDM.

Локальна шина (ЛШ) процесора є 16-розрядною. Тому завантаження РАД здійснюється в два етапи. В старші розряди запис інформації супроводжується сигналом EWH, а в молодші - EWL.

Обмін інформацією між процесором та ВПП відбувається через реєстр даних (РД), який входить до складу ВПП. Готовність ПВВ до обміну визначається за допомогою його реєстру стану (РС). Розряд РС за номером 7 містить біт "Готовність". При зверненні до ВПП у програмному режимі спочатку необхідно перевірити його готовність. Реєстри РД та РС мають адреси у адресному просторі процесора. Звертання до реєстрів ПВВ

здійснюється аналогічно звертанню до комірок ОП, але замість сигналів R, W і RDM формуються відповідно сигнали I (Input), O (Output) і RDD (ReaDy Device). Крім цього, ПБВ можуть формувати сигнали запиту переривань IRQ_i ($i=0, \dots, 7$), якщо вони готові до обміну даними. Дозвіл на формування запиту на переривання забезпечується записом одиниці у 6-й розряд PC (розряд дозволу переривання).

БПП забезпечує аналіз пріоритетів сигналів IRQ_i , які потрапляють на входи INR, формування сигналу INT вимоги переривання і вектору переривання VEC. Вхід M дозволяє маскування сигналів переривань від ПБВ. Перетворювач вектору (ПВ) дозволяє формувати початкову адресу підпрограми обслуговування переривання. Адреса передається на ШД по сигналу EV.

Система може бути настроєна на реалізацію розподіленого контролера переривань. У цьому випадку у склад кожного ПБВ входить блок переривань, який формує сигнал IRQ_i . Всі блоки зв'язані пріоритетним ланцюжком (на рис. 16.1 не показаний). Виходи IRQ_i об'єднані через монтажну логічну функцію I. Загальний сигнал IRQ приймає активний рівень (нульовий), коли є хоча б одна вимога переривання. Цей сигнал перевіряється в ПМК. Якщо необхідне виконати переривання програми, то ПМК видає по черзі два сигнали: LOCK (блокування пріоритетного ланцюжка) та INTA (сигнал підтвердження переривання). Після цього ПБВ видає на ШД вектор переривань, який може бути прийнятим у процесор.

АЛП містить арифметико-логічний блок (АЛБ) і надоперативний запам'ятовуючий пристрій (НОЗП), до складу якого входять 16 регістрів ($R0 \dots R15$) та допоміжний регістр RQ. Інформація із НОЗП може видаватися одночасно по двох каналах (A і B). Вибір регістрів здійснюється подачею адрес RA і RB на відповідні входи НОЗП. Адреси можуть видаватися з регістрів RA і RB або безпосередньо з ПМК, минаючи зазначені регістри

(відповідні зв'язки на рис. 16.1 умовно не показані). Запис адрес у RA і RB відбувається відповідно сигналами EWA і EWB.

АЛБ виконує мікрооперації, зазначені а табл.4.1, де через R і S позначені операнди, а через CI - вхідний перенос у молодший розряд. В якості операндів R і S може використовуватися інформація на локальній шині (BUS_D), а також з регістрів НОЗП по каналах А і В.

Табл. 4.1. Мікрооперації в АЛБ

Мнемоніка	Мікрооперація
ADD	$R+S+CI$
SUB	$R-S-1+CI$
OR	R or S
AND	R and S
NAND	not R and S
XOR	R xor S
NXOR	not (R xor S)

Результат мікрооперації Y через блок зсуву зсувач (БЗ) може бути записаний в регістр НОЗП за адресою RB. БЗ забезпечує передачу результату без зсуву, а також із зсувом на один розряд вліво або вправо. Якщо зсув не виконується, то результат може бути записаним у RQ. Крім того, через буфер Y (БУ), що керується сигналом OEY, результат мікрооперації може бути виданий на локальну шину.

При виконанні мікрооперацій в АЛБ формуються ознаки:

CO - перенос зі старшого розряду;

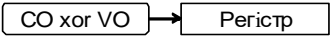
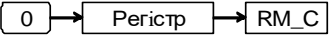

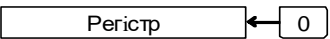
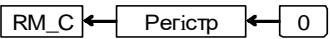
ZO - нульовий результат;

NO - негативний результат;

VO - переповнення розрядної сітки.

Ознаки можуть бути записані у відповідні розряди регістра RM, що входить до складу БРС. Ознаки, записані в RM, позначаються як RM_C, RM_Z, RM_N, RM_V. Логічні ланцюги БРС забезпечують підключення до входу CI АЛБ прямого або інверсного значення ознаки RN_C, а також сигналів логічного нуля або одиниці. Біт RM_C бере участь також у деяких мікроопераціях зсуву. Різновиди зсувів зазначені в табл. 4.2.

Табл. 4.2. Мікрооперації зсуву

Мнемоніка	Найменування	Схема зсуву
SRA	Зсув вправо арифметичний	
SRL	Зсув вправо логічний	
SR.9	Зсув вправо з переносом	
SLA	Зсув вліво арифметичний	
SLL	Зсув вліво логічний	
SL.25	Зсув вліво з переносом	

ПМК забезпечує аналіз логічних умов, що надходять на входи L1, L2,...,L6. До входів логічних умов можуть бути підключені зовнішні відносно ПМК сигнали (RDM, RDD, СТ та ін.). З виходу СТ БРС можуть бути видані прямі та інверсні значення ознак, що формуються безпосередньо в АЛБ, а також зберігаються в RM. ПМК формує керуючі сигнали, що забезпечують роботу всіх компонентів системи. (На рис. 4.1 зазначені тільки ті сигнали, що використовуються при записі мікрокоманд на мікроасемблері). Виходи буфера К забезпечують видачу константи К на ЛШ. Поле К константи належить слову мікрокоманди. Константа К може прийматися в ПОД, а також через буфери Р і М у ПМК. Виходи буфера М використовую-

ються для завдання початкової адреси мікрокопрограми, а буфера Р – для завдання адреси переходу в мікропрограмах.

Мнемонічний двохпрохідний мікроасемблер призначений для розробки мікропрограм.

Результатом роботи мікроасемблера є файл даних з розширенням ".pmk", який є вихідним для програмного емулятора.

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 т. ін.).

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

Коментарі, числові константи і мітки. Коментарі використовуються для пояснень. Ознакою початку коментарю служить символ “\”. Далі мікроасемблер ігнорує всі символи, які зустрічаються до наступного символу “\” або до кінця рядка.

Приклад.

```
\
*****
\   Приклад коментарів у мікропрограмі
\
*****
{ add R11, R11, R10,Z; } \ додати до R11 вміст R10
```

Числові константи застосовуються при завданні значень операндів і адрес. Ознакою числової константи є цифра на початку мнемоніки.

Приклади.

65535	\ десяткова константа
0FFFFh	\ шістнадцяткова константа
177777o	\ вісімкова константа
111111111111111%	\ двійкова константа

Мітки можуть включати до 10 символів (букви, цифри і символ “_”), причому, першим символом повинна бути буква. Ознакою кінця мітки служить кожний з наступних роздільників: пробіл, повернення каретки, переведення рядка, табуляція. Мітка не повинна збігатися з зарезервованою мнемонікою.

Приклади.

```
Loop
First_go
Мітка_1
LABEL1
```

Мікрокоманди АЛП. Арифметичні мікрокоманди, що виконуються в АЛП, записуються у вигляді

**<мнемоніка> [<оператор_зсуву>,) [<приймач_результату>,)
<джерело_1>, <джерело_2>,<вхідний_перенос>**

У квадратних дужках зазначені необов'язкові елементи конструкції.

Запис логічних мікрокоманд відрізняється від арифметичних відсутністю останнього операнда, тому що перенос не бере участь у логічних мікроопераціях.

Мнемоніка арифметичних і логічних мікрокоманд зазначена в табл. 17.1. Як приймач результату може бути зазначений кожний з регістрів R0 – R15. Крім того, в якості приймача може бути зазначений RQ (якщо зсуву не було), а також NIL. В останньому випадку результат у НОЗП не записується, але може бути виданий на локальну шину через БУ. Якщо приймач результату не вказується, то результат записується на місце першого джерела операндів.

Джерелами операндів можуть бути регістри НОЗП, а також один регістр (він вказується як перше джерело операндів) у комбінації з константою, BUS_D чи нулем. Нуль у полі джерела операнда позначається буквою Z. Регістри НОЗП можуть адресуватися непрямою адресацією. Якщо у якості джерела операндів зазначені RA і/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 і NZ), а також RM_C і not RM_C (інвертоване значення розряду RM_C). Мнемоніки операторів зсуву зазначені в табл. 17.2.

Приклади мікрокоманд.

add srl,r10,r10,r2,z	\відповідає мікрооперації \ r10 := 0.r(r10 + r10)
xor r5,r5	\ r5 := r5 xor r5 або r5 := 0
SUB R7,R7,BUS_D,NZ	\ R7 := R7-BUS_D
and rb,0Ch	\якщо в rb записаний код 011%, \ to r3:=r3 and 0000000000001100%
add r9,z,nz	\ r9:=r9+1

Інформація в 4-розрядні регістри адреси RA і RB записується з ЛШ. Комутація ліній ЛШ і входів регістрів задається директивою LINK (див. нижче).

Для завантаження регістра RM в БРС можуть використовуватися наступні мікрокоманди.

Load RM, Z	\ встановлення всіх розрядів в нуль
Load RM, NZ	\ встановлення всіх розрядів в одиницю
Load RM, FLAGS	\завантаження всіх ознак, сформованих \ при виконанні мікрооперації в АЛБ

Разом із зазначеними мікрокомандами можуть бути використані мікрокоманди заборони запису в розряди RM, а саме: CEM_C, CEM_Z, CEM_N, CEM_V.

Приклад.

load rm, flags; sem_v; \забезпечить завантаження
sem_n; sem_z \тільки розряду RM_C.

Мікрокоманди ПМК. У Табл. 17.3 приведені мнемоніки основних мікрокоманд передачі керування, використовуваних у ПМК. У таблиці симво-

лом <Y> позначені позиції запису умови, а символом <A> – адреса або мітка.

Табл. 17.3. Мікрооперації в ПМК

Мнемоніка	Мікрооперація
JZ	Безумовний перехід на нульову адресу
JMAP	Безумовний перехід за адресою на ЛШ
CONT	Безумовний перехід до наступної мікрокоманди
CJP <Y>,<A>	Умовний перехід по зазначеній адресі (мітки)
CJS <Y>,<A>	Умовний перехід до мікропідпрограми по зазначеній адресі (мітки)
CRTN <Y>	Умовне повернення з мікропідпрограми

Як умови можуть бути використані ознаки: RDM, RDD, INT, IRQ_i, RM_C, RM_Z, RM_N, RM_V, CO, ZO, NO, VO. Можна також зазначити заперечення умов (наприклад, not RM_C). Крім того, можна вказати умову Z, що ніколи не виконується, а також умову NZ, що завжди виконується.

Мікрокоманди керування пристроями та вузлами. Мнемоніка мікрокоманд керування зовнішніми пристроями, пам'яттю, регістрами і буфером БУ збігається з найменуванням відповідного керуючого сигналу (табл. 17.4).

Мікрокоманди для БПП. Для роботи з БПП передбачені спеціальні мікрокоманди. Для ініціалізації БПП виконується мікрокоманда reset. Сигнали вимоги переривань, що надходять з ПВВ, фіксуються в регістрі IR, що входить до складу БПП. Для читання вектора переривання з виходів VEC використовується мікрокоманда READ VR. Для видачі на ЛШ відповідної адреси необхідно в цьому ж такті сформувати сигнал EV. Для переривання, вектор якого зчитаний, необхідно виконати мікрокоманду

CLR IR, VR, яка забезпечує встановлення нуля у відповідному розряді регістру IR. При ініціалізації системи можна маскувати групу переривань від ПВВ. Для цього необхідно завантажити в БПП маску, використовуючи мікрокоманду LOAD MR, <маска>.

Табл. 17.4. Керування пристроями та вузлами

Мнемоніка	Найменуванням мікрокоманди
R	Читання з ОП
W	Запис в ОП
I	Ввід із зовнішнього пристрою
O	Вивід у зовнішній пристрій
EWH	Запис в старші розряди РАД
EWL	Запис в молодші розряди РАД
OEU	Видача результату Y з БУ на ЛШ
EWA	Запис в RA
EWB	Запис в RB
EV	Видача вектора з ПА на ЛШ

Директиви. Мікроасемблер підтримує директиву **ORG**, яка має вигляд

ORG <мітка/адреса>

Директива забезпечує розміщення виконавчого коду в пам'яті мікрокоманд по зазначеній адресі. Адреса задається у вигляді числової константи або раніше визначеної мітки.

Приклади.

org 010h

org start

Для завдання відповідностей використовується директива **EQU**, що записується у формі

EQU <ім'я>:<еквівалент>

Приклади.

equ start : 10

equ скласти: add

equ операнд2:15

Директива **INCLUDE** вставки в трансльовану мікропрограму тексту з файлу записується так:

INCLUDE <ім'я_файлу>

Файл, що задається в директиві повинний знаходитися в тій же директорії, що і трансльований файл.

Приклади.

include macro.lib

include routine

Директива макросу задається за допомогою ключового слова **MACRO** наступним чином

MACRO <ім'я> <формальні параметри>:<мікрокоманда> Директива завдання макросів дозволяє конструювати власні мікрокоманди і користуватися ними надалі як стандартними.

Приклади

MACRO inc reg : add reg, reg, z, nz

MACRO mov reg1,reg2 : or reg1,reg2,z

macro jmp label : cjp nz, label

Ім'я макросу надалі стає для транслятора звичайною стандартною мнемонікою з усіма правилами, що поширюються на неї. Імена формаль-

них операндів макросу не можуть бути зарезервованими мнемоніками. В програмі макрос задається вказівкою свого ключового імені і реальних операндів перерахованих через кому в тім же порядку, що і при завданні макросу.

Приклади.

inc r4 \ r4:=r4+1

mov r3,r8 \ r3:=r8

jmp delta

Директива **DW** завдання значень комірок пам'яті має вигляд

DW <адреса>:<значення>

Приклади

DW 12:0ffh

dw 03Fh:15

Директива установки регістрів **АССЕРТ** дозволяє встановити початковий стан регістрів R0,...,R15, RA,RB і RM. Форма запису

АССЕРТ <регістр>:<значення>

Приклади.

АССЕРТ RA:12

АССЕРТ R4:0ffh

АССЕРТ RM: 0101%

Директива АССЕРТ використовується також для установки стану зовнішніх пристроїв. Можна задати такі характеристики зовнішніх пристроїв (DEV):

- тип пристрою (І - вводу, О - виводу);
- адреса регістра стану (РС) в межах 64К (0000h – 0ffffh);
- адреса регістра даних (РД) в межах 64К;

- затримка в тактах формування сигналу RDD;
- затримка в тактах установки біта "Готовність" в регістрі стану.

Приклад.

```

асцепт dev[2]: i,      \ пристрій вводу
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 вводу можна задавати внутрішній буфер даних DEV_BUF, обсягом до 16 слів.

Приклад

```

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

```

Зазначені після ":" дані вводяться в процесор по черзі при кожному звертанні до РД даного пристрою вводу.

Директива АСЦЕПТ дозволяє задати швидкодію пам'яті за допомогою перемінної RDM_DELAY, наприклад,

```
АСЦЕПТ RDM_DELAY : 3
```

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

Для опису конфігурації зв'язків між компонентами системи використовується директива **LINK**.

Для установки відповідності входів I1,...,I6 ПМК і логічних умов використовується директива

```
LINK <ім'я_входу>:<умова>
```

Приклади

link L1:rdm

LINK 12:RDD

link 13:ct

Підключення 20-розрядного РАД до 16-розрядної ЛШ описується директивою

LINK EWN : <номер_розряду>

Номер розряду РАД[19...0], зазначений у директиві, розділяє регістр на дві частин. Старша частина, включаючи зазначений розряд, керується сигналом EWN, а молодша - EWL.

Приклад

LINK EWN : 16

Вказана директива забезпечує зв'язки між ЛШ і РАД так, що по сигналу EWN чотири молодших розрядів на ЛШ записуються в поле РАД[19...16]. По сигналу EWL всі 16 розрядів із ЛШ записуються в РАД[15...0], тобто в молодшу частину регістра.

Підключення вхідної 12-розрядної шини М ПМК до ЛШ задається директивою LINK, в якій перелічуються номери розрядів ЛШ, зв'язані з входами М.

Приклад

LINK М: 15,14,13,12,11,10,9,8,7,6,5,4

Дана директива визначає підключення до входів М 12-ти старших розрядів ЛШ.

Аналогічно задається підключення до ЛШ входів RA і RB.

Директива LINK також визначає в ПА адресу для різних номерів ПБВ.

Приклад

link vec[2]: 0CDEFh

У цьому випадку на виході ПА по сигналу EV видається слово 0CDEFh, якщо на входах є присутнім код 2.

Приклади

LINK RA : 11,12,13,14

LINK RB : 10,9,8,7

В один момент часу в різних вузлах системи можуть виконуватися різні мікрооперації. Всі мікрокоманди, що керують мікроопераціями, які виконуються в одному такті, записуються в операторних дужках "{" і "}", утворюючи повну мікрокоманду для даного такту роботи ЕОМ. Окремі мікрокоманди розділяються символом ";" і, крім того, можуть використовуватися роздільники типу "пробіл", "повернення каретки", "переведення рядка". Повна мікрокоманда може займати кілька рядків в тексті мікропрограми. При необхідності мітка записується перед операторної дужкою, яка відкривається.

Розглянемо приклади мікропрограм.

Приклад

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

link l1:rdm	\підключення виходу RDM до l1
link ewh:16	\установка зв'язків між РАД і ЛШ
accept rdm_delay:2	\затримка формування RDM
accept r7:1234h	\вихідна установка R7 (для \налагодження)
dw 1234h:070fh	\визначення даних в ОП за \адресою 1234h

```

{cont;xor nil,r0,r0;oeu;ewh;} \РАД[19...16]:=0
{cont;or nil,r7,z;oeu;ewl;} \РАД[15...0]:=r7
    ll1 {cjp rdm,ll1;r;or r15,bus_d,z;} \R15:=070fh (дані з ОП)
{} \кінець мікропрограми

```

Зауважимо, що мікрокоманду cont записувати не обов'язково.

Приклад

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

```

link ll1:rdm
link ewh:16
accept rdm_delay:3
accept ra:3
accept r3:0004h
accept r1:4
accept r2:16
accept r15:32
{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:=l(R1+R15).0 (результат в R1)
{or nil,ra,z;oeu;ewl;} \ запис адреси в РАД
    ll2 {cjp rdm,ll2;w;or nil,r1,z;oeu;} \запис результату в пам'ять
{} \кінець мікропрограми

```

Приклад

Мікропрограма налаштування БПП і читання вектора переривання з ПА в АПП.

link vec[2]: 0CDEFh \ завдання в ПА адреси для переривання від
 \ 2-го ПВВ

link L[4]:INT \ підключення виходу INT до входу I4 ПМК

{reset;} \ ініціалізація БПП

{load mr,11111011%;} \ дозволити обробку переривання IRQ₂

< мікропрограма виконання команди, включаючи формування адреси
 наступної команди>

{cjp not INT, MakeInt;} \ якщо INT=0,то перейти до читання вектора

< інакше почати обробку наступної команди >

MakeInt \ початок обробки переривання

{ev;read vr;or r9,bus_d,z;} \ читання вектора в R9

{clr ir,vr;} \ скидання розряду в регістрі IR БПП, що

\ відповідає даному перериванню

< продовження мікропрограми обслуговування переривань >

Приклад.

Для системи з розподіленим контролером переривань настроїти ВПП
 з номером 4 на ввід даних, ВПП з номером 6 на вивід даних.

АССЕРТ I, \ пристрій вводу

DEV[4]:

1234h, \ адреса РС

1236h, \ адреса РД

10, \ затримка в тактах формування RDD

50, \ затримка в тактах формування сигналу

\ готовності в РС

4444h \ вектор переривання

АССЕРТ O, \ пристрій виводу

DEV[6]:

12a4h,	\ адреса РС
12a6h,	\ адреса РД
8,	\ затримка в тактах формування RDD
30,	\ затримка в тактах формування сигналу
	\ готовності в РС
444eh	\ вектор переривання

Присутність в опису ВПП вектора переривань автоматично забезпечує роботу ВПП у складі пріоритетного ланцюжка. У протилежному випадку ВПП може працювати в режимі переривань тільки разом з централізованим БПП.

5. ЛАБОРАТОРНА РОБОТА №3

ПЕРЕТВОРЕННЯ ДАНИХ В ЕОМ НА МІКРОПРОГРАМНОМУ РІВНІ

Мета роботи - вивчити архітектуру ЕОМ, що містить блок мікропрограмного керування і арифметико-логічний пристрій з зосередженою логікою та двухадресним надоперативним запом'ятовуючим пристроєм (НОЗП), одержати навички розробки мікропрограм.

Теоретичні відомості

Перетворення даних в ЕОМ може виконуватися на програмному, мікропрограмному та апаратному рівні.

На програмному рівні обробка інформації відбувається за допомогою *програми* – зв'язаного списку команд. Програма міститься в оперативній пам'яті. Команди виконуються в певній послідовності, що визначається вихідними даними, проміжними результатами, ознаками тобто. Чергова адреса команди формується в лічильнику команд. Множина всіх команд, що може виконувати процесор, називається *системою команд процесора*.

Команда – інформаційне слово, що в загальному випадку визначає операцію, що виконується з операндами, місце розташування операндів і результату операції, а також вказує на адресу наступної команди.

Процесор послідовно зчитує команди з пам'яті і виконує їх за допомогою мікропрограм і апаратних засобів. Як правило, в CISC-системах (CISC – Complex Instruction Set Computing) це відбувається на мікропрограмному рівні, а в RISC-системах (RISC – Reduced Instruction Set Computing) можлива апаратна інтерпретація реалізації команд.

Мікропрограма – це зв'язаний список мікрокоманд, виконання яких в певній послідовності забезпечує необхідне перетворення даних.

Мікропрограми записані в пам'ятті мікрокоманд (ПМК), що входить в склад блоку мікропрограмного управління (БМУ).

Мікрокоманда є інформаційним словом, що в загальному випадку містить наступну інформацію:

- зона управляючих сигналів;
- зона тривалості управляючих сигналів;
- зона формування адреси наступної мікрокоманди;
- зона службових розрядів (наприклад, контрольних).

Під управлінням мікрокоманди в різних пристроях ЕОМ виконуються *мікрооперації* – елементарні машинні дії. Послідовність мікрооперацій, що забезпечує необхідне перетворення інформації, називають *мікроалгоритмом*. Кожній команді, що зчитується з оперативної пам'яті, відповідає певна мікропрограма в ПМК, що реалізує певний мікроалгоритм (команда інтерпретується мікропрограмою).

В спеціалізованих обчислювачах, які виконують тільки скінчену кількість алгоритмів, перетворення даних може виконуватись тільки на мікропрограмному рівні. В таких системах кожний мікроалгоритм перетворення даних реалізується виконанням послідовності мікрокоманд, тобто певною мікропрограмою. Така послідовність мікрокоманд може виконати складне перетворення, а саме: введення даних, виконання послідовності різних операцій, запис результату. На програмному рівні аналогічне перетворення даних вимагає виконання певної послідовності команд, тобто програми. Крім безпосереднього перетворення даних це вимагає витратити час на читання команд з пам'яті, їх дешифрування тощо. Таким чином, на мікропрограмному рівні можна прискорити обчислення відносно програмного рівня, хоча при цьому втрачається універсальність системи.

В даній роботі розглядаються способи обробки операндів, поданих у прямих та доповняльних машинних кодах (ПК та ДК), на мікропрограмному рівні в ЕОМ з мікропрограмним управлінням (розділ 4).

Підготовка до лабораторної роботи

1. Вивчити загальну інформацію до виконання лабораторних робіт 3 і 4 на моделюючому комплексі ЕОМ з мікропрограмним управлінням (розділ 4).

2. Визначити свій варіант завдання у відповідності до табл. 5.1 – 5.4, в яких $a_7, a_6, a_5, a_4, a_3, a_2, a_1$ - сім молодших розрядів двійкового номера залікової книжки студента (a_1 – молодший розряд).

3. Для 16-розрядного процесора розробити операційну схему, функціональний (Ф) і структурний (С) мікроалгоритми обчислення функції за формулою

$$Z = \begin{cases} X \cdot Y, & \text{якщо } g = 1; \\ 2^s X \pm 2^c Y, & \text{якщо } g = 0. \end{cases}$$

Операнди $X, Y \in 16$ -розрядними двійковими числами (два старші розряди є знаковими) у заданих машинних кодах (ПК, ДК). Ознака g визначається у вихідному стані значенням розряду з двійковим номером $a_3a_2a_1$ (від 0 до 7) регістра стану процесора (наприклад, R14 в НОЗП процесора). Табл. 5.1 визначає спосіб множення, табл. 5.2 – операцію додавання (віднімання), табл. 5.3 – форму подання даних (ПК або ДК), а табл. 5.4 – значення операндів для тесту мікропрограми.

3. Операнди X та Y зчитуються із оперативної пам'яті системи, результат також записується в пам'ять. Результати, що мають довжину,

більшу за 16 розрядів, записуються в пам'ять 16-розрядними словами в сусідні комірки (молодші розряди – за меншими адресами). Адреси оперативної пам'яті задати самостійно.

4. Подати фрагмент структурної схеми ЕОМ (див. розділ 4), що містить процесор і пам'ять. Показати на схемі основні вузли процесора та мнемоничні позначення сигналів, що використовуються в мікроасемблері (RDM, EWN, EWL, OEY...), вивчити їх призначення.

5. Розробити з використанням мікроасемблера мікропрограму реалізації отриманого мікроалгоритму.

6. Виконати числовий приклад, використовуючи дані табл. 5.4. Виписати контрольні значення проміжних результатів, які будуть використані при налагодженні мікропрограми.

Приклад операційної схеми множення додатних 16-розрядних чисел першим способом показаний на рис. 5.1 (знакові розряди операндів дорівнюють нулю). Множення здійснюється з молодших розрядів множника і з зсувом суми часткових добутоків вбік молодших розрядів. В вихідному стані множник записаний в R10. Регістр R9 в вихідному стані містить множене, а R12 - повинний бути обнулений. В останньому з зазначених регістрів при виконанні операції накопичуються старші розряди суми часткових добутоків. Молодші розряди результату при зсуві заповнюють звільнені розряди регістра R10 (регістра множника). Результат, який містить 2n розрядів, буде знаходитися в регістрах R12 і R10. У якості лічильнику циклів використовується R11.

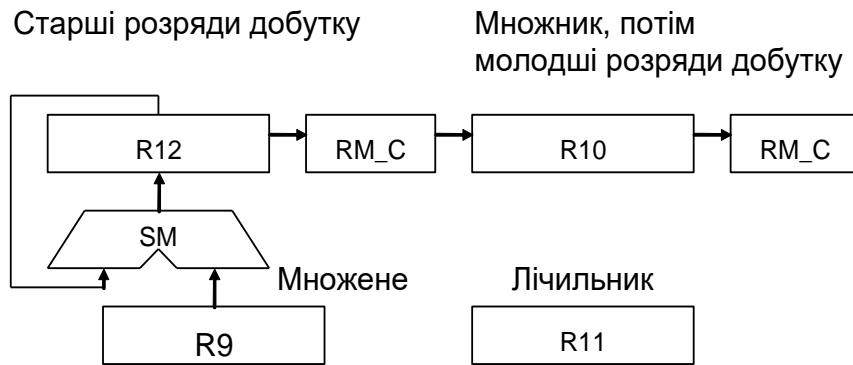


Рис. 5.1. Операційна схема множення

Мікропрограма множення цілих додатних 16-розрядних чисел (без знаків) за схемою на рис. 5.1 може мати такий вигляд.

```

\-----
link l3: ct          \Завдання зв'язку між L1 і СТ
accept r12:0         \Початкова
accept r9:7fffh      \установка
accept r10:7fffh     \регістрів
accept r11:17        \і лічильника циклів
\-----

      {load rm, z;}          \Обнулення RM
label1 {cjp not rm_c,label2;} \Аналіз цифри множника
      {add r12,r12,r9,z;}    \Додавання множеного до R12
label2 {or srl,r12,r12,z;}   \Зсув в R12 і R10 суми
      {or sr.9,r10,r10,z;}   \часткових добутків
      {sub r11,r11,z,z;load rm,flags;cem_c;} \Декремент і перевірка
      {cjp not rm_z, label1;} \на нуль лічильника R11
      { }\----- кінець множення -----

```

Таблиця 5.1

a5	a4	Спосіб множення (при g=1)
0	0	4
0	1	1
1	0	2
1	1	3

Таблиця 5.2

a6	a1	Арифметична операція (при g=0)
0	0	$2^2 X - 2^{-1} Y$
0	1	$2^{-1} X + 2^{-2} Y$
1	0	$2^3 X - 2 Y$
1	1	$2^2 X + 2^{-1} Y$

Таблиця 5.3

a3	a2	Форма подання чисел		
		X	Y	Z
0	0	ДК	ДК	ПК
0	1	ПК	ПК	ДК
1	0	ПК	ДК	ПК
1	1	ДК	ПК	ДК

Таблиця 5.4

a7	a1	Значення операндів	
		X	Y
0	0	7	-5
0	1	-7	9
1	0	-9	-5
1	1	7	-6

Порядок виконання роботи

Налагодити розроблену мікропрограму з використанням програмного емулятора. Виконати обчислення функції Z з різними наборами вихідних даних. Зробити висновки по роботі.

Контрольні питання

1. Розробіть мікропрограму обчислення заданого викладачем арифметичного виразу.
2. Охарактеризуйте основні способи множення чисел.

3. Як забезпечити арифметичний і логічний зсув слів подвоєної довжини?
4. Яким чином можна керувати записом інформації в RM?
5. Поясніть призначення директив мікроасемблера.
6. Що таке мікроалгоритм, мікропрограма, мікрооперація і мікрокоманда?
7. Які мікрооперації в розглянутій системі можна сполучати, а які не можна?

Список літератури

1. Жабин В.И., Жуков І.А., Ткаченко В.В., Клименко І.А. Арифметичні та управляючі пристрої цифрових ЕОМ: Навчальний посібник. – К.: ВЕК +, 2008. – 176 с.
2. Жабин В.И., Жуков І.А., Ткаченко В.В., Клименко І.А. Мікропроцесорні системи: Навчальний посібник. – К. Видавництво «СПД Гуральник», 2009. – 492 с.
3. Жабін В.І., Ткаченко В.В. Цифрові автомати. Практикум. – К.: ВЕК+, 2004.– 160 с.
4. Комп'ютерна схемотехніка. Проектування обчислювальних систем з мікропрограмним керуванням: Метод. вказ. до виконання лаб. робіт/ Уклад.: Дичка І.А., Жабін В.І., Тарасенко В.П. – К.: НТУУ „КПІ”, 2006. – 64 с.

6. ЛАБОРАТОРНА РОБОТА №4

ОБРОБКА ІНФОРМАЦІЇ В ЕОМ НА ПРОГРАМНОМУ І МІКРОПРОГРАМНОМУ РІВНЯХ

Мета роботи – вивчити етапи формування системи команд процесорів, різновиди форматів команд та способів адресації операндів. Навчитися розробляти мікроалгоритми і мікропрограми виконання етапів команд з використанням мнемонічного мікроасемблера .

Теоретичні відомості

Системою команд процесора називають сукупність всіх команд , які може виконувати процесор. Залежно від кількості реалізованих команд комп'ютери прийнято ділити на два основних типи :

RISC – з скороченою системою команд ;

CISC – з комплексною системою команд .

Для RISC-систем кількість команд визначається десятками , а для CISC - сотнями .

Команда – інформаційне слово, що в загальному випадку містить такі дані:

- операція, що виконується з операндами;
- місце розташування операндів в пам'яті;
- місце запису результату операції;
- спосіб визначення адреси наступної команди.

Всі команди за функціональною ознакою діляться на групи :

- Основні команди – забезпечують перетворення інформації, тобто змінюють вміст регістрів. До них відносять: команди пересилання, зсуву, арифметичні і логічні команди;

- Команди передачі управління – забезпечують безумовні і умовні переходи, виклик і повернення з підпрограм;

- Команди вводу-виводу – забезпечують взаємодію процесора з зовнішніми пристроями (ЗП).

Для систем з поєднаним адресним простором ОП і ЗП команди даного типу можуть бути відсутніми;

- Системні команди – змінюють режими роботи процесора. Наприклад, дозволяють і забороняють переривання, блокують і розблоковують операції введення-виведення, встановлюють і скидають режим покрокового виконання програми і т.і.

Команди можуть відрізнятися структурою. Вони мають інформаційну частину (ІЧ) і адресну частину (АЧ). Інформаційна частина визначає виконувану операцію, формат команди, способи адресації операндів і т.д. Адресна частина складається їх полів A_i , які містить адреси операндів або інформацію для їх знаходження. Команди можуть мати різну кількість полів (рис. 6.1).

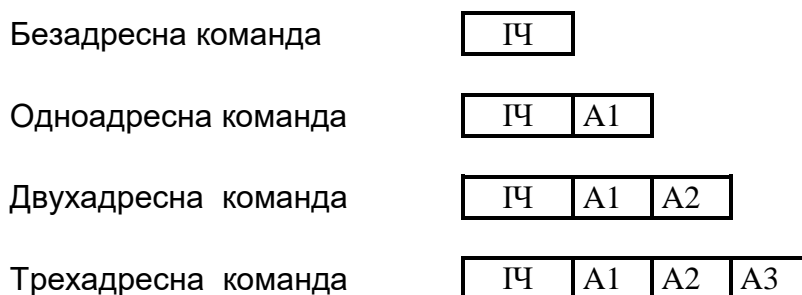


Рис. 6.1. Структури команд

Залежно від кількості адрес довжина команди може змінюватися. Якщо команда безадресна (без операндів), то це, як правило, системна команда. Одноадресні команди широко використовуються в якості команд передачі управління, команд вводу-виводу. При використанні

одноадресних команд для арифметичних і логічних операцій один операнд за замовчуванням знаходиться в регістрі процесора (регістрі-акумуляторі), а другий визначається адресним полем. Двоадресні команди - команди основної групи. В команді Вказуються два операнда, результат поміщається на місце одного з них. Трехадресні команди за рахунок поля АЗ дозволяють, наприклад, визначати адресу приймача результату або адресу наступної команди.

Етапи виконання команди основної групи:

- вибірка команди з оперативної пам'яті за адресою із лічильника команд;
- розпокування (дешифрування) команди: визначення формату команди, типу операції, адреси операндів та результату, вибірка операндів;
- виконання операції та запис результату;
- формування в лічильнику команд адреси наступної команди.

Виконавчою (фізичною) адресою називають послідовність нулів і одиниць, яка видається на адресні шини системи. Існують різні способи адресації операндів, тобто визначення виконавчої адреси операнда за інформацією в адресному полі. У багатоадресних командах спосіб адресації кожного операнда може бути різним.

Адресація операндів може здійснюватися з використання і без використання регістрів загального призначення (РЗП).

До способів адресації операндів без використання РЗП відносяться:

- пряма адресація операндів (в адресному полі команди вказується виконавча адреса операнда);
- непряма адресація (в адресному полі команди вказується виконавча адреса комірки пам'яті (вказівника адреси), в якій знаходиться виконавча адреса операнда);

- безпосередня адресація (в адресному полі команди вказується безпосередньо значення операнда).

Такі адресні поля в команді можуть мати велику довжину (32 і більше розрядів), що призводить до великої довжині самої команди. Це створює додаткові витрат часу на вибірку і розпакування команди.

Скоротити довжину команди дозволяє застосування способів адресації операндів з використанням РЗП. Адреса регістра (його номер) в НОЗП процесора має невелику розрядність (зазвичай не більше байта), що дозволяє зменшити довжину команди. У таких способах адресації виконавча адреса оперативної пам'яті (операнда або вказівника) визначається вмістом РЗП.

У даній роботі розглядається пряма (без використання РЗП) і пряма регістрова (з використанням РЗП) адресація операндів на прикладі одноадресних команд. При виконанні операцій з двома операндами один з них задається в адресному полі команди, а інший за замовчуванням знаходиться в регістрі-акумуляторі (АС).

Схема виконання команди додавання $ADD\ AC, M$ (до АС додається операнд М із ОП) при прямій адресації операнда пояснюється рис. 6.2. В адресному полі команди міститься виконавча адреса операнда М. Для завантаження даних в процесор відбувається одне звернення до ОП. Ще одне звернення необхідно для вибірки команди за адресою з лічильника команд (ЛК).

При прямій регістровій адресації операндів різко зменшується довжина команди (рис. 6.3.). Довжина поля R, в якому зберігається номер РЗП, повинна бути достатньою для нумерації будь-якого РЗП процесора. Для даної системи, що містить 16 РЗП, поле R має 4 розряди. При виконанні такої команди виконується одне звернення до пам'яті, що прискорює обчислення.

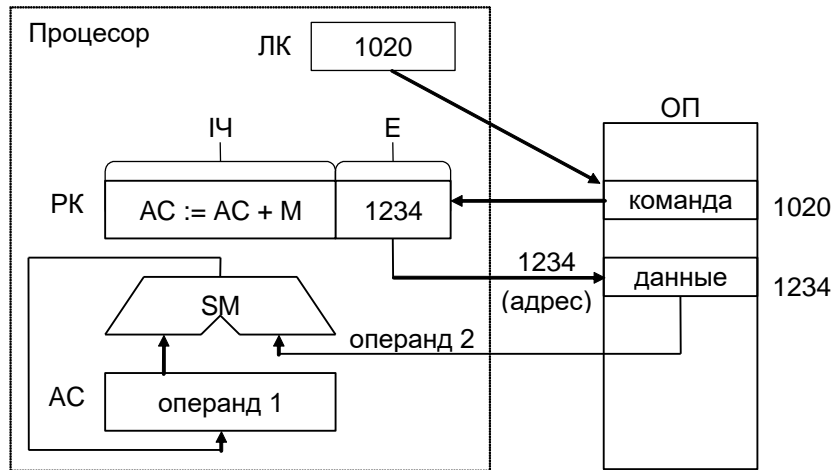


Рис. 6.2. Схема виконання команди при прямій адресації операнда

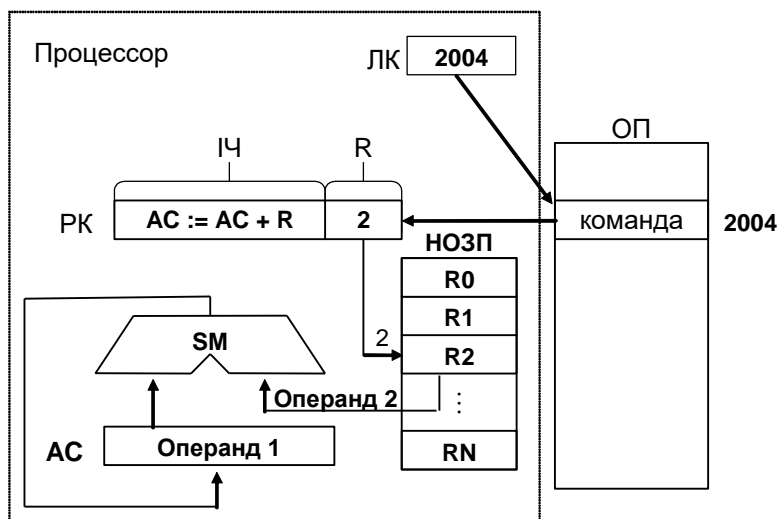


Рис. 6.3. Схема виконання команди при прямій регістровій адресації операнда

Однак, даний тип адресації придатний тільки для отримання проміжних результатів в процесорі. Він не дозволяє обмінюватися інформацією між процесором і ОП. Тому разом з цим типом адресації повинні використовуватися і інші способи адресації, що забезпечують взаємодію процесора з ОП.

Підготовка до лабораторної роботи

1. Вивчити загальну інформацію до виконання лабораторних робіт 3 і 4 на моделюючому комплексі ЕОМ з мікропрограмним управлінням (розділ 4).

2. Визначити свій варіант завдання у відповідності до табл. 6.1 – 6.4, в яких $a_7, a_6, a_5, a_4, a_3, a_2, a_1$ – сім молодших розрядів двійкового номера залікової книжки студента (a_1 – молодший розряд).

3. Подати архітектуру системи. Архітектура подається за допомогою моделі програміста, в якій вказуються регістри загального призначення, робочі регістри, представляється адресний простір ОП, НОЗП і ЗП.

4. Розробити на рівні операцій змістовний алгоритм обчислення функції

$$F = (X \diamond Y) \times Z,$$

де « \diamond » – операція, що задана табл. 6.1; « \times » – операція множення, що визначена табл. 6.4.

Операнди $X, Y, \text{ і } Z$ у доповняльному коді мають 16 розрядів з урахуванням двох знакових розрядів. Операнд X вводиться з пристрою введення. Адреси портів зовнішнього пристрою (ЗП) задані табл. 6.2. Готовність ЗП до обміну даними з процесором перевіряти на програмному рівні з використанням команди умовного переходу. Операнди $Y, \text{ і } Z$ знаходяться в ОП. Результат F записуються в ОП словами по 16 розрядів (молодші розряди – за меншими адресами). Послідовність введення даних в процесор визначається розробником. Вихідні дані задані в табл. 6.1-6.4. Відсутні дані, якщо необхідно, вибираються самостійно (їх вибір повинний бути обґрунтований).

На схемі алгоритму кожний блок (операторний або логічний) повинний відповідати одній команді процесора (додавання, множення, пересилання, введення, запис, умовний перехід і т.і.).

5. Сформувати систему команд процесора і розробити програму обчислень на базі команд одержаної системи. Необхідно обґрунтувати вибір системи команд, необхідних для обчислення функції. Програма обчислень представляється в кодах команд, а також у мнемонічному виді. Мнемоніка команд вибирається самостійно.

Формат одноадресних команд, а також регістра стану і даних ЗП показано на рис. 6.4 і 6.5. В командах з прямою адресацією операндів адреса ОП міститься в розрядах команди 9...0, тобто займає 10 розрядів. Це забезпечує доступ до ОП з адресами від 0 до 1023. Програма і дані повинні бути розміщені в цьому діапазоні адрес ОП. При використанні для прямої регістрової адресації адреса (номер) РЗП займає тільки молодші розряди адресного поля команди. Якщо для РЗП виділено в НОЗП 8 регістрів R0-R7, то для їх адресації достатньо 3 розрядів. В адресному полі залишається 7 розрядів, тобто при регістровій адресації можна побудувати двоадресну команду. Розробник може використовувати таку можливість.

7. Представити структурну схему ЕОМ, з зазначенням необхідних вузлів і керуючих сигналів, що використовуються при розробці мікропрограм. Мікропрограма описується на мікроасемблері системи COMPLEX і повинна пояснюватися відповідними коментарями.

Порядок виконання роботи

1. Налаштувати розроблену мікропрограму з використанням програмного емулятора в режимі трасування.

2. Виконати в автоматичному режимі записану в пам'яті програму з різними наборами операндів.

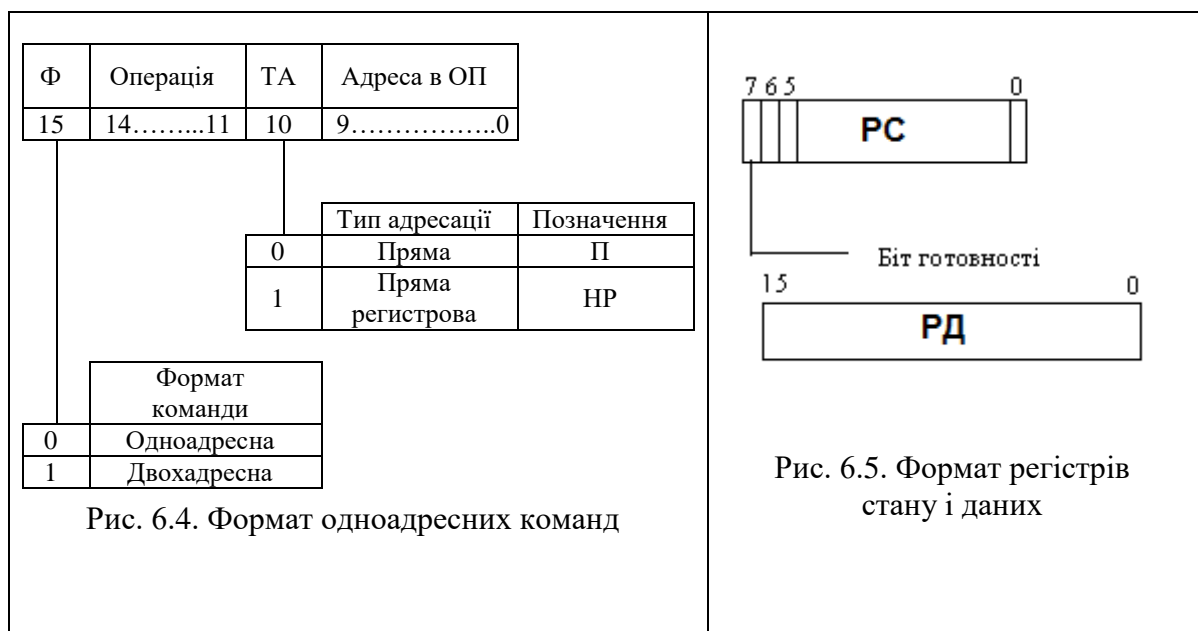


Табл.6.1

a3 a2 a1	Операція \Diamond
0 0 0	$X + Y$
0 0 1	$X - Y$
0 1 0	$X \& Y$
0 1 1	$X \vee Y$
1 0 0	$X \oplus Y$
1 0 1	$X \bar{\oplus} Y$
1 1 0	$\bar{X} \& Y$
1 1 1	$\bar{X} \vee Y$

Табл.6.2

a7	a6	a5	Адреси	
			портів	ЗП
			РС	РД
0	0	0	02Н	04Н
0	0	1	12Н	14Н
0	1	0	22Н	24Н
0	1	1	32Н	34Н
1	0	0	42Н	44Н
1	0	1	52Н	54Н
1	1	0	62Н	64Н
1	1	1	72Н	74Н

Табл. 6.3

a4	Тип адресації
0	Пряма
1	Пряма реєстрова

Табл. 6.4

a2	a1	Спосіб множення
0	0	1-й
0	1	2-й
1	0	3-й
1	1	4-й

3. Зробити висновки по роботі. У висновку необхідно вказати, які перетворення виконуються на програмному, мікропрограмному і апаратному рівнях, як здійснюється взаємодія між різними рівнями перетворення інформації.

Контрольні питання

1. Охарактеризуйте етапи виконання команд основної групи, передачі керування, введення-виведення, системних.
2. Охарактеризуйте основні способи адресації операндів з використанням і без використання РЗП.
3. Як забезпечити правильне зчитування і запис даних у пам'ять з урахуванням швидкодії пам'яті?
4. Яким чином можна керувати записом інформації в RA і RB, навіщо використовуються зазначені регістри?
5. Поясніть призначення директив мікроасемблера, які визначають роботу з пам'яттю.
6. Як забезпечити обмін даними між процесором і ЗП в програмному режимі?

Список літератури

- 1 Арифметичні та управляючі пристрої цифрових ЕОМ: Навчальний посібник / В.І.Жабін, І.А.Жуков, І.А.Клименко, Стиренко С.Г. – К.: ВЕК +, 2008. – 176 с.
2. Жабин В.И., Жуков І.А., Ткаченко В.В., Клименко І.А. Мікропроцесорні системи: Навчальний посібник. – К. Видавництво «СПД Гуральник», 2009. – 492 с.
3. Булгаков С.С., Мещеряков В.М., Новоселов В.В., Шумилов Л.А. Проективання цифрових систем на комплектах мікропрограмувальних ВІС. - М.: Радіо і зв'язок, 1984. - 240 С.
3. Жабін В.І., Ткаченко В.В. Цифрові автомати. Практикум. – К.: ВЕК+, 2004.– 160 с.