Міністерство освіти і науки України

Національний технічний університет України
«Київський політехнічний інститут»

АРХІТЕКТУРА КОМП'ЮТЕРІВ

Методичні вказівки до виконання лабораторних робіт для студентів напряму підготовки 6.050102 «Комп'ютерна інженерія»

Рекомендовано
Вченою радою ФІОТ НТУУ «КПІ»
(___.___.201_ р., протокол № __)

Київ
НТУУ «КПІ»

2015

ЗМІСТ

ВСТУП

Методичні вказівки містять матеріали до виконання лабораторних робіт з дисципліни «Архітектура крмп'ютерів» для студентів напряму підготовки 6.050103 «Програмна інженерія».

Комплекс лабораторних робіт, призначений для закріплення знань та одержання практичних навичок студентами в області проектування пристроїв компютерів. Для виконання кожної роботи надано теоретичні відомості, рекомендації та приклади проектування пристроїв компютерів, посилання на додаткову літературу та питання для контролю знань за відповідною тематикою.

Кожній лабораторній роботі повинна передувати самостійна підготовка студентів, в процесі якої вони докладно вивчають опис практичної роботи, відповідні розділи конспекту лекцій та літературні джерела. В процесі підготовки складається звіт, в якому повинні бути відображені всі пункти теоретичного завдання, а також заготовлені для виконання експериментальної частини практичної роботи таблиці, алгоритми, схеми і таке інше. Перед початком лабораторної роботи результати підготовки перевіряються викладачем. Під час такої перевірки студент повинен представити заготовлений звіт і відповісти на контрольні питання.

Перед початком наступного заняття в лабораторії студент представляє викладачеві повністю оформлений звіт за виконану роботу. Звіт повинен містити короткі теоретичні відомості, необхідні для виконання завдання, одержані схеми, алгоритми, таблиці і т.і, а також висновки по результатам роботи.

Залік за виконання лабораторої роботи студент одержує після співбесіди за тематикою виконаної роботи.

1. ПРОГРАМНИЙ КОМПЛЕКС ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Програмний комплекс ПРОГМОЛС 2.0 (**ПРОГ**рама **МО**делювання **Л**огічних **С**хем) 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 лаборкторні роботи 1 і 2. Вигляд моделі показано на рис. 1.1.

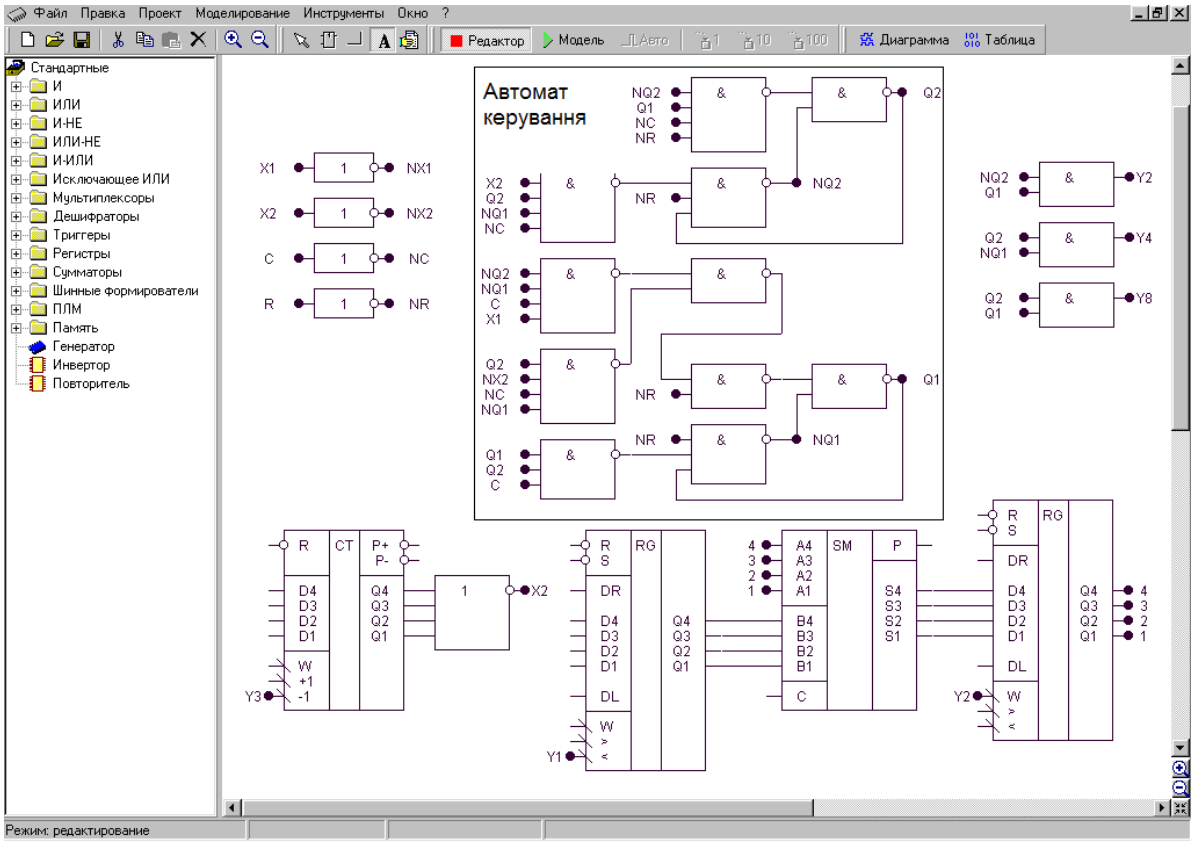


Рис. 1.1. Зображення моделі операційного пристрою з автоматом керування

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 сформувати 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 мікрооперaціями.

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 частин (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.

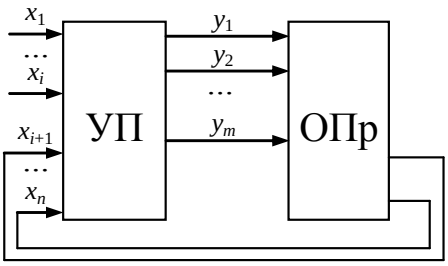


Рис. 2.1. Загальна структура АЛП

Побудова таких АЛП відбувається за наступними етапами:

1. Для кожної операції будується операційна схема та функціональний мікроалгоритм (Ф-микроалгоритм). 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-микроалгоритм) виконання заданих операцій, що враховує спосіб управління мікрооперація-ми на вузлах ОПр.

5. Виконується синтез управляючого пристрою.

6. Складається функціональна та принципова схеми всього АЛП.

Приклад. Побудувати схему АЛП для реалізації операції множення чисел за першим способом. Синтезувати схему, що дозволяє обчислити добуток $Z=Y\times X$ двох правильних дробів $Y= 0, y_1, y_2 \ldots y_n$ та $X= 0, x_1, x_2 \ldots x_n$.

Виконання завдання

Операційна схема, що реалізує перший спосіб множення, подана на рис. 2.2, де *RG*1 – регістр накопичення суми часткових добутків, *RG*2 – регістр множника, *RG*3 – регістр множеного, *RG*4 (*CT*) – лічильник циклів, *SM* – комбінаційний суматор. *DR* – вхід заповнення старшого розряду при зсуві вправо. Регістри *RG*1 та *R*G2 реалізують мікрооперації зсуву, лічильник *RG*4 дозволяє формувати ознаку нуля. За нульовим вмістом регістру *RG*4 результат обчислення є сформованим в регістрах *RG*1 та *R*G2.

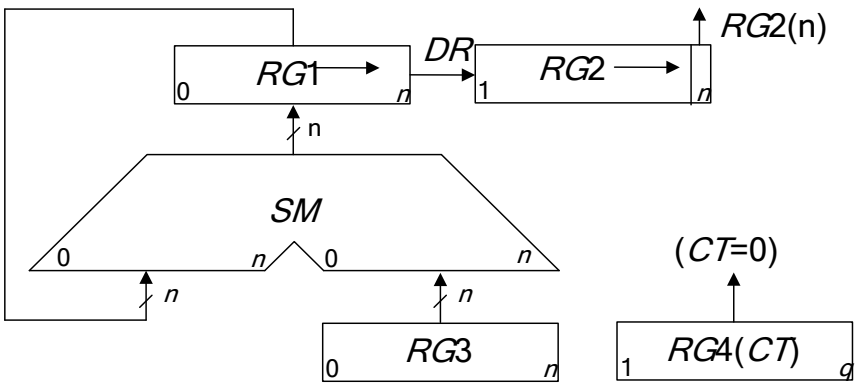


Рис. 2.2. Операційна схема множення

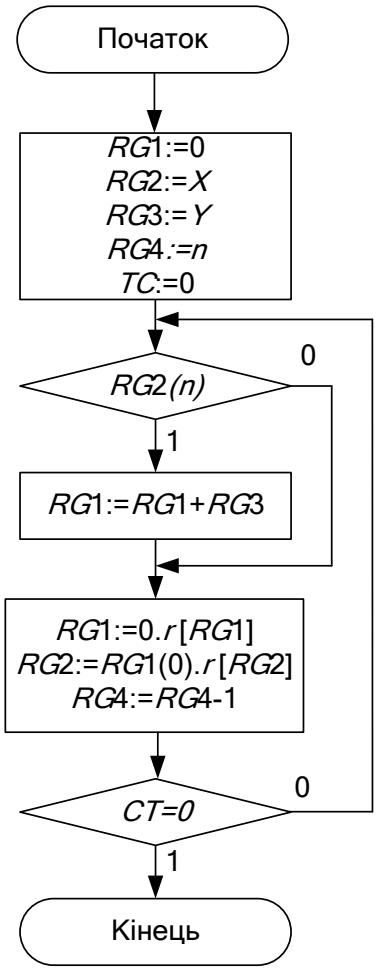


Рис. 2.3. Ф-мікроалгоритм множення чисел

Для розробленої операційної схеми побудуємо Ф-мікроалгоритм (рис. 2.3), де *RG*2*(n)* – значення молодшого розряду регістру *RG*2.

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

№ такту	*RG*1	*RG*2	*RG*2*(n)*	*RG*3	*RG*4	МО
ПС	0000	101	1	0111	11	Початковий стан
1	0000 +0111 0111 0011	110	0	0111 0111	10	*RG*1+ *RG*3 *RG*1→, *RG*2→, *RG*4 – 1; *CT≠0*
2	0001	111	1	0111	01	*RG*1→, *RG*2→, *RG*4 – 1; *CT≠0*
3	0001 +0111 1000 0100	011	0 1	0111 0111	00	*RG*1+ *RG*3 *RG*1→, *RG*2→, *RG*4 – 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.

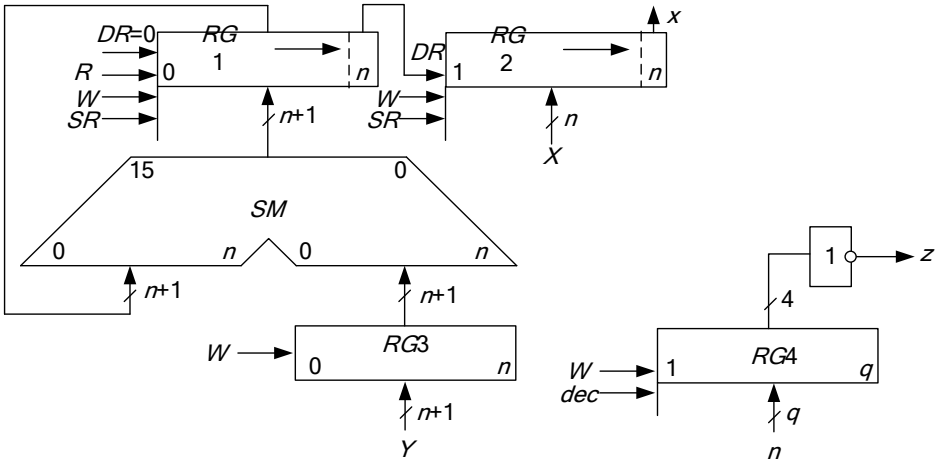


Рис. 2.4. Функціональна схема операційного пристрою

Таблиця 2.2. Таблиця управляючих сигналів

Вузол	Мікрооперація	Управляючий сигнал
*RG*1	Скидання в нуль	*R*
	Запис	*W*
	Зсув вправо	*SR*
*RG*2	Запис	*W*
	Зсув вправо	*SR*
	Старший розряд при зсуві вправо	*DR*
*RG*3	Запис	*W*
*RG*4	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 завжді 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 стану лічильника після декременту в наступному такті, коли новий стан лічильника вже буде встановлено.

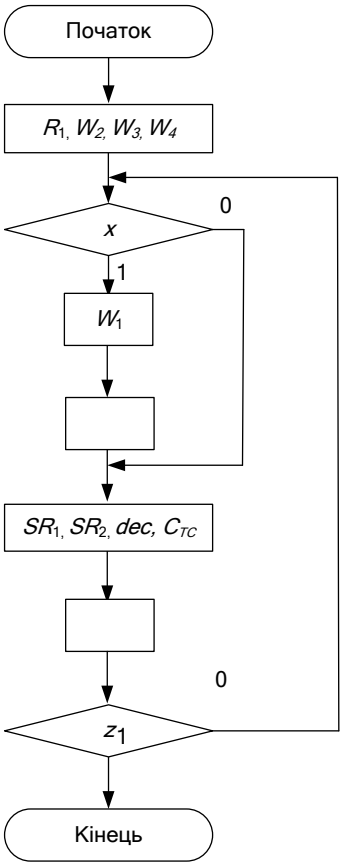


Рис. 2.5. Структурний мікроалгоритм

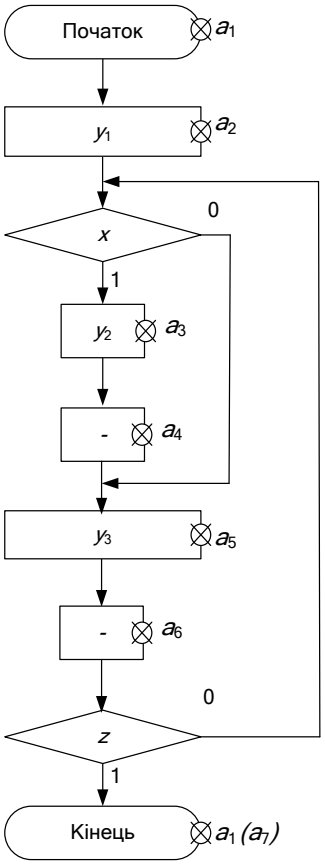


Рис. 2.6. Закодований структурний мікроалгоритм

Таблиця 2.3. Кодування сигналів управління

Управляючі сигнали	Код
R_1	y_1
W_2	
W_3	
W_4	
R_{TC}	
W_1	y_2
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-микроапгоритм 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

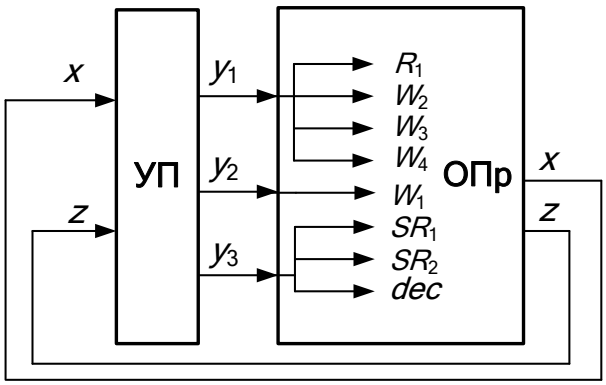


Рис. 2.7. Узагальнена структурна схема АЛП

функціональних вузлів ОПр.

Операцію множення AB можно 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.

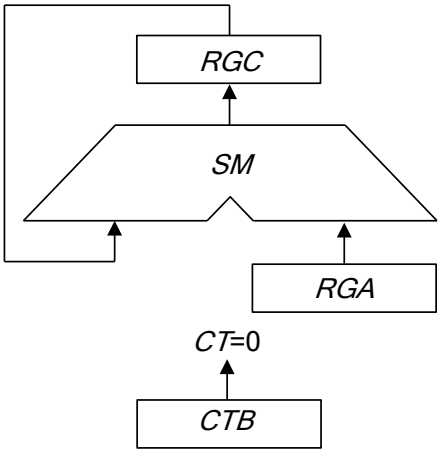


Рис. 2.8. Операційна схема пристрою для обчислення функції

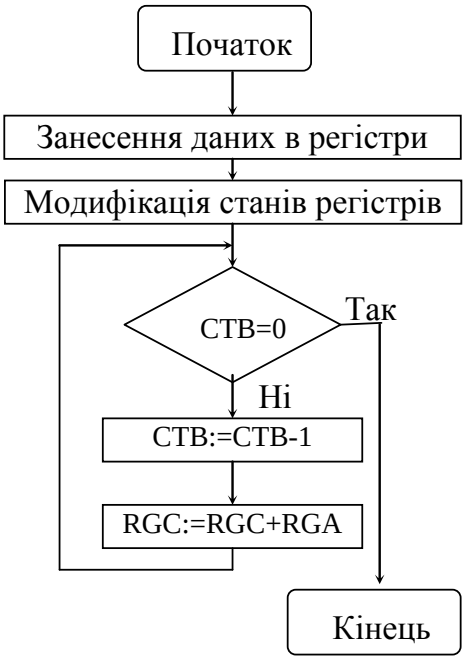


Рис. 2.9. Змістовний мікроалгоритм

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*.

Підготовка до лабораторного заняття

1. Варіанти завдання визначаються молодшими розрядами $a_7,\ldots,a_1$ двійкового номера залікової книжки відповідно з табл. 2.7-2.9.

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 використати 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 завданнь

a_7	a_6	a_5	a_4	Спосіб множення
0	0	0	0	1-й
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	Функція
0	1	0	0	$D=2C+4AB$
0	1	0	1	$D=C+4AB$
0	1	1	0	$D=2C+0,5AB$
0	1	1	1	$D=A(B-1)+0,5C$
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

Опис методів множення чисел наведено в додатку А, а етапи синтезу автоматів – в додатку Б.

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 виходи автомата до входів операційного пристрою не підключати. Налагодити 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; табли-

ці 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 короткочасних 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 управляючи 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. *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.

4. *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.

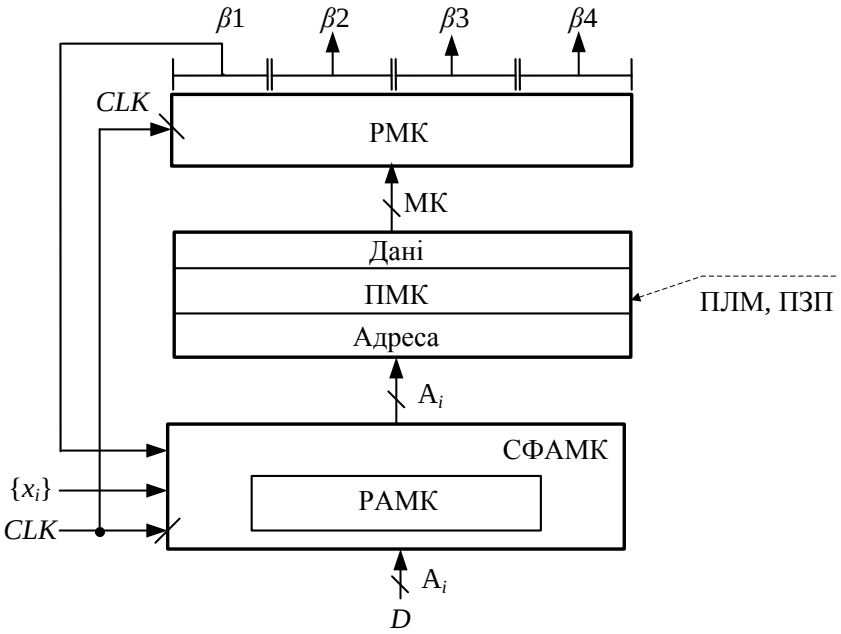


Рис. 3.1. Структурна схема БМУ

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.

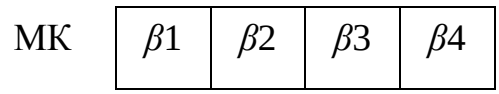


Рис. 3.2. Формат мікрокоманди

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 функції, 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 використають для контролю апаратури.

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

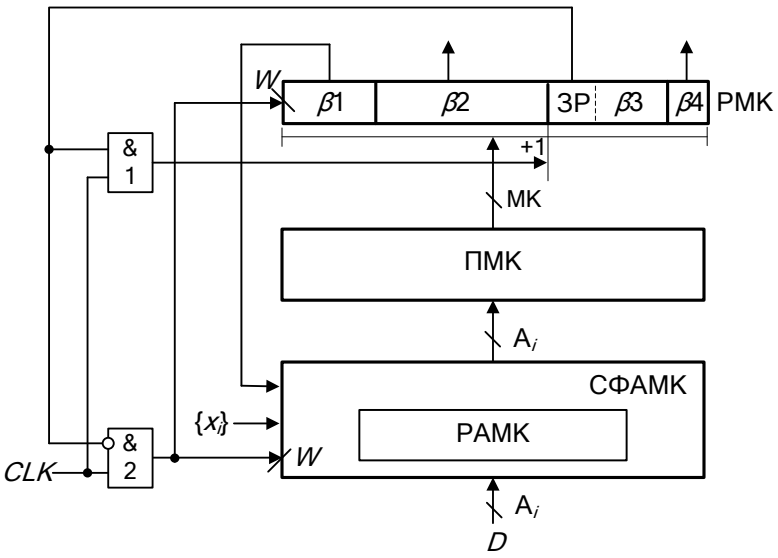


Рис. 3.3. Формування тривалості управляючих сигна-

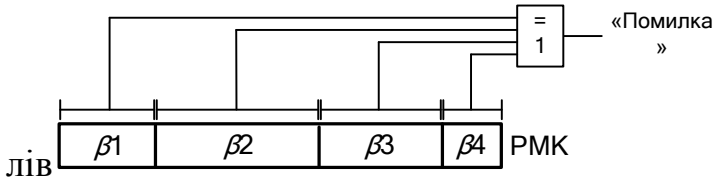
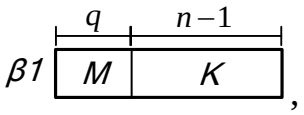


Рис. 3.4. Схема контролю слова МК на парність

БМУ з примусовою адресацією

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



де M – поле управління мультиплексором;

q – довжина поля управління мультиплексором;

K – константа, що визначає адресу наступної мікрокоманди;

n – розрядність адреси мікрокоманди.

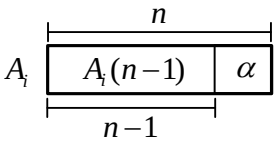
Довжина поля управління мультиплексором визначається за формулою:

$$q =]\log_2(k+2)$$

де k – кількість зовнішніх умов.

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

Формат адреси мікрокоманди має наступний вигляд:



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.

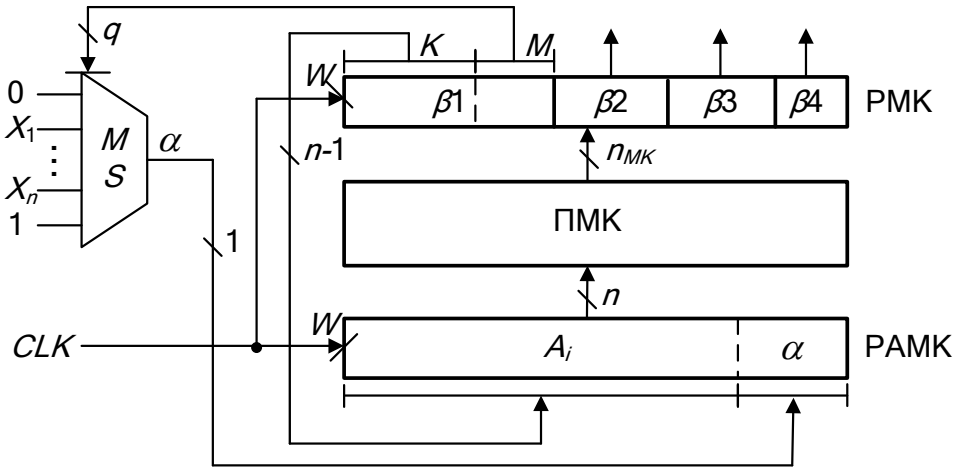


Рис. 3.5. Структурна схема БМУ з примусовою адресацією

БМУ з відносною адресацією

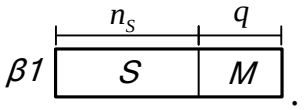
За відносної адресації адреса наступної МК визначається за формулою:

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

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:

$$n_S =]\log_2 N[+ 1,$$

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

Структурна схема БМУ наведена на рис. 3.6.

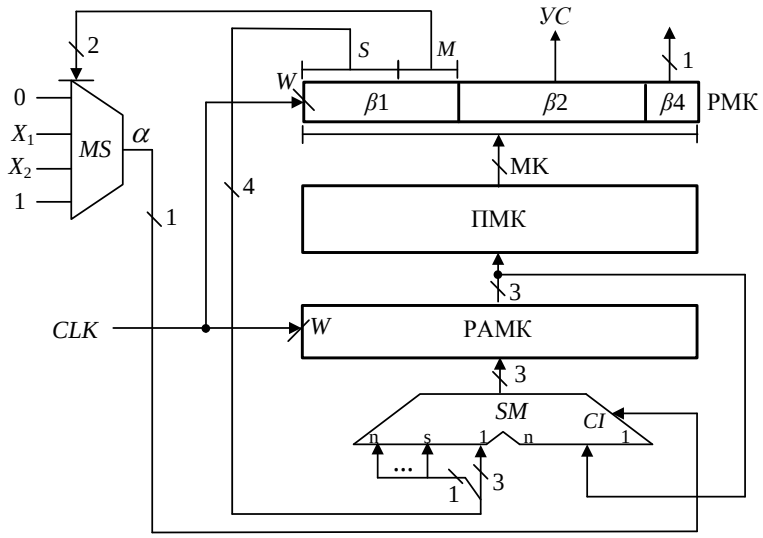


Рис. 3.6. Структурна схема БМУ з відносною адресацією

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 – 0007*h*;
- 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.

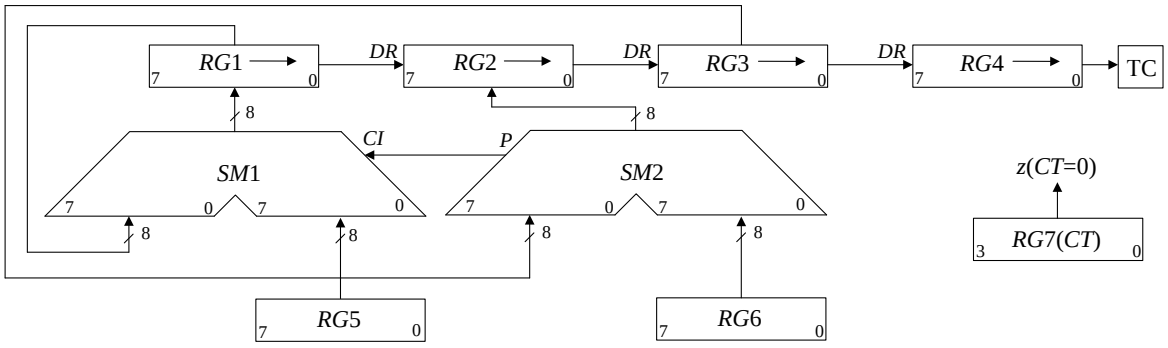


Рис. 3.7. Структурна схема пристрою множеня

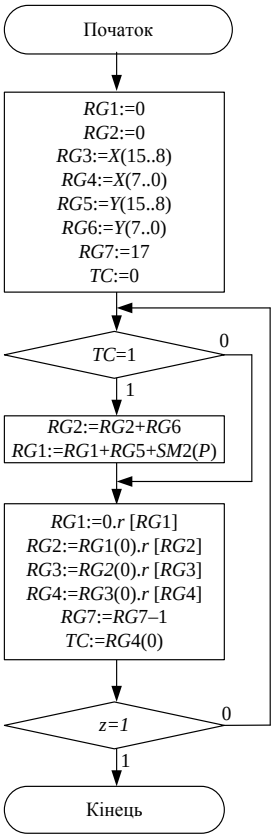


Рис. 3.8. Змістовний мікроалгоритм

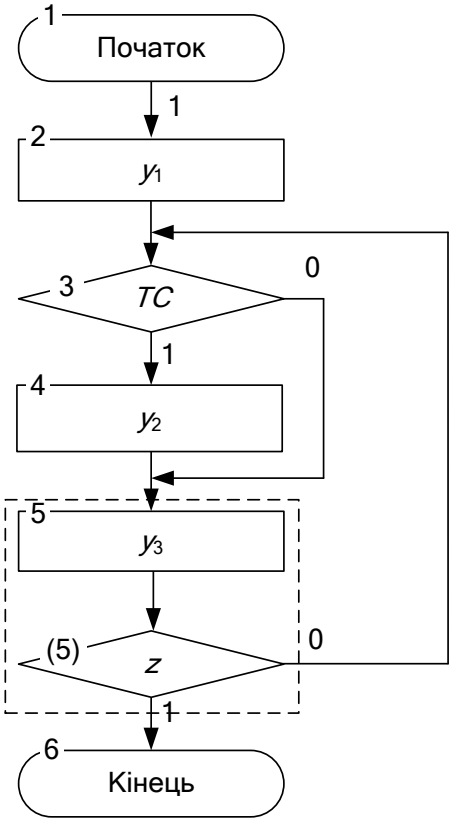


Рис. 3.9. Закодований алгоритм управління пристроєм множення

Визначимо формат зони $\beta 1$:

$$n_a =]\log_2 16[= 4;\ n_K = 3;$$

$$n_M =]\log_2 4[= 2;\ n_{\beta 1} = 5.$$

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):

$$n_{\beta 2} =]\log_2 4[= 2.$$

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

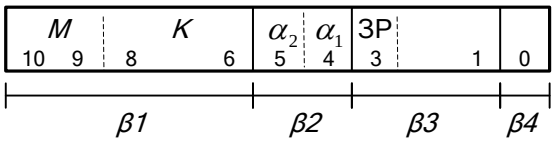
За виразом (3.3) розрахуємо довжину зони $\beta 3$:

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

$$n_{\beta 3} =]\log_2 3[+ 1 = 3.$$

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).

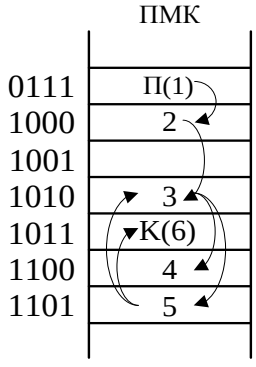


Рис. 3.10. Розміщення мікрокоманд в ПМК

Карта програмування БМУ наведена у табл. 3.3.

Таблиця 3.3. Карта програмування БМУ

№ МК	Адреса	β1		β2	β3		β4
		K	*M*	α_2 α_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.

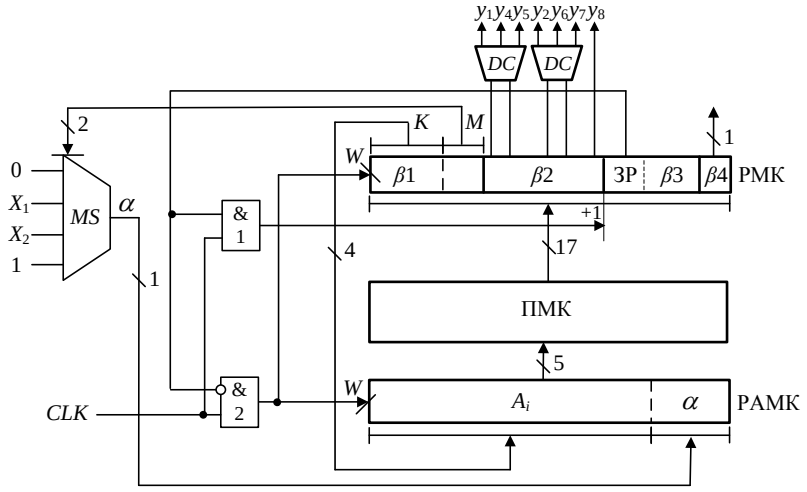


Рис. 3.11. Схема БМУ з примусовою адресацією

Підготовка до роботи

1. Варіанти завдання визначаються молодшими розрядами a7,…,a1 двійкового номера залікової книжки..

2. Розробити структурну схему операційного пристрою та змістовний мікроалгоритм обробки додатних чисел відповідно до

завдання наведеного 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 структурный схемі повинні бути зазначені розрядність регістрів та шин.

3. Розробити функціональну схему операційного пристрою.

4. 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	$D=2C+4AB$	4
0	0	1	$D=2A(B+1)+0,5C$	4
0	1	0	$D=A(B+1)+2C$	4
0	1	1	$D=A(B-1)+0,5C$	4
1	0	0	1-й спосіб множення	7
1	0	1	2-й спосіб множення	7
1	1	0	3-й спосіб множення	7
1	1	1	4-й спосіб множення	7

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

6. 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
Інщі 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 $\beta 2$ при горизонтальному способі мікропрограмування?

11. Назвіть способи формування структури зони $\beta 2$, переваги та недоліки кожного із способів.

12. Як визначити довжину зони $\beta 3$ формування управляючих сигналів БМУ при асинхронному способі управління виконанням МК?

13. Назвіть способи формування структури зони $\beta 1$, переваги та недоліки кожного із способів.

14. Як скоротити довжину зони $\beta 1$ при застосуванні примусової адресації МК?

15. Наведіть приклад застосування зони $\beta 4$.

Список літератури

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

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

4. *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.

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

4. ЗАГАЛЬНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

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.

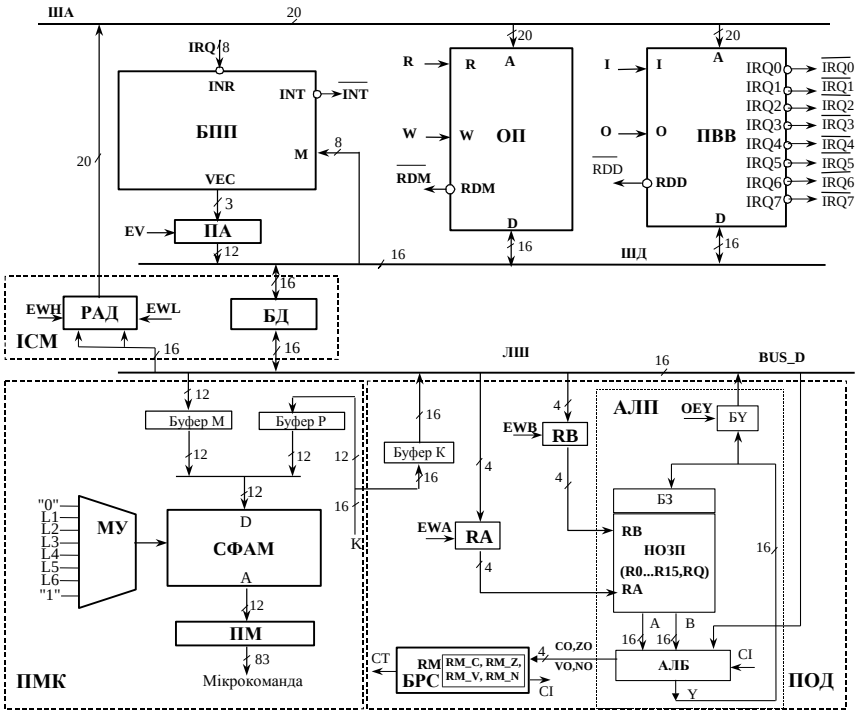


Рис. 4.1. Структура ЕОМ

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 регістра стану (БРС). ПМК містить мультиплексор умов (МУ), схему формування адреси мікрокоманди (СФАМ), пам'ять мікрокоманд. Інтерфейс ІСМ містить регістр адреси (РАД) і буфер даних (БД).

Програмний эмулятор настроєний на 16-розрядні регістри 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 регістр 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 регістр даніх (РД), який входить до складу ВПП. Готовність ПВВ до обміну визначається за допомогою його регістру стану (РС). Розряд РС за номером 7 містить біт "Готовність". При зверненні до ВПП у програмному режимі спочатку необхідно перевірити його готовність. Регістри РД та РС мають адреси у адресному просторі процесора. Звертання до регістрів 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, ПВВ можуть формувати сигнали запиту переривань IRQ_i (i=0,…,7), якщо вони готові до обміну даними. Дозвіл на формування запиту на переривання забезпечується записом одиниці у 6-й розряд РС (розряд дозволу переривання).

БПП забезпечує аналіз пріоритетів сигналів IRQ_i, яки потрапляють на входи INR, формування сигналу INT вимоги переривання і вектору переривання VEC. Вхід M дозволяє маскування сигналів переривань від ПВВ. 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 ПВВ входить блок переривань, який формує сигнал IRQ_i. Всі блоки зв'язані приоритетним 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 приоритетного ланцюжка) та INTA (сигнал підтвердження переривання). Після цього ПВВ 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...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 A і B.

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;

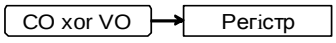
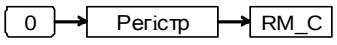
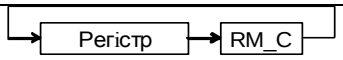
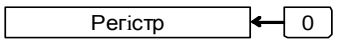
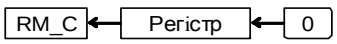
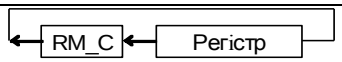
ZO - нульовий результат;

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 мікрооперaціях 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, CT та ін.). З виходу CT 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 мікроассемблері). 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 використову-

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 мікроассемблера 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
1111111111111111%    \ 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 мікрооперaціях.

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 указується 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%,
                            \ то 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; cem_v; \ забезпечить завантаження
cem_n; cem_z           \тільки розряду RM_C.
```

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

лом <У> позначені позиції запису умови, а символом <А> – 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 <У>,<А>	Умовний перехід по зазначеній адресі (мітки)
CJS <У>,<А>	Умовний перехід до мікропідпрограми по зазначеній адресі (мітки)
CRTN <У>	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
OEY	Видача результату Y з БУ на ЛШ
EWA	Запис в RA
EWB	Запис в RB
EV	Видача вектора з ПА на ЛШ

Директиви. Мікроассемблер 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 **ACCEPT** дозволяє встановити початковий стан регістрів R0,...,R15, RA,RB і RM. Форма запису

ACCEPT <регістр>:<значення>

Приклади.

```
ACCEPT RA:12
ACCEPT R4:0fffh
ACCEPT RM: 0101%
```

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

- тип пристрою (I - вводу, O - 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.

```
accept 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

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

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

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

```
ACCEPT 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 l1,…,l6 ПМК і логічних умов використовується директива

LINK <ім'я_входу>:<умова>

Приклади

link L1:rdm

LINK l2:RDD

link l3:ct

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

LINK EWH : <номер_розряду>

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

Приклад

LINK EWH : 16

Вказана директива забезпечує зв'язки між ЛШ і РАД так, що по сигналу EWH 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 M: 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 ПВВ.

Приклад

link vec[2]: 0CDEFh

У цьому випадку на виходи 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 мікрооперaціями, 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 операторной 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;oey;ewh;}    \РАД[19...16]:=0
{cont;or nil,r7,z;oey;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 l1: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;oey;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;oey;ewl;}      \ запис адреси в РАД
   ll2  {cjp rdm,ll2;w;or nil,r1,z;oey;}  \запис результату в пам'ять
{}                                        \кінець мікропрограми
```

Приклад

Мікропрограма настроювання БПП і читання вектора переривання з ПА в АЛП.

```
link vec[2]: 0CDEFh     \ завдання в ПА адреси для переривання від
                        \ 2-го ПВВ
link L[4]:INT           \ підключення виходу INT до входу l4 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.

```
ACCEPT      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
ACCEPT      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 приоритетного ланцюжку. У противному випадку ВПП може працювати в режимі переривань тільки разом з централізованим БПП.