МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

АРХІТЕКТУРА КОМП'ЮТЕРІВ

Методичні вказівки до виконання лабораторних робіт

для студентів напряму підготовки

«6.050103 – Програмна інженерія»

кафедри обчислювальної техніки

Рекомендовано
Вченою радою факультету
інформатики та обчислювальної
техніки НТУУ «КПІ»
Протокол № _ від
__._____.2015р.

Київ

НТУУ «КПІ»

2015

Архітектура комп'ютерів 1. Арифметичні та управляючі пристрої. Методичні вказівки до виконання лабораторних робіт. [Текст] / Уклад.: В.І.Жабін, В.В. Ткаченко, І.А. Клименко – К.: НТУУ «КПІ», 2015. – 74 с.

Методичні вказівки призначені для студентів напряму підготовки 6.050102 «Комп'ютерна інженерія» професійного спрямування «Комп'ютерні системи та мережі» кафедри обчислювальної техніки всіх форм навчання. Наведено завдання та методичні вказівки до виконання лабораторних робіт з дисципліни «Архітектура комп'ютерів – 1. Арифметичні та управляючі пристрої», питання для самоконтролю, список необхідної літератури.

Укладачі: В.І. Жабін, д.т.н., професор
В.В. Ткаченко, к.т.н., доцент
І.А. Клименко, к.т.н., доцент

Рецензент: Теленик С.Ф., доктор технічних наук, професор
завідувач кафедри
автоматики і управління в технічних системах

За редакцією укладачів

ЗМІСТ

ВСТУП

В методичних рекомендаціях узагальнені матеріали до виконання лабораторних робіт до курсу «Архітектура комп'ютерів» для студентів напряму підготовки 6.050102 «Комп'ютерна інженерія» професійного спрямування «Комп'ютерні системи та мережі» кафедри обчислювальної техніки всіх форм навчання.

Навчальний матеріал методичних вказівок відповідає одному кредитному модулю дисципліни «Архітектура комп'ютерів», а саме «Архітектура комп'ютерів – 1. Арифметика та управління».

Матеріал лабораторних робіт присвячений вивченню принципів організації та дослідженню арифметико-логічних і управляючих пристроїв, а також побудові функціональних та принципових електричних схем цифрових ЕОМ. В методичних вказівках приведені завдання та рекомендації до виконання лабораторних робіт. До лабораторних робіт надані теоретичні відомості, необхідні для виконання кожної роботи, приклади проектування, рекомендації до виконання завдання та саме завдання. До кожної лабораторної роботи надаються контрольні питання що застосовуються для контролю знань за відповідною тематикою.

Методичні рекомендації містять список рекомендованої додаткової літератури.

Виконання лабораторних робіт дозволяє розширити і закріпити теоретичні знання з дисципліни, опанувати навички проектування і дослідження арифметичних та управляючих пристроїв цифрових ЕОМ. Кожній лабораторній роботі повинна передувати самостійна підготовка студентів, в процесі якої вони докладно вивчають опис лабораторної роботи, відповідні розділи посібника, конспекту лекцій та літературні джерела. В процесі підго-

товки складається звіт про лабораторну роботу, в якому повинні бути відображені всі пункти теоретичного завдання, а також заготовлені для виконання експериментальної частини лабораторної роботи таблиці, осі для часових діаграм і таке інше. Перед початком лабораторної роботи результати підготовки перевіряються викладачем. За цим студент повинен представити заготовлений звіт і відповісти на контрольні питання. Перед початком наступного заняття в лабораторії студент представляє викладачеві цілком оформлений звіт за попередньою роботою. Звіт повинен містити короткі теоретичні відомості, необхідні для виконання завдання, відповіді на контрольні питання, усі схеми, формули, таблиці, діаграми, графіки, отримані при виконанні завдання та в процесі експериментального дослідження схем, а також висновки за роботою. Залік за виконання лабораторної роботи студент одержує після співбесіди за тематикою виконаної роботи.

Теоретичний матеріал та зміст лабораторних робіт відповідають навчальному плану дисципліни «Архітектура комп’ютерів».

Автори методичних вказівок вдячні рецензентам за слушні зауваження, що дозволили покращити якість матеріалу.

ПЕРЕЛІК СКОРОЧЕНЬ

АЛБ – Арифметико-логічний блок

АЛП – Арифметико-логічний пристрій

БМУ – Блок мікропрограмного управління

БУ – Блок управління

ВМ – Вертикальне мікропрограмування

ГМ – Горизонтальне мікропрограмування

ДК – Доповнювальний код

ЕОМ – Електронно-обчислювальні машини

ЗК – Зворотний код

МА – Мікроалгоритм

МК – Мікрокоманда

МО – Мікрооперація

МП – Мікропрограма

НОЗП – Надоперативний запам’ятовуючий пристрій

ОПр – Операційний пристрій

ОП – Основна пам’ять

ОС – Операційна схема

ОТ – Обчислювальна техніка

ПЗП – Постійний запам’ятовуючий пристрій

ПК – Прямий код

ПМК – Пам’ять мікрокоманд

ПЛМ Програмовані логічні матриці

УГП – Умовне графічне позначення

УП – Управляючий пристрій

УС – Управляючий сигнал

ЛАБОРАТОРНА РОБОТА №1
АРИФМЕТИКО-ЛОГІЧНІ ПРИСТРОЇ З РОЗПОДІЛЕНОЮ ЛОГІКОЮ

Мета роботи: вивчити основні методи множення чисел у прямих кодах і способи їх апаратної реалізації, одержати навички в проектуванні й налагодженні схем управління операційними пристроями з розподіленою логікою.

Теоретичні відомості [1, 2, 3, 4, 5, 6]

Додаткові теоретичні відомості

Синтез арифметико-логічних пристроїв з розподіленою логікою

За структурою розрізняють АЛП з розподіленою та зосередженою логікою (інакше 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. 2.2): управляючий пристрій, що забезпечує формування всіх управляючих сигналів; операційний пристрій, 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-микроалгоритм). Рекомендується обирати такі мікроал-

горитми виконання операцій, що краще сполучаються, тобто вимагають однакового напрямку зсувів в регістрах, однакову розрядність регістрів, одні й ті самі джерела операндів суматорів і таке інше.

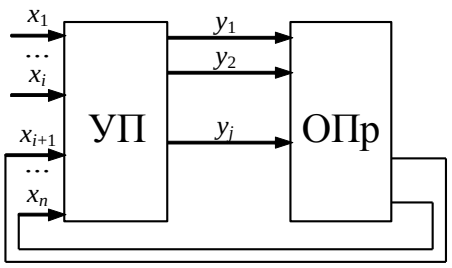


Рис. 2.2. Загальна структура АЛП

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

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

4. Складається закодований структурний мікро алгоритм (С-мікроалгоритм) виконання заданих операцій.

5. Виконується синтез управляючого пристрою.

6. Складається функціональна та принципова схеми АЛП.

Приклад 1. 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 $X= 0, x_1, x_2 \ldots x_n$. Вважати, що розрядність дробів $n = 16$.

Виконання завдання

Операційна схема, що реалізує перший спосіб множення, подана на рис. 2.3, де *RG*1 – регістр накопичення суми часткових добутків, *RG*2 – регістр множника, RG3 – регістр множеного, *RG*4 (*CT*) – лічильник циклів, *TC* – тригер переносу, *SM* – комбінаційний суматор. Регістри *RG*1 та *RG*2 реалізують мікрооперації зсуву, лічильник *RG*4 дозволяє формувати ознаку нуля

– що визначає закінчення обчислення добутку. За нульовим вмістом регістру *RG*4 результат обчислення формується в регістрах *RG*1 та *RG*2.

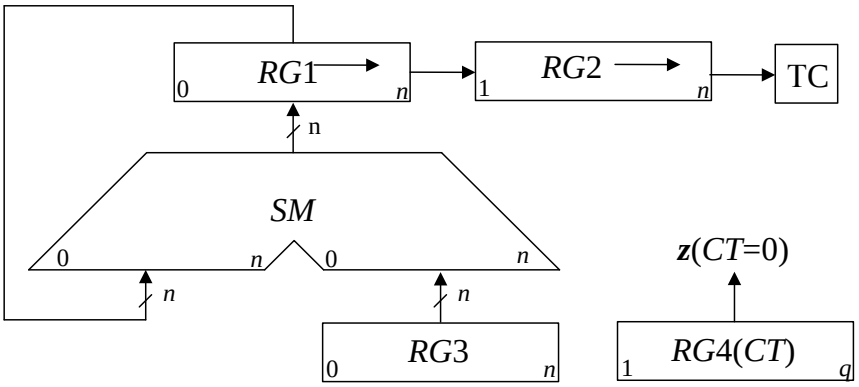


Рис. 2.3. Операційна схема множення

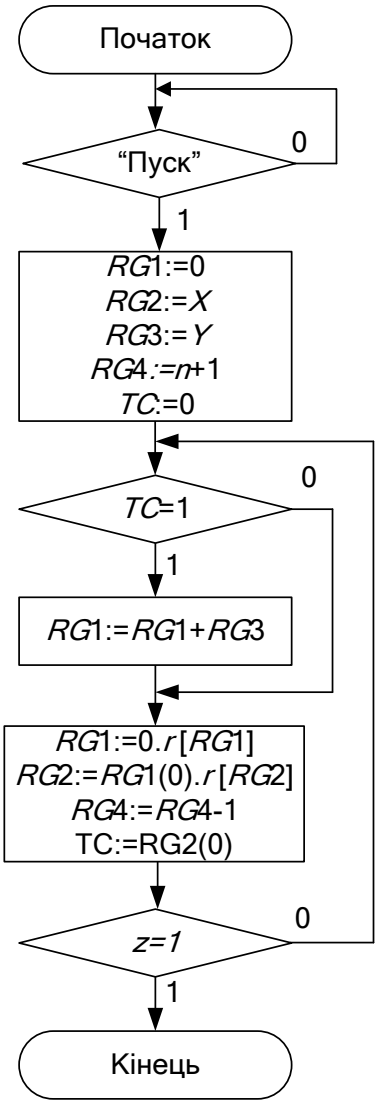


Рис. 2.4. Ф-мікроалгоритм

Для розробленої операційної схеми множення чисел побудуємо Ф-мікроалгоритм. Припустимо, що ОПр входить до складу АЛП із централізованим управлінням, отже робота цього блоку розпочинається із надходження сигналу “Пуск” від центрального блоку управління. Функціональний микроалгоритм зображений на рис. 2.4, де *TC* – стан тригера переносу, *z* – значення ознаки нуля в лічильнику циклів *RG*4.

Логічне моделювання потактової роботи ОПр приведене в табл. 2.1

Значення операндів:

$Y = 5_{10} = 0101_2$;

$X = 7_{10} = 0111_2$;

$Z = 35_{10} = 00100011_2$.

Розрядність дробів $n = 4$.

Таблиця 2.1. Логічне моделювання роботи ОПр

№ такту	*RG*1	*RG*2	*TC*	*RG*3	*RG*4	*z*	МО
ПС	0000	0101	0	0111	0101	0	Початковий стан
1	0000	0010	1	0111	0100	0	*RG*1→, *RG*2→, *RG*4 – 1; *z* = 0
2	0000 +0111 0111	0010	1	0111	0100	0	*RG*1+ *RG*3
	0011	1001	0	0111	0011	0	*RG*1→, *RG*2→, *RG*4 – 1; *z* = 0
3	0001	1100	1	0111	0010	0	*RG*1→, *RG*2→, *RG*4 – 1; *z* = 0
4	0001 +0111 1000	1100	1	0111	0010	0	*RG*1+ *RG*3
	0100	0110	0	0111	0001	0	*RG*1→, *RG*2→, *RG*4 – 1; *z* = 0
5	0010	0011	0	0111	0000	1	*RG*1→, *RG*2→, *RG*4 – 1; *z* = 1

На підставі ОС множення та Ф-мікроалгоритму складемо перелік управляючих сигналів для всіх функціональних частин ОПр та побудуємо функціональну схему.

Функціональна схема ОПр зображена на рис. 2.5. Перелік управляючих сигналів наведений в табл. 2.2.

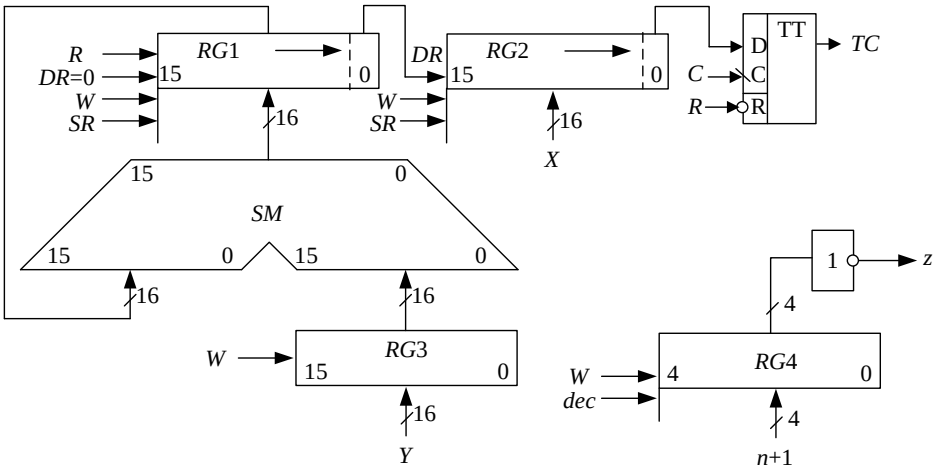


Рис. 2.5. Функціональна схема операційного пристрою

Таблиця 2.2. Таблиця управляючих сигналів

Елемент	Мікрооперація	Управляючий сигнал
*RG*1	Скидання	*R*
	Запис	*W*
	Зсув вправо	*SR*
	Заповнення старшого розряду при зсуві вправо	*DR*
*RG*2	Запис	*W*
	Зсув вправо	*SR*
	Старший розряд при зсуві вправо	*DR*
*RG*3	Запис	*W*
*RG*4	Запис	*W*
	Декремент лічильника	*dec*
TC	Скидання	*R*
	Запис молодшого розряду множника у тригер переносу	*C*

За побудованою функціональною схемою будуємо функціонально-структурний мікроалгоритм (ФС-мікроалгоритм), що зображений на рис 2.6. Індекс указує до якої з функціональних частин пристрою множення належить управляючий сигнал.

Кодування сигналів управління та логічних умов наведене в табл. 2.3 – 2.4.

Для забезпечення перепаду сигналів управління SR_1, SR_2, dec, C_{TC} (вершину з цими сигналами охоплює петля рис. 2.6) необхідно ввести порожню додаткову вершину.

Закодований ФС-мікроалгоритм зображений на рис. 2.7, де управляючі сигнали та сигнали логічних умов відповідають рис. 2.6 та табл. 2.2 – 2.4.

Отриманий закодований ФС-микроапгоритм є вихідним для здійснення синтезу управляючого пристрою.

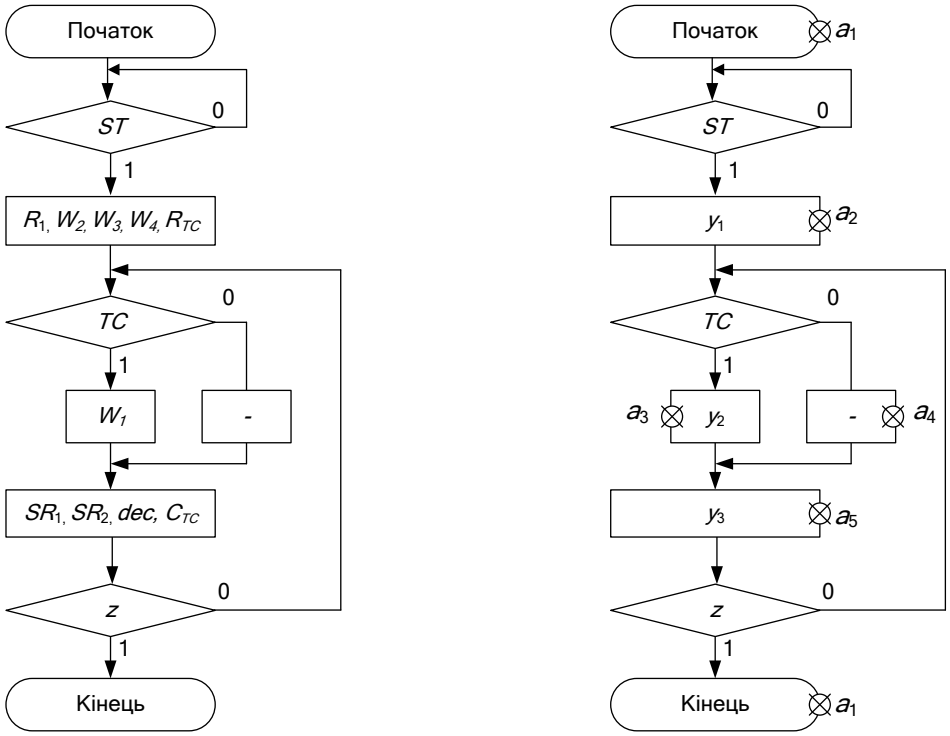


Рис. 2.6. Функціонально-структурний мікроалгоритм

Рис. 2.7. Закодований функціонально-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
Пуск	ST
Аналіз молодшого розряду множника	TC
Нульовий вміст лічильника	z

Отриманий закодований ФС-микроапгоритм є вихідним для здійснення синтезу управляючого пристрою.

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

Розмітка ФС-микроалгоритма для автомата Мура наведена на рис. 2.7. Стани автомата позначені символами a_i. Часова діаграма роботи управляючого пристрою зображена на рис. 2.8. Часова діаграма відповідає потактовій роботі ОПр для прикладу, виконаного в табл. 2.1.

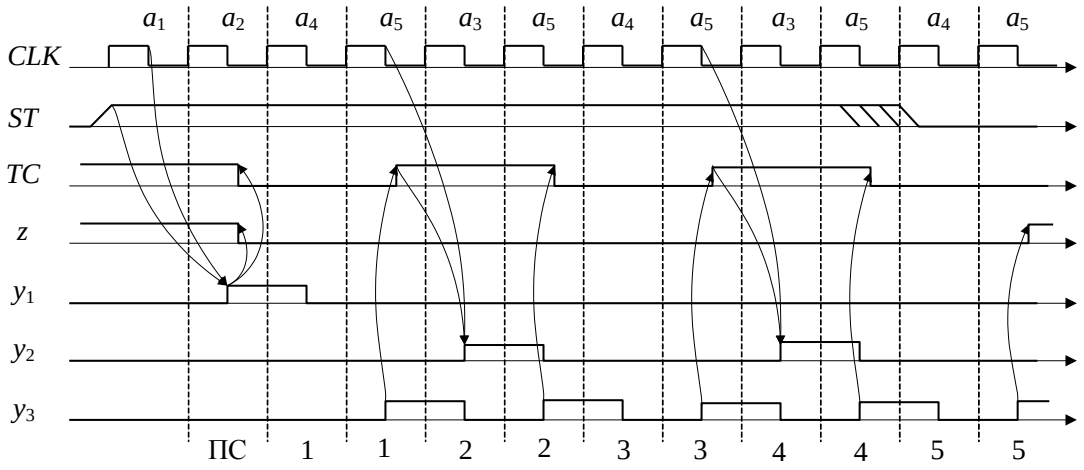


Рис. 2.8. Часова диаграма роботи пристрою управління

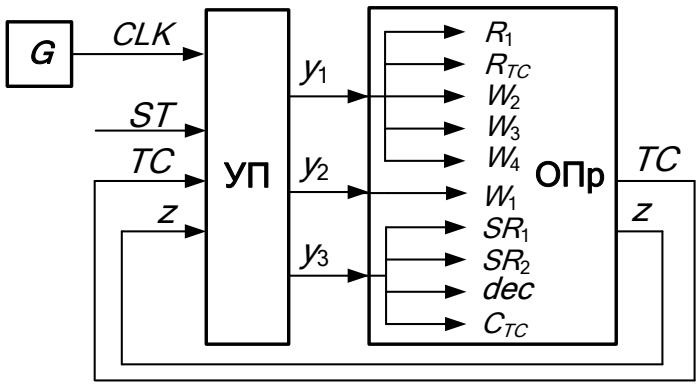


Рис. 2.9. Узагальнена структурна схема АЛП

На рис. 2.9 зображена узагальнена структурна схема АЛП множення. Управляючі сигнали з виходів пристрою управління підключаються до входів відповідних функціональних частин ОПр.

Схема електрична функціональна АЛП для множення додатних чисел наведена у додатку А. Опис функціональної схеми наведений у прикладі 7.1.

Підготовка до лабораторного заняття

1. Розробити структурну схему операційного пристрою та змістовний мікроалгоритм множення додатних чисел відповідно до завдання наведеного у табл. 2.7), де $a_6,\ldots,a_1$ – молодші розряди двійкового номера залікової книжки. 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.

2. Розробити функціональну схему операційного пристрою.

3. Виконати логічне моделювання роботи операційного пристрою за допомогою цифрової діаграми із зазначеними викладачем значеннями операндів.

4. Здійснити синтез пристрою управління, тип управляючого автомату обрати із табл.2.9. Пам’ять автомата реалізувати на тригерах, тип яких обрати з табл. 2.8. Ураховувати, що мікрооперації на регістрах виконуються за зворотним перепадом управляючих сигналів.

5. Побудувати часові діаграми роботи автомата для кожної комбінацій значень логічних умов.

Таблиця 2.7. Варіанти завдання

a_6	a_5	a_4	Спосіб множення	Розрядність операндів
0	0	0	1	16
0	0	1	2	8
0	1	0	3	16
0	1	1	4	8
1	0	0	1	8
1	0	1	2	16
1	1	0	3	8
1	1	1	4	16

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

Порядок виконання роботи

1. В моделюючій програмі ПРОГМОЛС 2.0 побудувати схему операційного пристрою для множення чисел 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 в синхронному режимі. Опис програмного комплексу ПРОГМОЛС 2.0 наведений у додатку М.

2. Підключити до управляючих входів операційного пристрою виходи автомата. Зробити комплексне налагодження схеми в синхронному режимі й 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. Що таке мікроалгоритм операції?
5. Визначте основне призначення арифметико-логічного пристрою в ЕОМ.
6. Наведіть типи арифметико-логічних пристроїв, та їх основні відмінності.

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

8. Що відображує операційна схема виконання операції?

9. Що відображує функціональна схема пристрою?

10. В чому відмінність функціонального та структурного мікроалгоритмів?

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

12. У чому відмінність автоматів Милі та Мура?

13. Намалюйте узагальнену структурну схему управляючого автомата.

14. Охарактеризуйте основні етапи проектування управляючого автомата.

15. Як перейти від змістовного мікроалгоритму до закодованого мікроалгоритму?

16. Як побудувати граф автомата?

17. Як здійснюється оцінка станів автомата?

18. Як визначити необхідну тривалість управляючих сигналів?

19. Від чого залежить кількість тригерів, необхідних для побудови пам’яті автомата?

20. Як скласти структурну таблицю автомата?

21. Складіть таблицю переходів для *JK*-, *RS*-, *T*- і *D*-тригерів. Наведіть їх умовне графічне позначення.

22. Чи можливий перехід автомата в стан, що непередбачений графом, при використанні тригерів із внутрішньою затримкою (тригерів, керованих рівнем сигналів)?

23. Коли можливе виникнення помилкових управляючих сигналів (що непередбачені графом автомата) і чим визначається їх тривалість?

24. Наведіть способи усунення короткочасних помилкових управляючих сигналів.

25. У чому суть «протигоночного» кодування станів автомата?

26. Як забезпечити перепад управляючого сигналу у випадку, коли операторну вершину з цим сигналом охоплює «петля»?

27. Як визначити час переходу автомата з одного стану в інший?

ЛАБОРАТОРНА РОБОТА №2

БЛОКИ МІКРОПРОГРАМНОГО УПРАВЛІННЯ

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

Теоретичні відомості [1, 2, 3, 4, 5, 6]

Додаткові теоретичні відомості

Синтез блоків мікропрограмного управління

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

Спрощена структурна схема БМУ наведена на рис. 3.3.

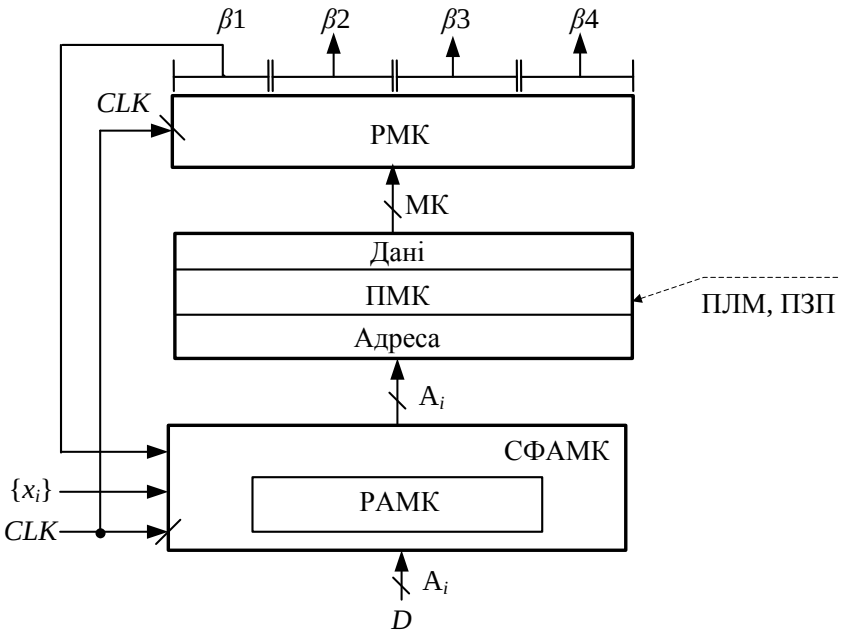


Рис. 3.3. Структурна схема БМУ

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. 3.4 наведений формат мікрокоманди.

МК	$\beta 1$	$\beta 2$	$\beta 3$	$\beta 4$

Рис. 3.4. 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. 3.10.

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. 3.11. 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).

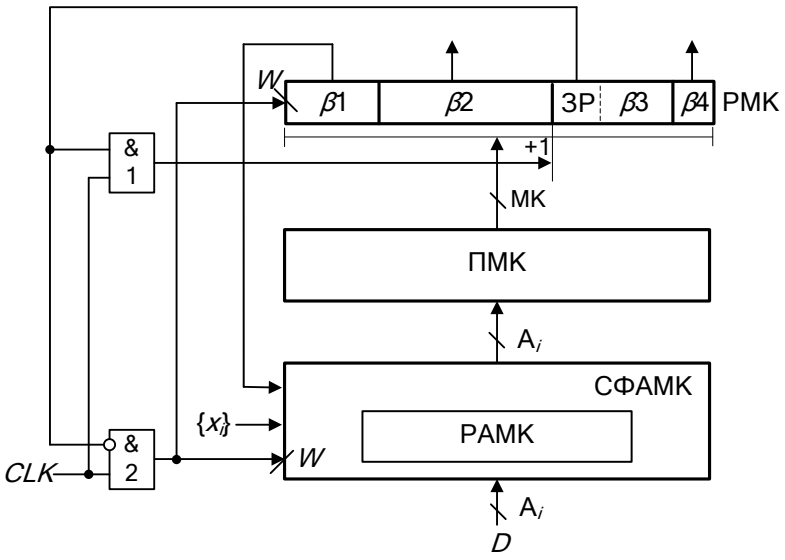


Рис. 3.10. Структурна схема БМУ з урахуванням зони затримки УС

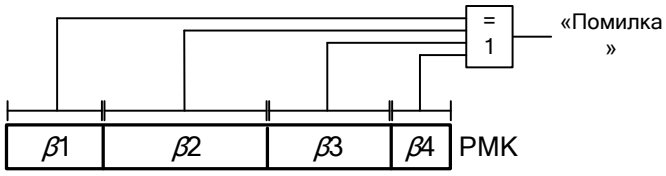
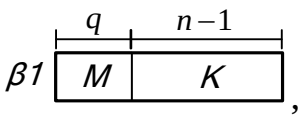


Рис. 3.11. Схема контролю слова МК на парність

Синтез 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:

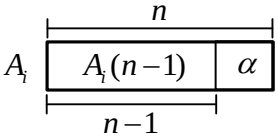
$$q =]\log_2(k+2)[, \qquad (3.4)$$

де k – кількість зовнішніх умов.

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

$$n_k = n-1$$

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. 3.14. На цій та подальших схемах БМУ входи для занесення початкової адреси *D* в РАМК умовно не показані.

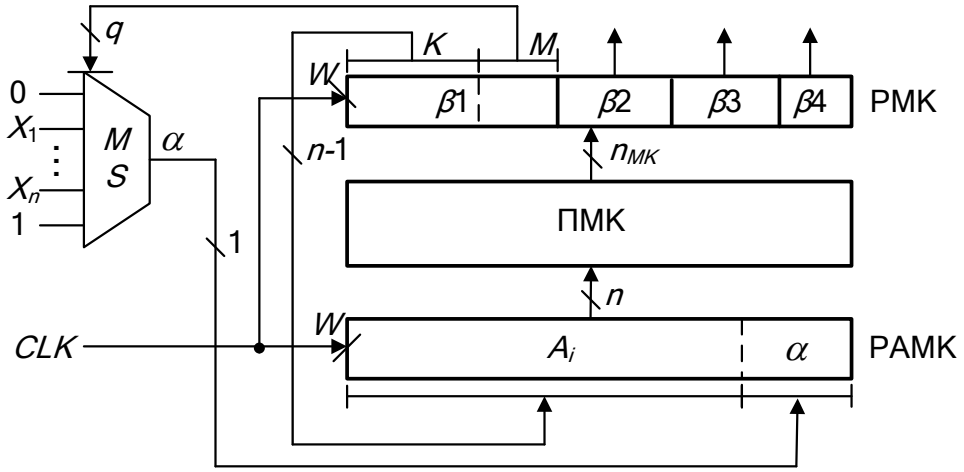


Рис. 3.14. Структурна схема БМУ з примусовою адресацією

БМУ з відносною адресацією

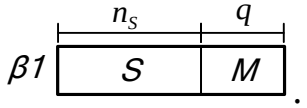
За відносної адресації адреса наступної МК визначається за формулою:

$$A_{i+1} = A_i + S + \alpha, \tag{3.5}$$

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:

$$n_S =]\log_2 N[+ 1, \tag{3.6}$$

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

Структурна схема БМУ наведена на рис. 3.23.

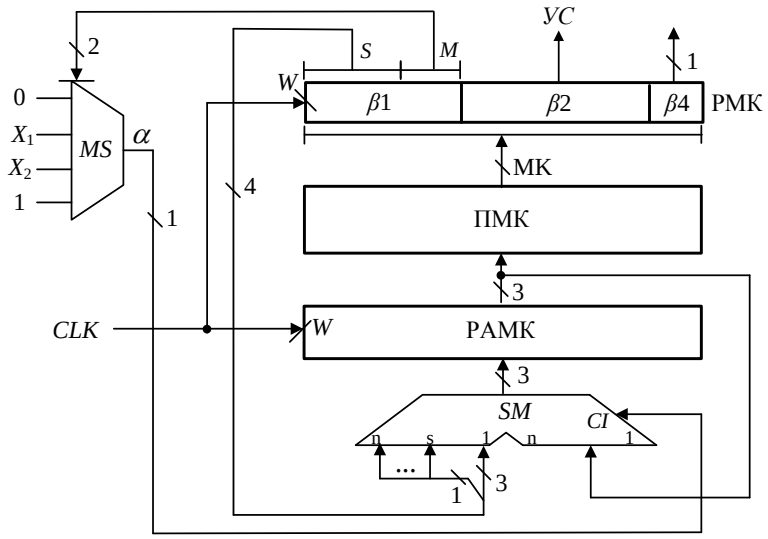


Рис. 3.23. Структурна схема БМУ з відносною адресацією

Приклад 3.8. Побудувати структурну схему БМУ і карту пам'яті мікропрограм для мікроалгоритму виконання операції множення. Мікроалгоритм повинен 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. 3.24.

Мікроалгоритм управління роботою пристрою наведений на рис. 3.25. Змістовний МА наведений на рис. 3.26.

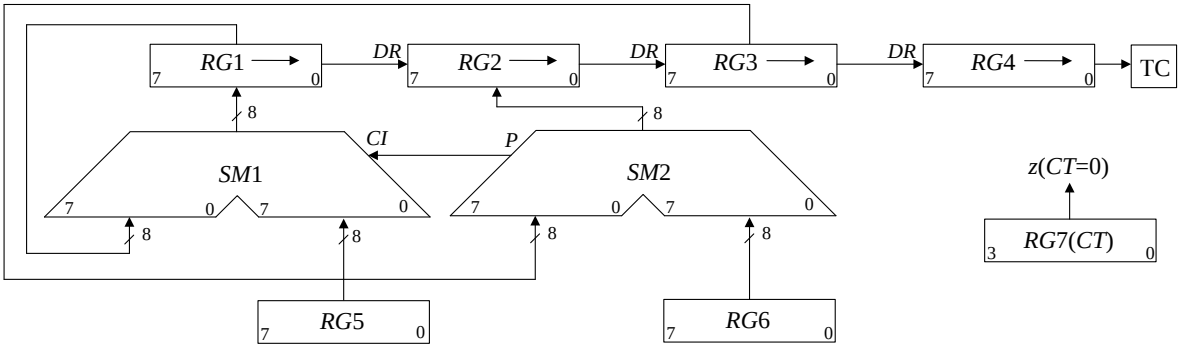


Рис. 3.24. Структурна схема пристрою множення

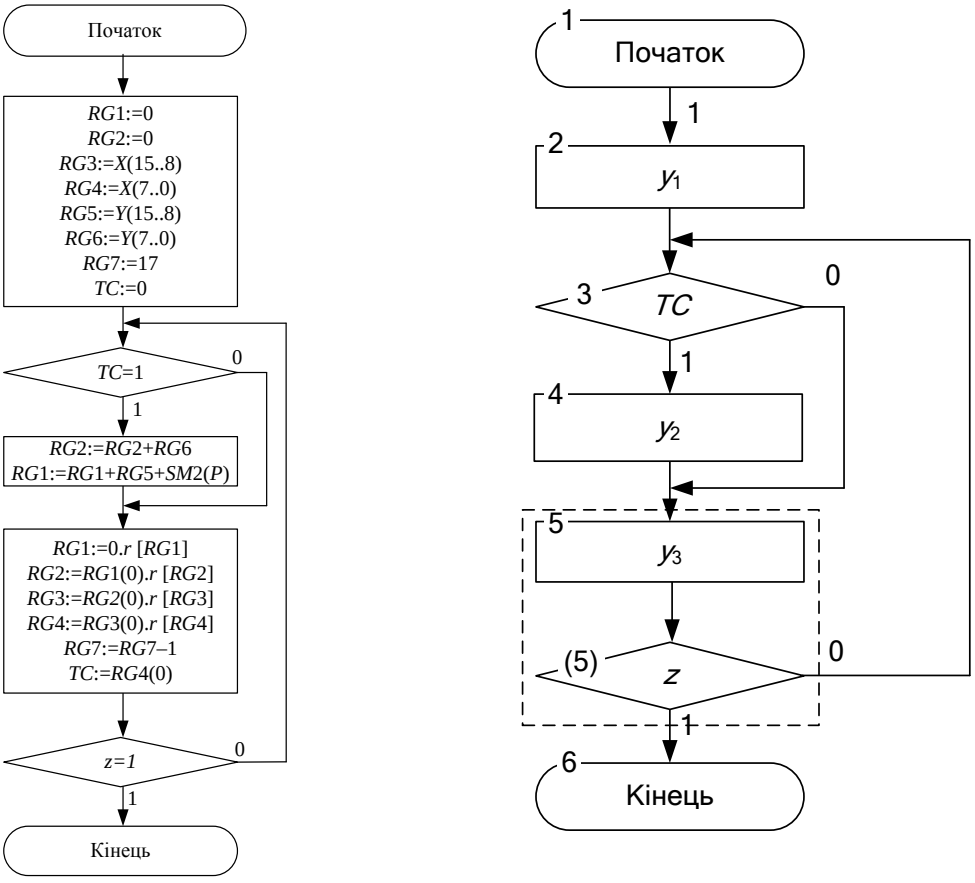


Рис. 3.25. Змістовний мікроалгоритм

Рис. 3.26. 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 3.11. 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 3.12. Кодування сигналів

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

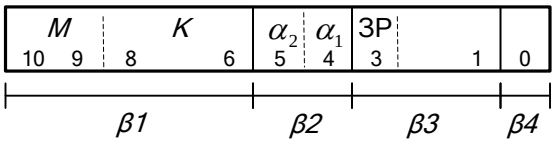
За виразом (3.3) розрахуємо довжину зони $\beta 3$:

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

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

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

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



Розміщуємо мікрокоманди в пам'яті мікрокоманд (рис. 3.27).

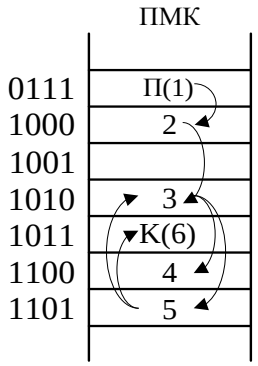


Рис. 3.27. Розміщення мікрокоманд в ПМК

Карта програмування БМУ наведена у табл. 3.13.

Таблиця 3.13. 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, розробленого для реалізації множення, зображена на рис. 3.17.

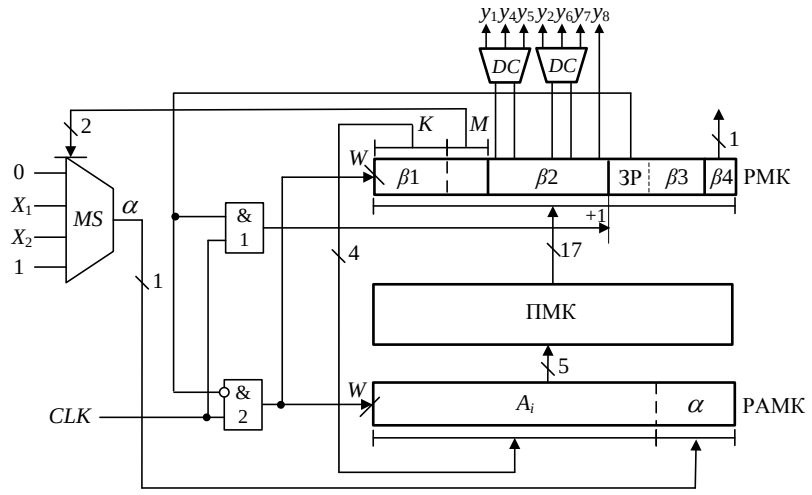


Рис. 3.17. Схема БМУ з примусовою адресацією

Підготовка до роботи

1. Побудувати структурну схему БМУ і карту пам'яті мікропрограм для мікроалгоритму виконання операції множення. Мікроалгоритм повинен забезпечувати управління арифметико-логічним пристроєм із розподіленою логікою відповідно до варіанту лабораторної роботи 1 (БМУ повинен замінити управляючий пристрій із жорсткою логікою).

2. Під час виконання завдання необхідно враховувати дані наведені у табл. 3.14 – 3.15.

Таблиця 3.14. Вихідні дані до проектування

a_4	a_2	Спосіб адресації мікрокоманд	Структура ПМК	Ємність ПМК (слів)	Використати зону β_4 для перевірки слова МК
0	0	примусовий	лінійна	64	на непарність
0	1	примусовий	матрична		на парність
1	0	відносна	лінійна		на непарність
1	1				на парність
Спосіб мікропрограмування – горизонтальний;					
Забезпечити занесення початкової адреси мікроалгоритму в регістр адреси мікрокоманд.					

Таблиця 3.15. Вихідні дані до проектування

a_6	a_5	a_4	Тривалість мікрооперації підсумовування	Початкова адреса мікропрограми
0	0	0	7	$18h$
0	0	1	4	$0ah$
0	1	0	3	$06h$
0	1	1	6	$0eh$
1	0	0	11	$13h$
1	0	1	4	$07h$
1	1	0	5	$11h$
1	1	1	2	$0bh$

Виконання роботи

1. Використовуючи моделюючу систему ПРОГМОЛС 2.0 побудувати і налагодити БМУ. Опис програмного комплексу ПРОГМОЛС 2.0 наведений у додатку М.

2. На спроектованому пристрої виконати числовий приклад із заданими викладачем значеннями операндів.

3. У протоколі навести функціональну схему пристрою для виконання операції множення.

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. 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 $\beta 4$.

ЛАБОРАТОРНА РОБОТА №3
ПОБУДОВА БЛОКІВ ОБРОБКИ ДАНИХ НА ЕОМ з МІКРОПРОГРАМНИМ УПРАВЛІННЯМ

Мета роботи: Вивчення схемотехнічних особливостей, системи мікроопе-рацій ЕОМ з мікропрограмним управлянням і принципів побудови блоків обробки даних

Теоретичні відомості: 1.2., 1.4

Підготовка до виконання практичної роботи

1. Вивчити структуру ЕОМ з мікропрограмним управлінням і спо-соби побудови операційних блоків.

2. Записати номер варіанту (номер залікової книжки) у двійковому поданні і виділити шість молодших розрядів $h_6 - h_1$. За отриманими значен-нями двійкових розрядів вибрати варіанти завдань.

3. Розробити операційну схему, Ф- і ФС-микроалгоритмы обчис-лення заданої функції для 16-розрядних двійкових чисел зі знаком (старший розряд – знаковий) відповідно до табл. 6.1.

4. Побудувати структурні схеми шістнадцятирозрядних операцій-них блоків. Для кожного операційного блоку визначити за допомогою часо-вих діаграм мінімальну тривалість такту.

5. Розробити мікроалгоритм обчислення функції, що задана у табл. 6.1, забезпечивши формування результату у регістрі вказаному у табл. 6.2. Для подання вихідних даних застосувати доповнювальний код. Під час роз-робки мікроалгоритму передбачити формування на довільному виході опе-раційного блоку ознаки, що вказує на наявність хоча би в одному розряді результату одиниці.

6. Скласти мікропрограму у кодах мікрокоманд.

7. Виконати числовий приклад застосувавши дані з табл. 6.3. Скласти цифрову діаграму стану регістрів.

8. Вивчити загальні положення до виконання практичних робіт із застосуванням лабораторного комплексу *COMPLEX*.

Таблиця 6.1. Варіанти завдань

h_2	h_1	Функція
0	0	$F = 16(X1 + X2 - 1) \& (X3 - X4)/2$
0	0	$F = 8(X1 \vee X2) + (X3 - 1 - X4)/16$
1	1	$F = 16[(X1 - 1) \& X2) - (X3 + X4)/4$
1	1	$F = 8(X1 - X2) + (X3 \oplus X4 - 1)/16$

Таблиця 6.2. Варіанти завдань

h_2	h_4	Регістр
0	0	*RQ*
0	0	*R*4
1	1	*R*5
1	1	*R*6

Таблиця 6.3. Варіанти завдань

h_2	h_1	*X*1	*X*2	*X*3	*X*4
0	0	–17	12	17	3
0	0	12	2	–	15
1	1	18	–	23	11
1	1	–9	10	31	–21

Порядок виконання роботи

1. Налагодити розроблену мікропрограму за допомогою лабораторного комплексу *COMPLEX*. Перевірити правильність обчислення заданої функції на числових прикладах.

2. Зробити висновки.

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; схему алгоритму, налагоджену мікропрограму у кодах мікропрограм, висновки за роботою.

Контрольні питання

1. Які мікрооперації в розглянутій системі можна суміщувати, а які ні?
2. Приведіть структуру і функціональне призначення БОД.
3. Приведіть структуру мікрокоманди для БОД і функціональне призначення керуючих сигналів для БОД.
4. Охарактеризуйте основні способи додавання та віднімання чисел в ЕОМ. Опишіть загальний склад устаткування, необхідний для реалізації операцій додавання та віднімання в ЕОМ.
5. Що таке арифметичний та логічний зсув? Як забезпечити арифметичний та логічний зсув слів подвоєної довжини?
6. Наведіть структуру і функціональне призначення СУСЗ.
7. Які мікрооперації реалізуються в АЛБ. Як задати в мікропрограмі початкові значення в регістрах АЛБ.

8. Приведіть структуру МК для АЛБ і призначення управляючих сигналів.

9. Наведіть структуру і функціональне призначення СУСЗ.

10. Які мікрооперації виконуються над ознаками результату в СУСЗ? Які типи зсувів забезпечує СУСЗ?

11. Приведіть структуру МК для СУСЗ і призначення управляючих сигналів.

ЛАБОРАТОРНА РОБОТА №4
РОЗРОБКА МІКРОПРОГРАМ ПЕРЕТВОРЕННЯ ДАНИХ В ЕОМ

Мета роботи. Вивчити архітектуру ЕОМ, що містить арифметико-логічний пристрій із двоадресним СОЗУ й блок мікропрограмного управління, одержати навички розробки мікропрограм

Теоретичні відомості []

Додаткові теоретичні відомості

Приклад 2.5. Розробити мікроалгоритм управління АЛП із зосередженою логікою для виконання операції множення за третім способом.

Виконання завдання

Для виконання задачі обираємо АЛП з односпрямованою магістраллю та двоадресним НОЗП, який дозволяє зменшити кількість мікрооперацій для перетворення даних.

Операційна схема множення за третім способом зображена на рис. 2.18. Операція множення виконуються із старших розрядів множника, за рахунок зсуву його вліво, сума часткових добутків також зсувається вліво, множене – нерухоме. Множник зберігається у регістрі *R*1, множене – у регістрі *R*3. У регістрі *R*2 накопичується сума часткових добутків. Регістр *R*4 застосовується як лічильник циклів. Розряд *C* регістру стану застосовується для збереження ознак при зсуві регістрів *R*1 та *R*2. Формування чергової суми часткових добутків у регістрі *R*2 відбувається із поширенням переносу у регістр *R*1. При цьому на суматорі *SM*1 підсумовується вміст регістру *R*1 з нулями, на вхід суматора *CI* подається розряд *C* регістру стану, де збережений вихідний перенос суматора *SM*2.

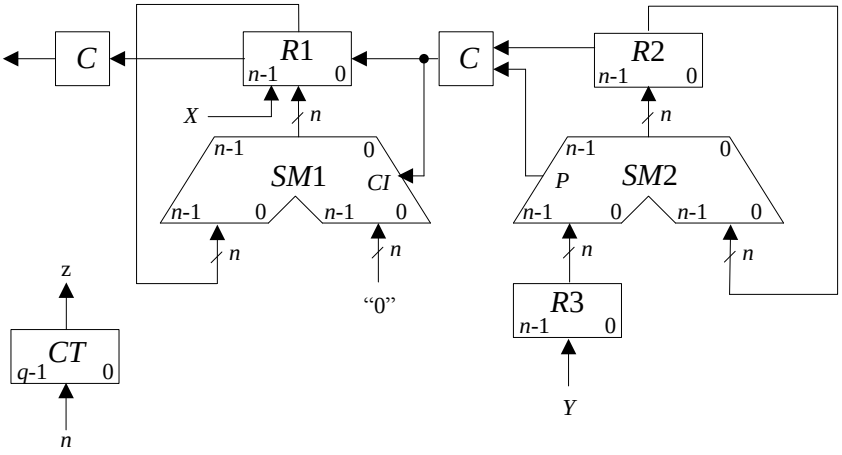


Рис. 2.18. Операційна схема операції множення

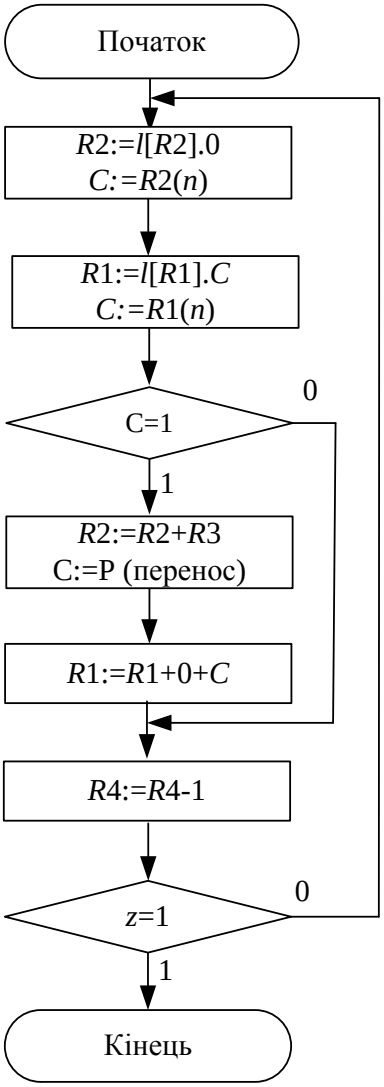


Рис. 2.20. Управляючий мікроалгоритм виконання операції множення

Мікропрограма множення цілих 16-розрядних чисел за схемою на **Ошибка! Источник ссылки не найден.**.1 може мати такий вигляд.

```
 link   l[3]: ct                    Завдання зв'язку між
                                    L3 й CT
accept  r12:0                       \ Занесення вихідних
accept  R15  :7fffh                 \ даних
accept  r10:7fffh                   \ в регістри
accept  r11:17                      \ і лічильник циклів
        {cont;load rm, z;}          \ Обнулення RM
label1  {cjp  not rm_c, label2;}    \ Аналіз цифри множ-
                                    ника
        {add r12, r12, R15, z;}     \ Додавання множеного
                                    до R12
label2  {or   srl, r12, r12, z;}    \ Зсув в R12 й R10
                                    суми
        {or   sr.9, r10, r10,       \ часткових добутків
        z;}
        {sub r11,r11,z,z;           \ Декримент і переві-
        load rm, flags;cem_c;}      рка
        {cjp  not rm_z, label1;}    \ на нуль лічильника
                                    (R11)
        {}                          \ на нуль лічильника
                                    (R11)
```

Підготовка до лабораторної роботи

1. Записати номер варіанту (номер залікової книжки) у двійковому поданні і виділити шість молодших розрядів $h_6 - h_1$. За отриманими значеннями двійкових розрядів вибрати варіанти завдань.

2. Для 16-розрядного процесора розробити операційну схему, Ф- і ФС-мікроалгоритмы заданої операції для 16-розрядних двійкових чисел зі знаком (старший розряд – знаковий) відповідно до табл. 3.1**Ошибка! Источник ссылки не найден.** і табл. 3.2. Дані з табл. 3.2 визначають форму подання даних (прямій або доповнювальний код).

3. Розробити із використанням мікроасемблера мікропрограму реалізації отриманого мікроалгоритму. Для реалізації мікроалгоритму використати регістри R10-R15.

4. Виконати числовий приклад, використовуючи дані табл. 3.3. Виписати контрольні значення проміжних результатів, які будуть використані при налагодженні мікропрограми.

Таблиця В1-2.5 – *Вибір арифметичної операції*

*h*4	*h*3	*h*2	*h*1	Множення (*D* = *A* x *B*)
0	0	0	0	1-м способом
0	0	0	1	2-м способом
0	0	1	0	3-м способом
0	0	1	1	4-м способом
*h*4	*h*3	*h*2	*h*1	Ділення (*D* = *A* / *B*)
0	1	0	0	1-м способом
0	1	0	1	2-м способом
*h*4	*h*3	*h*2	*h*1	Обчислення функції
0	1	1	0	$D=2C-4AB$
0	1	1	1	$D=A(B-1)+0{,}5C$
1	0	0	0	$D=2A(B+1)+0{,}5C$
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 3.2. Варіанти завдань

h_1	h_2	h_3	Форма подання			
			A	B	C	D
0	0	0	ДК	ДК	ПК	ДК

0	0	1	ПК	ПК	ДК	ДК
0	1	0	ПК	ДК	ПК	ДК
0	1	1	ДК	ПК	ПК	ДК
1	0	0	ПК	ПК	ДК	ДК
1	0	1	ДК	ДК	ПК	ДК
1	1	0	ДК	ПК	ДК	ДК
1	1	1	ПК	ДК	ДК	ДК

Табл. 1.3. Варіанти завдань

h_6	h_1	Значення		
		A	B	C
0	0	7	–5	12
0	1	–7	5	–9
1	0	–7	–5	24
1	1	7	5	–17

Порядок виконання роботи

Налагодити розроблену мікропрограму з використанням програмного емулятора.

Зробити виводи по роботі.

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; схему алгоритму, налагоджену мікропрограму у кодах мікропрограм, висновки за роботою.

Контрольні питання

1. Розробіть мікропрограму обчислення заданого арифметичного вираження.
2. Охарактеризуйте основні способи множення чисел.
3. Поясніть, як забезпечити арифметичний й логічний зсув слів подвоєної довжини?
4. Поясніть, яким чином можна управляти записом інформації в RM?
5. Поясніть призначення директив мікроасемблера.

6. Поясніть, що таке мікроалгоритм, мікропрограма, мікрооперація й мікрокоманда?

7. Які мікрооперації в розглянутій системі можна сполучати, а які не можна?

ЛАБОРАТОРНА РОБОТА №5
ВЗАЄМОДІЯ ПРОЦЕСОРА З ОСНОВНОЮ ПАМ'ЯТТЮ. СПОСОБИ АДРЕСАЦІЇ ОПЕРАНДОВ БЕЗ ВИКОРИСТАННЯ РОН

Мета роботи: Изучить основные принципы взаимодействия процессора с общей памятью в системах с объединенными и разделенными шинами адреса и данных. Изучить алгоритмы основных циклов обращения процессора к ОП. Изучить способы адресации операндов без использования РОН.

Теоретичні відомості []

Додаткові теоретичні відомості

Структурная организация ВС

Для изучения основных принципов взаимодействия процессора (П) и основной памяти (ОП) рассмотрим фрагмент вычислительной системы (рис. 1.1), основні характеристики якої наведені далі.

Системна шина (СШ) - разділена шина адреси (ША) и шина даних (ШД);

Основна пам'ять (ОП) об’ємом $2\text{М} = 2 \times 2^{20} = 2^{21}$. Минимальная информационная единица составляет 16-розрядное слово, то есть доступ к отдельному байту невозможен. Другими словами в системе реализована *выборка данных словами*.

При использовании нумерации байтов справа-налево адрес слова в памяти соответствует адресу младшего байта (рис.1.2). Формирование адреса (А) следующего слова выполняется по формуле $A_{i+1} = A_i + 2$, где значение 2 соответствует ширине выборки данных – 2 байта.

Таким образом необходимо адресовать 1М 16-розрядных слов или $2М : 2 = 2^{21} : 2 = 2^{20}$ слов. Для передачи 2^{20} слов потребуется 20-разрядная ША.

ОП виконує дві мікрооперації: запис слова за заданою адресою (ініціюється сигналом $\overline{W}$ (*Write*)) та читання слова (ініціюється сигналом $\overline{R}$ (*Read*)). Мнемонічно позначатимемо ці операції як {w;} та {r;} відповідно.

Разрядность шины данных 16 разрядов.

Разрядность шины адреса 20 разрядов.

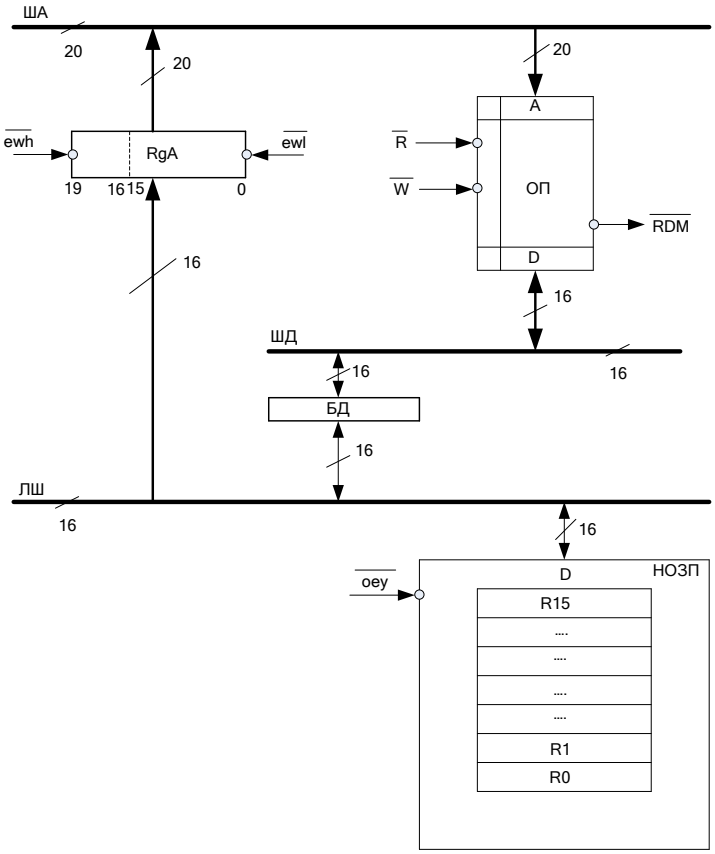


Рис. 1.1. Структурна організація обчислювальної системи

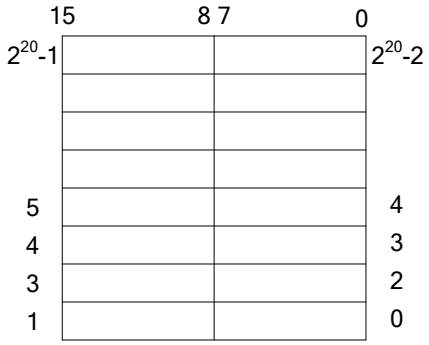


Рис. 1.2. Структурна основної пам’яті

Регистр адреси

Адреса ОП формується з використанням ЛШ. Оскільки ЛШ 16-розрядна, а ША 20-розрядна, то для формування адреси ОП необхідно два такти.

Інформація з ЛШ в регістр адреси (*RgA*) записується за два прийоми. Для цього застосовуються відповідні сигнали дозволу запису: $\overline{EWH}$ (*Enable Write High*) – дозвіл запису в старшу частину *RgA*, та $\overline{EWL}$ (*Enable Write Low*) – дозвіл запису в молодшу частину.

Мнемонічно ці сигнали позначаються наступним чином: {ewh;} та {ewl;}.

Поділ *RgA* на дві частини здійснюється за допомогою директиви LINK EWH.

Приклад:

```
link ewh : 16      \ молодша частина RgA включає розряди 15…0,
                   \ старша частина RgA – розряди 19…16
```

Спочатку на ЛШ формують, наприклад, молодші розряди адреси пам‘яті, та за допомогою сигналу {ewl;} записують в молодшу частину *RgA*, а потім на ЛШ формують старші розряди адреси й за допомогою сигналу {ewh;} записують в старшу частину *RgA*.

Як окремий випадок можна передавати 16-розрядну адресу, яка формується в одному з регістрів НОЗП. В цьому випадку в регістр *RgA* записується тільки молодша частина адреси за допомогою сигналу {ewl;}, при цьому за замовчуванням старша частина адреси приймається такою, що дорівнює нулю. Але в цьому випадку неможливо адресувати частину адресного простору, що адресується більш ніж шістнадцятьма розрядами (рис. 1.3).

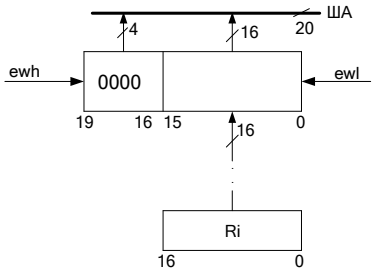


Рис. 1.3. Поділ регістру адреси на старшу і молодшу частини.

Буфер данных застосовується для передавання слів даних між НОЗП і ОП.

16-разрядное НОЗП входить у склад П і містить шістнадцять 16-розрядних регістрів.

Директиви мікроасемблера для роботи з пам'яттю

Крім директиви LINK EWH для роботи з ОП передбачено ще дві директиви мікроасемблера.

DW – директива завдання значень в комірках ОП. Ця директива (*Define Word*) дозволяє задати значення в одній або кількох суміжних комірках ОП одночасно.

Загальний вигляд директиви:

```
DW <адреса>:<значення>
```

Приклад:

```
dw 0A000h:36      \ за адресою A000 задати слово 36
dw 20h :2,4,8      \ за адресами 20h, 22h, 24h записати
                   \дані 2, 4, 8
```

ACCEPT RDM_DELAY – директива завдання швидкодії ОП.

Загальний вигляд директиви:

```
ACCEPT RDM_DELAY:<кількість тактів>
```

Приклад:

```
accept          \ швидкодія ОП дорівнює 3 тактам
rdm_delay:3     \ роботи БОД
```

Після завершення циклу роботи (запис або читання слова) ОП формує сигнал «Готовність пам'яті» $\overline{RDM}$ (*ReaDy Memory*), нульовий рівень якого свідчить про готовність ОП до наступного циклу. Цей сигнал в умовних мікрокомандах ФАМ мнемонічно позначають rdm.

Приклад: Встановити в нуль 4 старші розряди *RgA*, а в молодші записати адресу з *R*14. За даною адресою зчитати слово з ОП в регістр НОЗП *R*10. Виконати його декримент вмісту регістру *R*10 і результат записати у наступну комірку ОП. Швидкодія ОП вдвічі нижча за швидкодію процесора.

Виконання завдання:

```
\ область налагодження зв'язків
    link l2:rdm              \ сигнал RDM подається на
                             \ другий вхід МУ
    link ewh:16              \ поділ RgA
    accept rdm_delay:2       \ сигнал RDM формується
                             \ через 2 такти (затримка
                             \ пам'яті)
\ область завантаження даних
```

```
    dw 0024h:0aaaah          \ в ОП за адресою 24h
                             \ завантажується слово АААА
\ обрасть завантаження регістрів
    accept r14:24h           \ вихідна адреса записується в
                             \ R14
\ область програми
    {xor nil,r3,r3;oey;ewh;} \ RgA[19…16] := 0000,
                             \ завантаження нулів в старшу
                             \ частину регістру адреси
    {or nil,r14,z;oey;ewl;}  \ RgA[15…0]:=R14,
                             \ завантаження адреси із
                             \ регістру R14 в молодшу
                             \ частину регістру адреси
ss1 {r;cjp rdm,ss1;or r10,bus_d,z;}
                             \ R10:=ОП <24h>, зчитування
                             \ даних із ОП за адресою 24h
    {sub r10,r10,z,z;}       \ R10:=R10 – 1
    {add r14,r14,nz,nz;}     \ R14:=R14 + 2, формування
                             \ адреси наступної комірки
                             \ пам’яті
    {or nil,r14,z;oey;ewl;}  \ RgA[15…0]:=R14

ss2 {w;cjp rdm,ss2;or nil,r10,z;oey;}
                             \ ОП <26h> :=R10, запис слова
                             \ із регістру R10 НОЗП в ОП за
                             \ адресою 26h
    {}                       \ кінець
```

Способи адресації операндов без використання регістрів загального призначення

Виконавчою (фізичною) адресою називають послідовність нулів й одиниць, що видається на адресні шини. Такі послідовності можуть мати більшу довжину (32 і більше розрядів), тому не завжди вдається розмістити фізична адреса в слові команди, або одному з регістрів процесора. У таких випадках адресне поле в команді містить необхідну інформацію для обчислення виконавчої адреси.

Існує три способи адресації операндів без використання регістрів загального призначення (РЗП):

- Пряма (абсолютна) адресація операндів;
- Непряма адресація;
- Безпосередня адресація.

В багатоадресних командах спосіб адресації кожного операнда може бути окремим.

Розглянемо всі перераховані типи адресації на прикладі одноадресної команди.

Пряма адресація операндів

Узагальнена структура команди із прямою адресацією має такий вигляд:

КОП	E

де КОП – код операції, *E* – виконавча адреса (*E* – адреса операнда *x* – *Ax*).

За такого способу адресації команда містить адресу операнда в ОП. Схема вибірки операнда представлена на рис. 1.4, де РК – регістр команд; *AC* – акумулятор; *SM* – суматор. Для завантаження даних у процесор відбувається одне звертання до ОП.

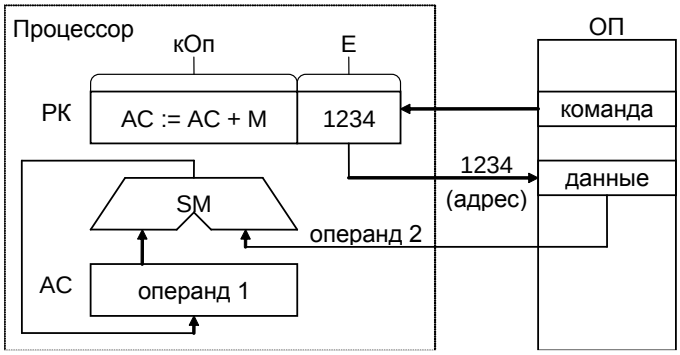


Рис. 1.4. Схема вибірки операнда за прямої адресації

Непряма адресація операндів

Узагальнена структура команди з непрямою адресацією має такий вигляд:

КОП	@A

де КОП – код операції, *@A* – адреса області ОП (покажчик адреси операнда, або адреса комірки пам'яті, де зберігається адреса операнда, тобто *A*(*Ax*)).

За такого способу адресації в команді зберігатися адреса області ОП, з якої можна одержати виконавчу адресу операнда, або інформацію, необхідну для його обчислення. Схема вибірки операнда представлена на рис. 1.5. Для вибірки даних відбувається два звертання до ОП.

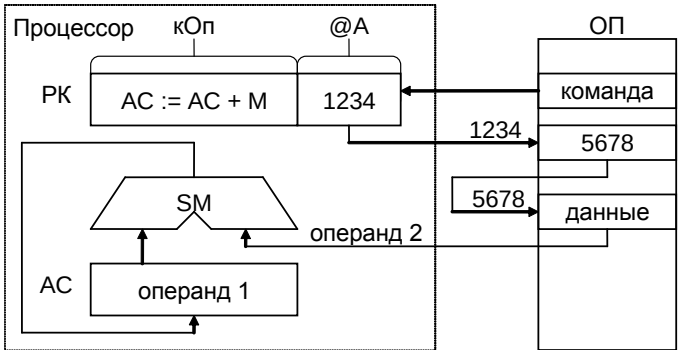


Рис 1.5. Схема вибірки операнда за непрямої адресації

Непряма адресація застосовується під час роботи з підпрограмами, записаними в ПЗУ.

Безпосередня адресація операндів

Узагальнена структура команди з безпосередньою адресацією має такий вигляд:

КОП	I

де *I* – операнд, використовуваний у команді.

За такого способу адресації необхідні дані (операнди) вказуються безпосередньо в самій команді. Дані вибираються з командою, тому додаткових звертань до ОП не потрібно, що показано на схемі на рис. 1.6. Однак цей спосіб адресації застосовується тільки для роботи з константами.

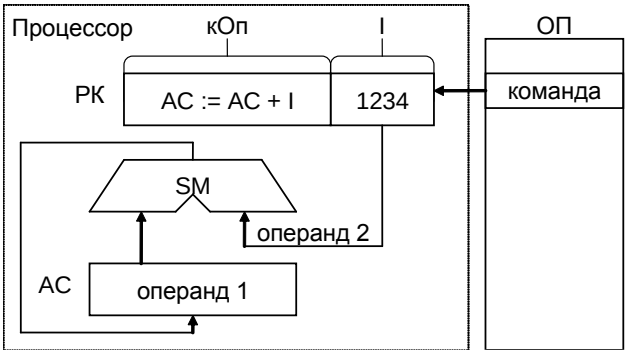


Рис 1.6. Схема вибірки операнда за безпосередньої адресації

Основні цикли звернення процесора до основної пам'яті

До основних циклів звернення до ОП належать цикли читання, запису, читання-модифікація запис.

Мікроалгоритми читання даних из ОП в регістри НОЗП

Формат команди читання:

MOV Ri,<A> / *ОП<Ax>* → *Ri* пряма адресація

/ *ОП<A(Ax)>* → *Ri* непряма адресація,

де *Ri* – номер регістру НОЗП, *A* – адреса комірки пам'яті, A_x – адреса комірки пам'яті де зберігається значення *x*.

Основні етапи циклу читання для прямої адресації даних зображені за допомогою мікроалгоритму на рис. 1.7.

1. A_x – адреса операнда *x*, що у вихідному стані знаходиться в одному з регістрів НОЗП, записується в регістр адреси, при цьому виконуються наступні дії:

$$R_i \rightarrow \text{ЛШ} \rightarrow RgA \rightarrow \text{ША} \rightarrow \text{А}_{\text{ОП}},$$

де $\text{А}_{\text{ОП}}$ – вхід адреси пам'яті.

{or nil,ri,z;oey;ewl;}

2. Процесор генерує сигнал *r*: R → ОП (читання даних за адресою, що виставлена на ША) і переходить у режим очікування.

3. Система знаходиться в режимі очікування готовності поки немає сигналу «Готовність». Через визначений час очікування, що дорівнює часу читання даних ($t_{\text{чит.}}$) дані видаються на ШД ($D_{\text{ОП}} \rightarrow$ ШД → БД → ЛШ) а пам'ять видає сигнал *RDM* – «готовність».

4. Якщо є сигнал «Готовність», дані записуються у заданий регістр НОЗП (БД → *Ri*).

```
ss1  {r;cjp rdm,ss1;or r10,bus_d,z;}
```

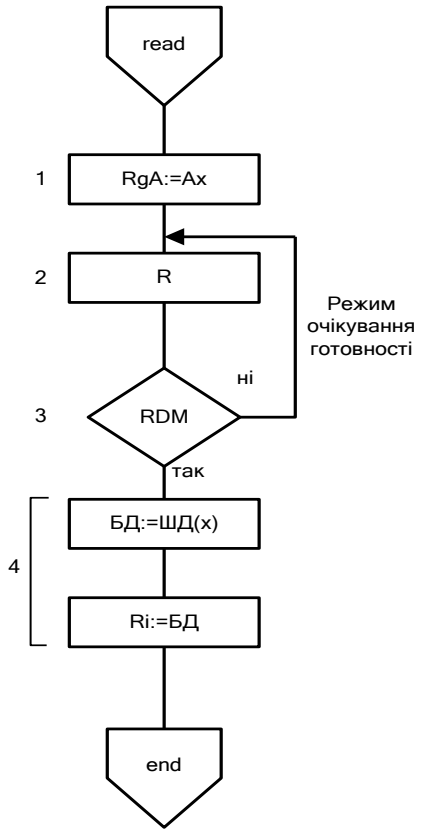
Основні етапи циклу читання для непрямої адресації даних зображені за допомогою мікроалгоритму на рис. 1.8.

Під час застосування непрямої адресації, на першому етапі в регістр адреси *RgA* записується адреса адреси операнда *A*(*Ax*) (рис. 1.8). На третьому етапі в БД зчитується адреса операнда, яка має бути знов записана у

регістр адреси *RgA,* що відбувається на етапі 5 відповідного мікроалгоритму.

```
ss1  {r;cjp rdm,ss1;or nil,bus_d,z;oey;ewl;}
```

Далі повторюються етапи 1 – 4, аналогічно застосуванню прямої адресації.



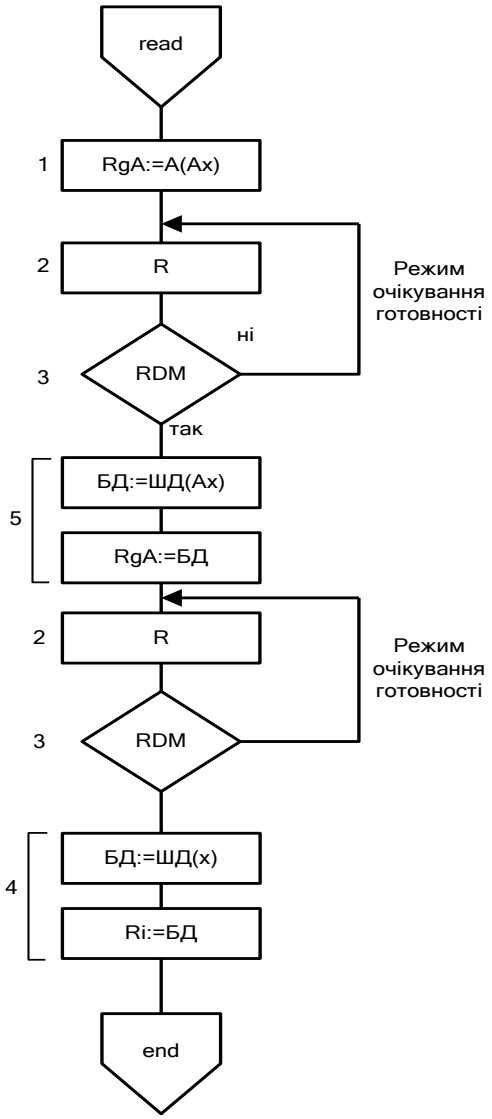


Рис. 1.7. Мікроалгоритм читання даних за прямої регістрової адресації

Рис. 1.8. Мікроалгоритм читання даних за непрямої регістрової адресації

Мікроалгоритми запису даних в ОП з регістрів НОЗП

Формат команди запису:

```
MOV <A>,Ri / Ri →ОП<Ax>  пряма адресація
           / Ri →ОП<A(Ax)> непряма адресація,
```

де *Ri* - номер регістру НОЗП, A – адреса комірки пам'яті, A_x – адреса комірки пам'яті де зберігається значення x.

Основні етапи циклу запису для прямої адресації даних зображені за допомогою мікроалгоритму на рис. 1.9.

1. A_x – адреса операнда x, що у вихідному стані знаходиться в одному з регістрів НОЗП, записується в регістр адреси, виконуються наступні дії:

$$R_i \rightarrow \text{ЛШ} \rightarrow RgA \rightarrow \text{ША} \rightarrow \text{А}_{\text{ОП}},$$

де $\text{А}_{\text{ОП}}$ – вхід адреси пам'яті.

```
{or nil,ri,z;oey;ewl;}.
```

2. Процесор виставляє дані, що мають бути записані на ШД:

$$R_i \rightarrow \text{БД} \rightarrow \text{ ШД} \rightarrow D_{\text{ОП}},$$

де D – вхід даних ОП.

```
ss2  {w;cjp rdm,ss2;or nil,ri,z;oey;}.
```

3. Процесор генерує сигнал запису w: $W \rightarrow$ ОП (запис даних за адресою, що виставлена на ША) і переходить у режим очікування.

Система знаходиться в режимі очікування готовності поки немає сигналу «Готовність».

Через визначений час очікування, що дорівнює часу запису даних ($t_{зап.}$) пам'ять видає сигнал *RDM* – «готовність».

4. Якщо є сигнал «Готовність» цикл запису закінчений.

Основні етапи циклу запису для прямої адресації даних зображені за допомогою мікроалгоритму на рис. 1.10.

У випадку застосування непрямої адресації цикл запису даних в ОП можна поділити на два кроки. На першому кроці необхідно вибрати адресу операнда із ОП, для чого застосувати етапи 1 – 3 та 5 аналогічно до мікроалгоритму, зображеного на рис. 1.4. Далі, на другому кроці, виставляються дані на ШД та відбувається звичайний цикл запису, аналогічно етапам 2 – 4 мікроалгоритму на рис. 1.5.

Наступний фрагмент мікропрограми виконує запис вмісту регістру *Rj* в ОП за адресою, розташованою в регістрі *Ri*, за застосування непрямої регістрової адресації.

```
                {and nil, ri,z;oey; ewl;}
          ss1 {r;cjp rdm ss1;and nil,bus_d,z;oey;ewl;
ss2 {w; cjp rdm ss2;and nil,rj,z,oey;}
```

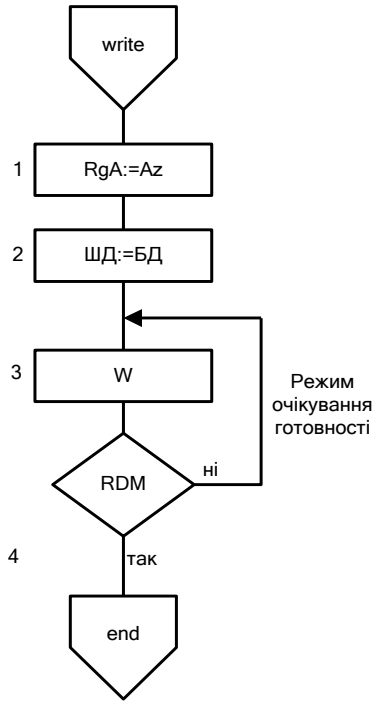


Рис. 1.9. Мікроалгоритм циклу запису за прямої регістрової адресації

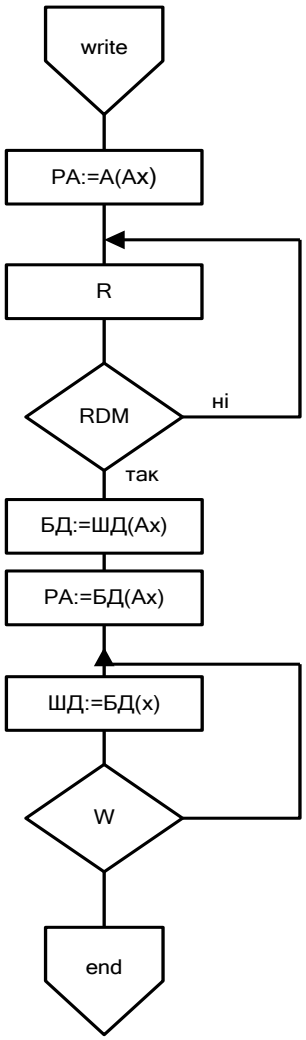


Рис. 1.10. Мікроалгоритм циклу запису за непрямої регістрової адресації

Приклад: Обчислити значення функції $z = x + y$. Дані розміщуються в ОП, для формування адреси наступної комірки пам'яті застосувати автоінкрементний спосіб. Адресу першого операнда задати у регістрі R8. Для операнда *x* застосувати пряму адресацію даних, для операнда *y* та результату – непряму.

Виконання завдання:

```
\ область налагодження зв'язків
    link l2:rdm                 \ сигнал RDM подається на
                                \ другий вхід МУ
    link ewh:16                 \ поділ RgA
    accept rdm_delay:2          \ сигнал RDM формується
                                \ через 2 такти (затримка
                                \ пам'яті)
\ область завантаження даних
    dw 00e0h:0011h              \ x
    dw 00e2h:0a24h              \ A(y)
    dw 00e4h:0c64h              \ A(z)
    dw 0a24h:0008h              \ y
    dw 0c64h:00                 \ z
\ область завантаження регістрів
    accept r8:00e0h             \ адреса x
\ область програми
\ виборка значення x в регістр R10 (пряма регістрова адреса-
ція)
    {add nil, r8, z; oey; ewl;}     / R8 → ЛШ → RgA
/ якщо немає rdm, повернення на мітку, інакше записуємо дані з
/ БД у регістр R10
ss1 {r;cjp rdm,ss1;add r10,bus_d,z;oey;}
/ формування адреси наступної комірки пам'яті
    {add r8,r8,002h;}
/ выборка y в регістр R11 (непряма регістрова адресация)
    {add nil,r8,z;oey;ewl;}     / A(Ay) → RgA
/ якщо встановився rdm, Ay → RgA
ss2 {r; cjp rdm,ss2;add nil,bus_d,z;oey;ewh;}
ss3 {r;cjp rdm,ss3;add r11,bus_d,z;oey;} / y → R11
    {add r10, r10, r11;}        \ підсумовування
/ формування адреси наступної комірки пам'яті
    {add r8,r8,002h;}
```

```
/ запис z в ОП (непряма регістрова адресация)
     {add nil,r8,z;oey;ewl;}        / A(Az) → RgA
/ якщо встановився rdm, Az → RgA
ss4  {r; cjp rdm,ss4;add nil,bus_d,z;oey;ewh;}
/ видача у БД вмісту регістру R10, якщо є сигнал rdm запис у ОП
ss5  {w; cjp rdm,ss5;add nil,r10,z;oey;}
     {}                             \ кінець
```

На рис. 1.11 представлена карта розміщення операндів у ОП.

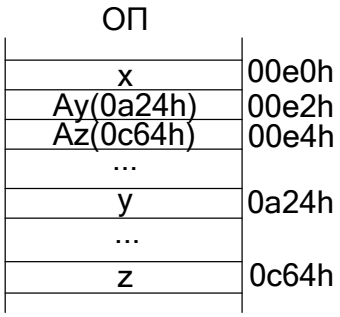


Рис. 1.11. Карта розміщення операндів у ОП

Підготовка до виконання завдання

1. Вивчити способи адресації операндів, в протоколі представити описи із застосування структурних схем способів адресації операндів без застосування РЗП.

2. Вивчити та представити у протоколі основні мікроалгоритми взаємодії П із ОП для різних способів адресації, навести часові діаграми виконання основних циклів взаємодії.

3. Для обчислення функції, заданої в табл. ХХ, розробити мікроалгоритми читання та запису операндів із ОП. Дані розміщуються у сусідніх комірках області даних основної пам'яті. Початкова адреса області даних у пам'яті $А_п$ задана у табл. 1.2. Спосіб формування наступної адреси комірки

пам’яті обрати із таблиці 1.1. Способи адресації операндів обрати з табл. 1.2: П – пряма адресація, НП – непряма адресація.

Під час виконання завдання номери робочих регістрів обрати самостійно. Адресу початкової комірки пам’яті $A_{\text{п}}$ розмістити в заданий регістр НОЗП (табл. 1.2). Для формування наступної адреси застосувати формули $\mathrm{A}_{i+1} = \mathrm{A}_i + 2$ та $\mathrm{A}_{i+1} = \mathrm{A}_i + 2$ для автоінкрементної і автодекриментної адресації відповідно.

Таблиця 1.1. Варіанти завдання

a_2	a_0	Функція
0	0	x + y = z
0	1	x & y = z
1	0	x V y = z
1	1	x – y = z

Таблиця 1.1. Варіанти завдання

a_0	Спосіб формування наступної адреси
0	автоінкрементний
1	автодекриментний

Таблиця 1.2. Варіанти завдання

a_4	a_2	a_0	X	Y	Z	Початкова адреса	Лічильник адреси
0	0	0	П	П	П	D0h	*R*7
0	0	1	П	П	НП	A0h	*R*10
0	1	0	П	НП	П	28h	*R*11
0	1	1	П	НП	НП	4Ah	*R*8
1	0	0	НП	П	П	20h	*R*9
1	0	1	НП	П	НП	F0h	*R*14
1	1	0	НП	НП	П	B2h	*R*13

1	1	1	НП	НП	НП	1Ah	*R*12

4. В протоколі навести структурну схему обчислювальної системи, карту розміщення даних у ОП, мікроалгоритми виконання операцій читання та запису даних.

Виконання лабораторної роботи

Налагодити розроблену мікропрограму із застосуванням моделюючої програми COMPEX. Під час налагодження виконати приклади із різними значеннями операндів.

ЛАБОРАТОРНА РОБОТА №6
ВИКОНАННЯ КОМАНД В ЕОМ

Мета роботи: Вивчити етапи виконання команд в ЕОМ. Навчитися розробляти мікроалгоритми вибірки, розпакування команд, виконання операцій і формування адреси наступної команди. Одержати навички розробки мікропрограм для ЕОМ з використанням мнемонічного мікроасемблера.

Теоретичні відомості []

Додаткові теоретичні відомості

Системи команд ЕОМ. Різновиди команд

Залежно від реалізованого в обчислювальній системі кількості команд, їх прийнято ділити на два типи:

- RISC - система зі скороченою системою команд;
- CISC - система з комплексною системою команд.

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

Всі команди по функціональній ознаці діляться на групи:

– основні команди - забезпечують перетворення інформації, тобто змінюють уміст регістрів. До них відносять: команди пересилання, зрушення, арифметичні й логічні команди;

– команди передачі керування - забезпечують безумовні й умовні переходи, виклик і повернення з підпрограм;

– команди вводу-виводу - забезпечують взаємодія процесора із зовнішніми пристроями. Для систем зі сполученим адресним простором ОП і ВУ команди даного типу можуть отсутствовать;

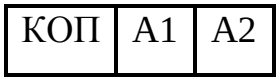
– системні команди - змінюють режими роботи процесора. Наприклад, дозволяють і забороняють переривання, блокую й розблокують операції вводу-виводу, установлюють і скидають режим покрокового виконання й т.п.

Кожна група може ділитися на підгрупи:

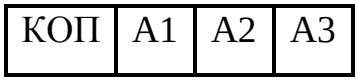
безадресна:

одноадресна:

двухадресная:



трехадресная:



де КОП – код операції; A1, A2, A3 – адреса або інформація, використовувана для обчислення адреси операнда.

Залежно від кількості адрес довжина команди може змінюватися. Якщо команда безадресна (без операндів), то це, як правило, системна команда. Одноадресні команди широко використаються як команди передачі керування, команди вводу-виводу. Двухадресні команди - команди основної групи. Трєхадресні команди дозволяють міняти приймач.

Адресація команд

Адреса команд формується на лічильнику команд (СК). Значення, що зберігається в СК, безпосередньо або після об'єднання із умістом іншого регістра СОЗУ (базою) визначає фізична адреса команди в ОП.

Виконання команд основної групи

Для команд основної групи можна виділити чотири основних етапи виконання:

- Вибірка команди з ОП;
- Розпакування (обчислення виконавчих адрес і вибірка операндов);
- Виконання заданої операції;
- Формування адреси наступної команди.

Зазначені етапи можуть виконуватися одночасно з використанням паралельно функціонуючих апаратур, тому такий розподіл на етапи є умовним.

Так, у системах з випереджальна вибірка команд, під час виконання однієї команди відбувається зчитування іншої команди з ОП. Лічена команда записується в чергу команд. Черга являє собою пам'ять із адресацій типу FIFO. Для роботи таких процесорів, крім лічильника команд, необхідно мати додаткові покажчики, за допомогою яких здійснюється доступ до буфера команд.

Розглянемо Ф-микроалгоритмы виконання кожного з етапів виконання команди. Уважаємо, що вибірка команди з пам'яті відбувається за один цикл звертання до системної магістралі, адреса команди зберігається в СК, ША й ШД розділені.

Вибірка команди

У якості СК може використатися будь-який регістр СОЗУ. Для вибірки команди необхідно виконати цикл читання даних з ОП, використовуючи як адреса вміст СК. Якщо в системі використається один з різновидів базової адресації (див. наступну лабораторну роботу), то виконавча адреса повинен бути обчислений на основі СК і вмісту базового регістра. МА виконання цього етапу команди наведений на Рис. 5.1. Результатом завершення даного етапу є команда в регістрі команди процесора.

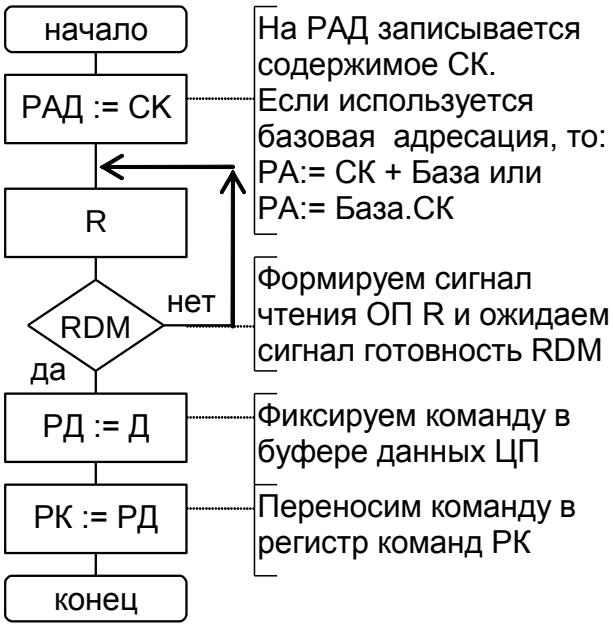


Рис. 5.1. МА вибірки команди з ОП.

Розпакування команд

У коді операції (кіп) можна виділити кілька функціональних полів:

– Формат команди (ФК) - дозволяє визначити довжину команди, кількість полів й їхнє положення в коді команди;

– Тип адресації (ТА) - задає способи адресації для кожного адресного поля А;

– Операція (Оп) - визначає виконувану операцію.

У деяких командах частина із цих полів може отсутствовать.

Мікропрограма розпакування аналізує вміст поля ФК, на основі чого визначає кількість полів типу адресації ТА й адреси А. Після цього здійснює вибірку операндов з ОП.

– Припустимо, що команда, завантажена в РК у попередньому пункті має наступний формат:

кіп (ФК.Оп)	ТА1	А1	ТА2	А2

де ТА1, ТА2 – поля задающие тип адресації відповідно для першого й другого адресного поля; А1, А2 – адресні поля.

Одна з гілочок МА розпакування даної команди наведена на Ошибка! Источник ссылки не найден..

– Передбачається, що для адресації першого операнда використається прямий тип адресації (задається полем ТА1), а для другого - непрямий (поле ТА2).

Як видно з мікроалгоритму, мікропрограми розпакування одне-, двох- і трехадресных команд будуть мати однакові ділянки. Кожна така ділянка здійснює вибірку операндов з ОП відповідно до одним зі способів адресації. Для зменшення обсягу мікропрограми розпакування, її повторювані участи можуть бути реалізовані у вигляді мікропідпрограм. Вони ж можуть виконувати аналіз полів типу адресації ТА.

Результатом завершення цього етапу є операнды, записані в робочі регістри процесора. У нашому прикладі це регістр РР1 і РР2, відповідно для першого й другого операнда.

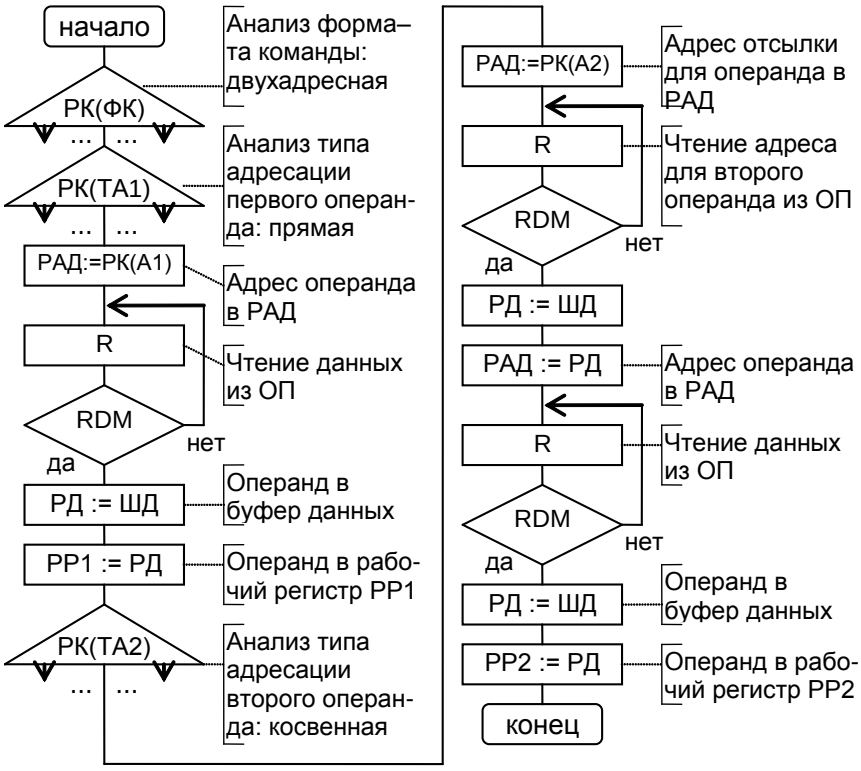


рис. 5.5. МА распаковки двухадресной команды.

Виконання заданої операції

На даному етапі необхідно проаналізувати вміст поля операції (Оп) у коді операції (кіп) і перейти до відповідної мікропідпрограми.

Для аналізу коду операції можуть використатися мікрооперації порівняння поля Оп з константами. У випадку формування в АЛУ результату “дорівнює”, БМУ звертається до відповідної мікропідпрограми. Недоліком такого способу є низька швидкодія, особливо для CISC систем. Іншим способом переходу до мікропідпрограми є обчислення її адреси на основі поля Оп. У такому випадку із РК виділяється поле операції Оп, що обробляється в АЛУ. Результатом цієї обробки є адреса мікропідпрограми операції, що

видається на ЛШ процесора. БМУ зчитує ця адреса через буфер М и переходить до необхідної мікропідпрограми. (см. Структуру системи в розділі “опис лабораторного комплексу”). Такий спосіб аналізу поля операції дозволяє скоротити час до декількох тактів синхронізації.

Приклад мікроалгоритму розпакування поля операції наведений на Ошибка! Источник ссылки не найден. а) і б). На Ошибка! Источник ссылки не найден.,а вважається, що аналіз поля операції ведеться його порівнянням з константами,а на

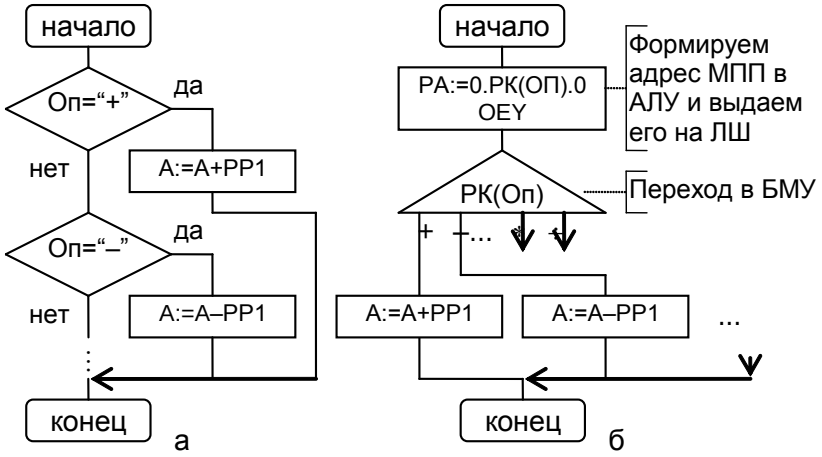


рис.5.6. МА выполнения команды.

Ошибка! Источник ссылки не найден.,б - адреса мікропідпрограми обчислюється в АЛУ.

Формування адреси наступної команди

У лічильнику команд перебуває адреса команди під час її виконання. Для визначення адреси наступної команди необхідно збільшити значення СК. Тому що довжина команд може бути різної (залежно від її формату й адресности), те СК инкрементируется на кількість байт, рівна довжині команди. Для визначення довжини команди може бути проаналізоване поле ФК, а також поля ТА.

МА обчислення адреси наступної команди представлена на Ошибка! Источник ссылки не найден., де l – довжина команди в байтах.

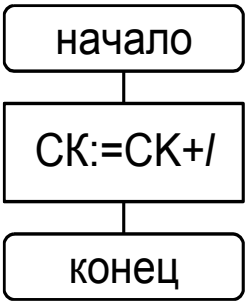


Рис. 5.7. МА вычисления адреса следующей команды.

Підготовка до лабораторної роботи

– Розробити Ф- і Фс-микроалгоритмы виконання всіх етапів одноадресної команди множення (вибірка, розпакування, виконання операції, формування адреси наступної команди). З використанням мікроасемблера написати мікропрограму, що реалізує Фс-микроалгоритм. Операція множення виконується за умовою лабораторної роботи №2.

– Вважати, що система команд містить команди двох форматів (одне- і двухадресные). Формат одноадресних команд показаний на Ошибка! Источник ссылки не найден.. Код операції множення прийняти рівним $a_5\ a_4\ a_2\ a_1$, де a_i – цифри номера залікової книжки, представленого у двійковій системі числення.

– Команда множення повинна забезпечити виконання наступного перетворення R14,R15 := R15 (M, тобто 32-розрядний результат повинен бути записаний у парі регістрів, в одному йз яких перебуває перший 16-розрядний операнд. Другий 16-розрядний операнд M вибирається з пам'яті.

Для мікроалгоритмів і мікропрограми докладно розробляються тільки ті галузі, які відповідають заданій команді й типу адресації. Тип адресації для розробки визначений у Ошибка! Источник ссылки не найден..

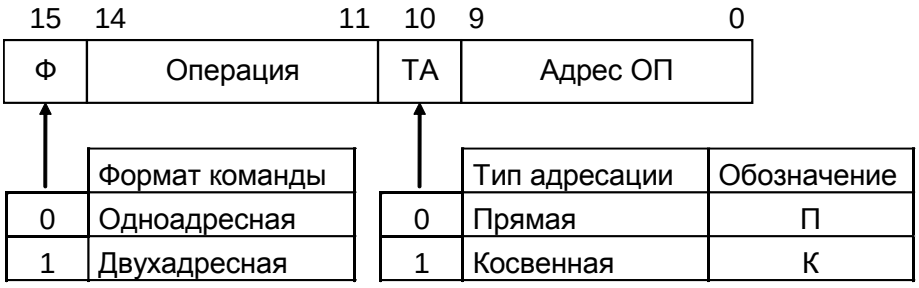


рис. 5.8. Обобщенный формат одноадресной команды

Табл. 5.2

a_3	Тип адресації
0	П (Пряма)
1	ДО (Непрямого)

Регістри R0...R7 є програмно доступними регістрами загального призначення (РОН), причому R7 є також лічильником команд, а регістр R6 зарезервований для подальшого використання. Регістр R15 використається як акумулятор, R8 й R9 як регістр команд. Інші регістри є робочими.

Порядок виконання роботи

– Налагодити розроблену мікропрограму з використанням програмного емулятора.

– Зробити виводи по роботі.

Контрольні питання

– Охарактеризуйте етапи виконання команд, приведіть мікроалгоритми їхньої реалізації.

– Охарактеризуйте основні способи адресації операндов з використанням і без використання Ронов.

– Як забезпечити правильне зчитування й запис даних на згадку?

– Яким образом можна управляти записом інформації в RA й RB, навіщо використаються зазначені регістри?

– Поясните призначення директив мікроасемблера, що визначають роботу з пам'яттю.

ЛАБОРАТОРНА РОБОТА № 7
ВИКОНАННЯ КОМАНД ВВОДУ-ВИВОДУ

Ціль роботи. Вивчити етапи виконання команд вводу-виводу. Навчитися розробляти мікроалгоритми й мікропрограми реалізації кожного етапу зазначених команд. Вивчити способи взаємодії процесора із зовнішніми пристроями в програмному режимі опитування готовності пристроїв. Одержати навички розробки мікропрограм з використанням мнемонічного мікроасемблера

Теоретичні відомості

Команди передачі управління

При природному (послідовному) порядку виконання команд, адреса наступної команди може бути обчислений у такий спосіб: СК := СК + *l*, де *l* – довжина команди.

Команди передачі керування дозволяють змінити природний порядок виконання програми. До цієї групи відносять наступні типи команд:

– безумовний перехід;

– умовний перехід.

Незалежно від того, до якому з перерахованих типів ставиться команда, можна виділити кілька загальних етапів її виконання, а саме:

– вибірка команди;

– розпакування команди;

– виконання команди.

Етапи вибірки й розпакування для команд передачі керування нічим не відрізняється від відповідних етапів для команд основної групи. Вибірка команди й завантаження операндов може здійснюватися кожним з розглянутих раніше способів.

На етапі виконання команди цієї групи модифікують або можуть модифікувати (команди умовного переходу) значення лічильника команд СК.

мал. 7.1. МА виконання команди безумовного переходу

Команди безумовного переходу

Для команд даного типу, отриманий у результаті вибірки й розпакування операнд, використається як адреса переходу. Мікроалгоритм виконання даної команди наведений на **Ошибка! Источник ссылки не найден.**, де БВК - блок вибірки команди - відповідає етапу вибірки, БРК - блок розпакування команди - етап розпакування, Е - ефективний (фізичний) адреса наступної команди.

Тому що БВК і БРК є однаковими для команд основної групи й передачі керування, то для формування ефективної адреси Е можуть використатися будь-які способи адресації, як з використанням Ронов, так і без їхнього використання.

Адреса наступної команди записується в СК. Це порозуміється тим, що стандартна мікропідпрограма вибірки команд із ОП використає вміст СК для обчислення адреси команди.

Команда умовного переходу

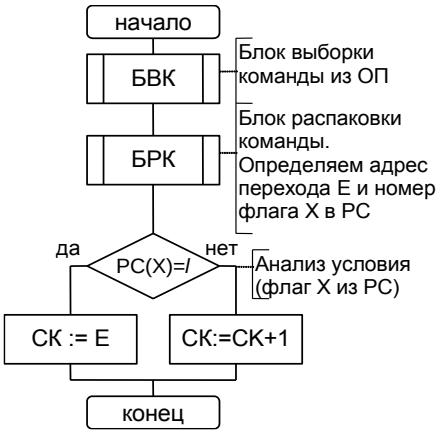
У відмінності від команди безумовного переходу, де в кожному разі СК модифікується обчисленою адресою Е, у командах умовного переходу, завантаження СК відбувається тільки при виконанні аналізованої умови. Як умова може виступати вміст одного або декількох битов у регістрі стану процесора. В останньому випадку говорять що перехід відбувається по масці. МА виконання команди умовного переходу наведений на **Ошибка! Источник ссылки не найден.**.

Якщо умова виконується, то в СК завантажується ефективна адреса Е. У противному випадку адреса наступної команди визначається також, як і для команд основної групи, тобто СК:=СК + l, де l – довжина команди.

Залежно від способу формування нового значення в СК, виділяють два типи команд переходу:

– абсолютні;

– відносні.



мал. 7.2. МА виконання команди умовного переходу

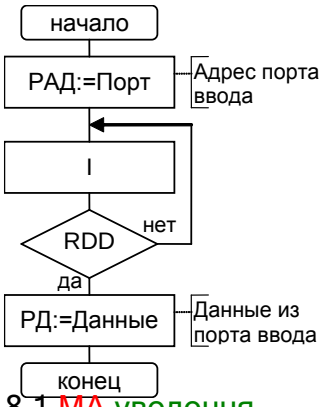
У першому випадку, ефективна адреса обчислюється на основі операндів, завантажених при розпакуванні команди. У другому - отриману адресу Е, може підсумуватися із умістом СК, тобто перехід виконується щодо поточної команди.

Команди вводу-виводу

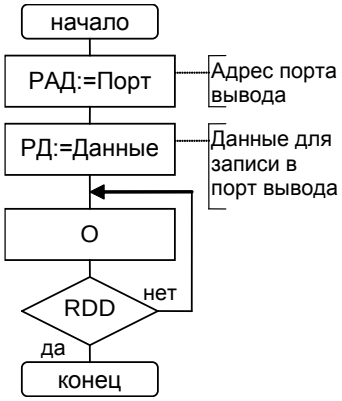
Якщо процесор використає роздільний адресний простір ОП і ВУ, то в його систему команд повинні входити команди роботи із зовнішніми пристроями - команди вводу-виводу. Найпростішими із зазначених команд є:

- команда уведення (IN), що дозволяє вважати дані з порту ВУ в процесор;
- команди виводу (OUT), призначена для запису даних із процесора в порт ВУ.

При обміні даними з портом вводу-виводу процесор формує сигнал уведення I (читання даних з порту) або виводу O (запис даних у порт). За змістом ці сигнали аналогічні сигналам читання R і запису W ОП. Типові МА уведення й виводу в порт ВУ наведені на **Ошибка! Источник ссылки не найден.**8.1 і **Ошибка! Источник ссылки не найден.** відповідно. Для визначення вірогідності даних, що вводять, і моменту фіксації вихідних даних у ВУ служить сигнал готовності RDD



мал. 8.1 МА уведення даних з порту уведення

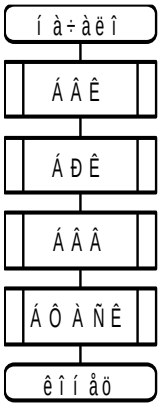


мал. 8.2 МА виводу даних у порт виводу

МА команди вводу-виводу містить чотири стандартних етапи виконання команди: вибірка, розпакування, виконання й обчислення адреси наступної команди. Узагальнений МА команди вводу-виводу представлений на Мал. , де БВВ – блок вводу-виводу, описуваний МА уведення (**Ошибка! Источник ссылки не найден.**8.1) для команди уведення або МА виводу (**Ошибка! Источник ссылки не найден.**8.2) для команди виводу. У блоці розпакування команди БРК обчислюється фізична адреса порту Е, з яким здійснюється обмін. У загальному випадку, для формування адреси можуть використатися будь-які способи адресації операндов, однак на практиці найбільше поширення одержали прямі й непряма регістрова адресації.

У ряді систем команди вводу-виводу ставляться до привілейованих команд, і доступні тільки в деяких режимах роботи процесора. У такому випадку, перед початком обміну даними з портом вводу-виводу, у мікропрограмі необхідно перевірити режим роботи процесора (прапор у регістрі стану процесора).

Взаємодія процесора із зовнішніми пристроями

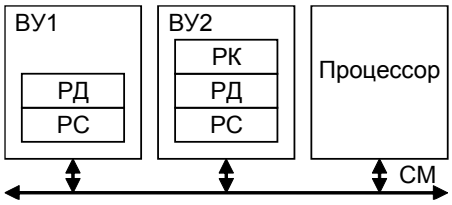


мал. 8.3. Узагальнений МА виконання команди вводу-виводу

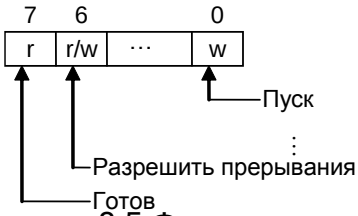
При обміні інформацією з ВУ процесор виступає як активний елемент, тобто управляє процесом обміну. Можна вказати три найбільше що часто використаються режиму обміну інформацією з ВУ:

– Режим програмного опитування готовності ВУ;
– Режим переривань програм;
– Режим прямого доступу до пам'яті (ПДП).

Будь-який зовнішній пристрій може бути представлене у вигляді сукупності декількох регістрів, відображуваних на порти вводу-виводу або адресний простір ОП (**Ошибка! Источник ссылки не найден.**8.4). Серед цих регістрів можна виділити регістр даних РД, регістр стану РС і регістр команд РК. Для керування простими пристроями досить два регістри: даних



мал. 8.4. Регістри ВУ



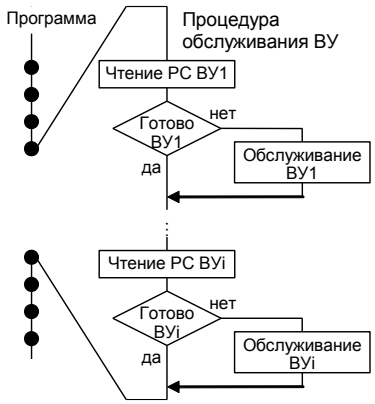
мал. 8.5 Формат регістра стану ВУ

і стану. Для складних пристроїв кількість регістрів може досягати декількох десятків.

РД служить для обміну інформацією між процесором і ВУ. РС дозволяє визначити стан зовнішнього пристрою й задати спосіб його взаємодії із системної (системною магістраллю СМ і процесором). Кожен розряд регістра стану може виконувати певну функцію. Можливий формат регістра стану наведений на **Ошибка! Источник ссылки не найден.**, де розряди позначені буквою r - доступні для читання, w - для запису, а r/w - для читання/запису. РК дозволяє настроїти ВУ на певний режим роботи або виконати задані дії.

Режим програмного опитування готовності ВУ (полинга)

У режимі програмного опитування процесор, що з ініціатором обміну є певною періодичністю, зчитує вміст РС. У прочитаних даних процесор аналізується біт готовності ВУ, і якщо він установлений, то робить обслуговування ВУ (обмін даними). Обмін даними здійснюється через РД.



мал. 8.6. Схема обслуговування ВУ в режимі опитування

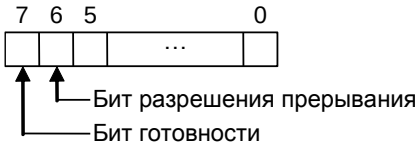
Для зчитування інформації із РС й обміну даними через РД використаються команда вводу-виводу. Схема обслуговування ВУ в режимі полинга наведена на **Ошибка! Источник ссылки не найден.**8.6.

Головним достоїнством даного режиму обслуговування ВУ є його простота. До недоліків можна віднести: непродуктивні витрати процесорного часу на опитування готовності ВУ, що знижує продуктивність системи, а також підвищену складність обслуговування аварійні ситуації в системі.

Підготовка до лабораторної роботи

– Доробити мікропрограму, отриману при виконанні лабораторних робіт 3, 4 й 5, включивши до складу команд одноадресні команди уведення й виводу. Формат команд зазначений на мал. Х. Код операції команди уведення $a_5\ a_4\ a_2\ a_1+5$ й операції виводу – $a_5\ a_4\ a_2\ a_1+6$.

Зовнішні пристрої містять регістр стану (РС) і регістр даних (РД). Звертання до РС можливо в будь-який момент часу. ДО РД можна звертатися тільки коли в РС установлений біт готовності зовнішнього пристрою. Регістр РС є 8-розрядним, а РД – 16-розрядним. Формат РС показаний на **Ошибка! Источник ссылки не найден.**8.7.



мал. 8.7. Формат регістра стану ВУ

Команда уведення здійснює уведення даних в акумулятор, функцію якого в системі виконує регістр R15.

За допомогою команди виводу здійснюється вивід слова з акумулятора в зовнішній пристрій.

Адреси регістрів зовнішніх пристроїв включаються в загальний адресний простір зовнішніх пристроїв. Адреса РД треба за адресою РС. При роботі з регістрами зовнішніх пристроїв використаються сигнали I й O, що формуються в блоці мікропрограмного керування.

– Розробити програму в кодах команд для передачі двох слів із пристрою уведення в пристрій виводу (адреси регістрів зазначені в Табл. 8.8.1). Перед звертанням до РД зовнішнього пристрою варто перевіряти готовність пристрою до обміну. Для цього необхідно прочитати РС пристрою й перевірити біт готовності.

Для занесення розробленої програми в основну пам'ять при налагодженні мікропрограми використається директива DW

Приклад 8.1

```
d 120h:7 \ записати за адресою 120h дані 7515h.
w 515h
```

Пристрою уведення й виводу настроюються директивою ACCEPT.

Приклад 8.2

```
link   l2:rdd   \ підключити до l2
                сигнал READY ВУ
link   l1:rdm   \ підключити до l1
                сигнал READY пам'яті
accept dev[2]:  \ пристрій виводу
       o,
       30h,     \ адреса РС
       32h,     \ адреса РД
       3,       \ затримка сигналу
                RDM у тактах
       14       \ затримка установки
                біта готовності РС
                 \ після звертання
                до РД у тактах
accept dev      20h, 22h, 3, 15
       [1]: i,
accept dev_buf  \ дані, які будуть
       [1]:123  уводитися в
       4h,       \ процесор із РД
```

```
5678h,
89abh,
0eeeeh
```

Порядок виконання роботи

- Налагодити розроблену мікропрограму з використанням програмного емулятора в режимі трасування.
- Поставити крапку останова наприкінці мікропрограми виконання команд. Виконати в автоматичному режимі записану в пам'яті програму обміну інформацією між зовнішніми пристроями.
- Зробити виводи по роботі.

Контрольні питання

- Охарактеризуйте етапи виконання команд уведення й виводу.
- Як забезпечити правильне зчитування даних із РД зовнішнього пристрою.
- Чи можуть адреси ВУ включатися в адресний простір основної пам'яті?
- Як ураховується при написанні мікропрограм затримка формування сигналу RDD?
- Поясните призначення директив мікроасемблера, що визначають роботу із зовнішніми пристроями.

Додаткова література /5, 6/

Табл. 8.1

a_3	a_2	a_1	**Адреси РС**	
			Увв	**УВыв**
0	0	0	02H	82H
0	0	1	12H	92H
0	1	0	22H	A2H
0	1	1	32H	B2H
1	0	0	42H	C2H
1	0	1	52H	D2H
1	1	0	62H	E2H

1	1	1	72H	F2H

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. *Каган Б.М.* Электронные вычислительные машины и системы. – М.: Энергоатомиздат, 1985. – 552 с.
4. *Карцев М.А.* Архитектура цифровых вычислительных машин.– М.: "Наука", 1978. – 295 с.
5. *Молчанов А.А., Корнейчук В.И., Тарасенко В.П.* Справочник по микропроцессорным устройствам. – К.: Техніка, 1987. – 288 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. *Пухальский Г.И., Новосельцева Т.Я.* Цифровые устройства: Учебное пособие для втуов. – Спб.: Политехника, 1996. – 885 с.
8. *Пухальский Г.И.* Проектирование микропроцессорных систем: Учебное пособие для ВУЗов. – Спб.: Политехника, 2001. – 544 с.
9. *Самофалов К.Г., Корнейчук В.И., Тарасенко В.П.* Цифровые ЭВМ. Теория и проектирование.– К.: Высш.шк. 1989. – 424 с.
10. *Тарабрин В.В., Лунин Л.Ф., Смирнов Ю.Н.* Интегральные микросхемы: Справочник. – М.: Радио и связь, 1990. – 528 с.
11. *Цифровые* ЭВМ. Практикум / К.Г.Самофалов, В.И. Корнейчук, В.П. Тарасенко, В.И.Жабин – К.: Высш.шк. 1989. – 124 с.