МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет менеджменту

Кафедра економічної кібернетики, комп'ютерних наук та інформаційних технологій

КОМП'ЮТЕРНА СХЕМОТЕХНІКА ТА АРХІТЕКТУРА КОМП'ЮТЕРІВ

Методичні рекомендації
для практичних занять та самостійної роботи
здобувачів першого (бакалаврського) рівня вищої освіти ОПП
«Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки»
денної форми здобуття вищої освіти

МИКОЛАЇВ
2025

УДК 004.2+621.38+681.3
К63

Друкується за рішенням науково-методичної комісії факультету менеджменту Миколаївського національного аграрного університету від 27 березня 2025 року, протокол № 7.

Укладачі:

О.О. Жебко – асистент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій, Миколаївський національний аграрний університет

Рецензенти:

І.О. Калініна – д.т.н., професор кафедри інтелектуальних інформаційних систем Чорноморського національного університету імені Петра Могили

В.В. Поживатенко – канд. фіз.-мат. наук, старший викладач кафедри вищої та прикладної математики

ЗМІСТ

Передмова

Курс дисципліни «Комп'ютерна схемотехніка та архітектура комп'ютерів» має важливе значення в теоретичній підготовці майбутніх фахівців і є обов’язковою компонентою підготовки здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки». Цей курс охоплює такі теми, як основні принципи побудови комп'ютерних систем, логіка роботи процесорів, організація пам'яті, інтерфейси взаємодії між компонентами системи тощо.

Мета дисципліни: сформувати у здобувачів необхідний обсяг теоретичних і практичних знань з основ теорії побудови та функціонування основних пристроїв, вузлів, базових елементів та архітектури сучасної комп'ютерної техніки.

Основними завданнями, що мають бути вирішенні у процесі викладення дисципліни, є надання здобувачам вищої освіти:

- знань про концепції та основні принципи роботи компонентів комп'ютера, таких як процесори, пам'ять, введення-виведення, та їх взаємодія;

- можливості ознайомитися з базовими елементами схемотехніки, такими як логічні вентилі, мультиплексори, дешифратори та інші;

- знань про внутрішню структуру та функціональні можливості процесорів, включаючи їхні операційні блоки, реєстри та механізми управління.

- навчання основним принципам організації пам'яті в комп'ютерних системах, включаючи кеш-пам'ять, ОЗП та ПЗП;

Предмет дисципліни: апаратні засоби комп'ютерів, їх структура, принципи роботи та взаємодія між окремими компонентами системи

Структура навчального курсу дозволяє здобувачам при подальшому навчанні або у професійній роботі отримати глибоке розуміння принципів роботи комп'ютерних систем на рівні апаратного забезпечення та набути практичних навичок для їх застосування.

Практична робота № 1

Знайомство з системою NI Multisim. Дослідження вольт-амперних характеристик компонентів електричних кіл

Дослідження вольт-амперних характеристик джерел напруги, струму, а також пасивних компонентів електричних кіл.

Основні теоретичні положення

Вольт-амперною характеристикою (ВАХ) електричного приладу називається залежність між струмом, що протікає через прилад і напругою, яка виділяється на його виводах. Для пасивних компонентів ВАХ визначається з закону Ома:

$$U = I \cdot R, \qquad (1)$$

і для постійного опору зображується у вигляді прямої лінії, крутизна якої в системі координат I, U залежить від величини опору R.

Для джерел постійної напруги ВАХ визначається рівнянням:

$$U = E - I \cdot r, \qquad (2)$$

де E – електрорушійна сила джерела напруги, тобто максимальне значення напруги джерела, яке визначається в режимі його холостого ходу (при розімкнутому колі навантаження, коли струм навантаження відсутній); r – внутрішній опір джерела живлення.

Для джерела постійного струму ВАХ будується в системі координат струм-напруга, тобто, залежність величини вихідного струму від напруги на навантаженні:

$$I = J - U \cdot g, \qquad (3)$$

де J – максимальне значення струму в режимі короткого замикання; g – внутрішня провідність джерела струму.

Формули (1)-(3) – справедливі і для електричних кіл змінного струму.

При використанні в електричних схемах резисторів, слід мати на увазі, що номінальний опір резистора справедливий тільки для температури 27°С. Якщо схема працює в іншому температурному середовище, використовується

коригуючий коефіцієнт ТКС, який береться із довідникової літератури. Дійсний опір резистора можна підрахувати по формулі:

$$R_T = R_o [1 + £_R (T_2 - T_1)],$$

де R_o – опір резистора при температурі 27°С;

T_1 – номінальна температура транзистора (27 °С);

T_2 – робоча температура транзистора;

$£_R$ – температурний коефіцієнт на 1 градус Цельсія.

У системі NI Multisim джерела напруги і струму є ідеальними. Внутрішній опір ідеального джерела напруги дорівнює нулю, тому його вихідна напруга не залежить від навантаження. Ідеальне джерело струму має нескінченно великий внутрішній опір, тому його струм не залежить від опору навантаження.

Для побудови ВАХ резисторів можна використовувати схему, що приведена на рис.1.1. У схемі використовується джерело е.р.с., на яке навантажується опір, що досліджується. У цій схемі зняття даних для побудови ВАХ забезпечується шляхом зміни установочних значень е.р.с. з записом відповідних значень струму, які контролюються амперметром. В другій схемі, навпаки, необхідно змінювати встановлені значення величини струму джерела струму, а напруга замірюється за допомогою вольтметра.

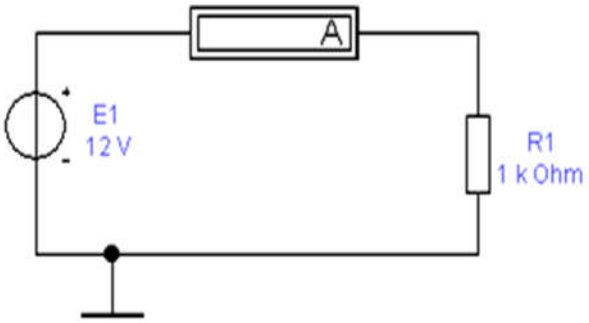


Рис. 1.1 – Схема для побудови ВАХ резистора

Для проведення дослідів по визначенню ВАХ реальних джерел живлення необхідно, перш за все, створити такі реальні джерела. Для цього послідовно з джерелом е.р.с. встановлюється активний опір (r_1), величина

якого повинна бути досить малою. Вона задається у відповідних варіантах. Схема, яка використовується для проведення вимірювань з метою побудови ВАХ реального джерела напруги, наведена на рис. 1.2.

Виконання роботи

Використовуючи схему, зображену на рис. 1.1, зробити заміри для побудови ВАХ резистора. Експериментальна ВАХ заміряється при температурі 27^0С (номінальний режим роботи резистора).

Встановити, відповідно до варіанту, лінійний коефіцієнт температурної залежності **TC1** і знову провести заміри в тих же точках. Перед тим, як ввести температурний коефіцієнт, треба встановити температуру, при якій працюватиме резистор (У вікні **Resistor Properties** вибрати вкладку *Analysis Setup*, зняти флажок *Use global temperature* та встановити температуру за варіантом).

Побудувати ВАХ ідеального джерела живлення.

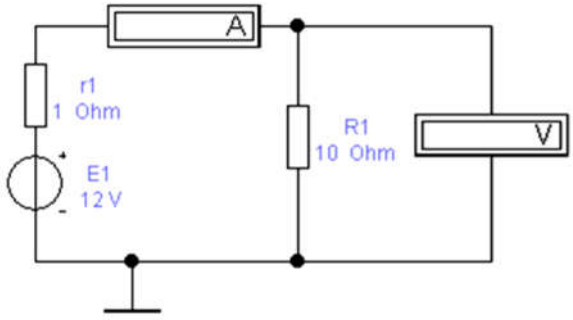


Рис. 1.2 - ВАХ ідеального джерела живлення

Побудувати схеми за рис.1.2 і, відповідно до варіанту у додатку, встановити параметри електричного кола. Виконуючи заміри в 2-х - 3-х точках, побудувати ВАХ реального джерела напруги. Повторити дослід для послідовного з'єднання двох джерел напруги (E1-r1 – E2-r1).

Розробити схеми і провести досліди для визначення еквівалентного опору при паралельному з'єднанні резисторів; при їх послідовному з'єднанні (взяти опори резисторів з додатку – R1а та R1б).

Завдання

1. Побудувати ВАХ резистора.

2. Побудувати ВАХ резистора з лінійним коефіцієнтом температурної залежності **ТС1**.

3. Побудувати ВАХ ідеальних і реальних джерел живлення.

4. Привести і обґрунтувати розроблені схеми для заміру еквівалентних параметрів резисторів. Порівняти експериментальні результати з обчисленими. У звіті необхідно привести схеми, таблиці і графіки проведених досліджень.

Таблиця варіантів до схем, що приведені на рис. 1.1. та 1.2.

Номер варіанту	**ТС1**	**Т, град**	**Е1, В**	**Е2, В**	**r1, Ом**	**R1а, Ом**	**R1б, Ом**
1	0.1	25	12	200	1.0	12	100
2	0.2	30	24	100	1.0	30	200
3	0.1	24	36	150	2.0	60	300
4	0.05	31	30	170	1.0	50	350
5	0.15	23	20	190	1.0	40	50
6	0.18	32	20	180	1.0	40	250
7	0.3	22	25	300	1.0	80	150
8	0.21	33	15	120	1.0	60	200
9	0.22	21	10	150	2.0	65	400
10	0.23	34	30	170	1.0	34	280
11	0.3	20	35	190	1.0	55	550
12	0.27	35	14	140	1.0	45	320
13	0.11	19	20	170	2.0	60	250
14	0.17	36	25	190	1.0	65	150
15	0.15	18	15	180	1.0	34	200

Практична робота № 2

Особливості послідовного з'єднання пасивних елементів

Основні теоретичні положення

Для електричного кола (Рис. 2.1), що живиться від ідеального джерела синусоїдального струму:

$$i(t) = I_m \sin(\omega t + \psi_i) .$$

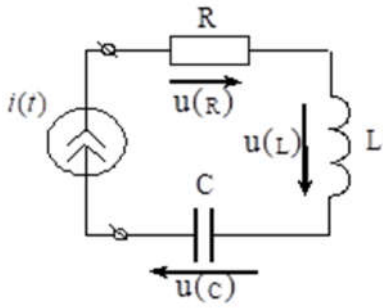


Рис. 2.1 – Ідеальне джерело синусоїдального струму

В відповідності до другого закону Кірхгофа, для миттєвих значень напруг маємо:

$$u = u_R + u_L + u_C \qquad (2.1)$$

Враховуючи лінійність елементів схеми, напруги на них u_R, u_L і u_C також змінюватимуться за синусоїдальним законом.

Останнє твердження дає можливість використовувати комплексну форму запису рівняння, тобто:

$$\dot{U} = \dot{U}_R + \dot{U}_L + \dot{U}_C . \qquad (2.2)$$

Враховуючи, що через кожен з елементів протікає один і той же струм, можемо записати:

$$\dot{U}_R = \dot{I}\, R; \qquad \dot{U}_L = j\omega L\, \dot{I}; \quad \dot{U}_C = -\frac{1}{j\omega C}\dot{I}$$

або

$$\dot{U}_R = \dot{I}\, R; \qquad \dot{U}_L = j\, X_L\, \dot{I}; \; \dot{U}_C = -j\, X_C\, \dot{I} .$$

Це дає можливість рівняння (2.3) записати у вигляді:

$$\dot{U} = \dot{I}\,(R + j\, X_L - j\, X_C) = \dot{I}\,\underline{Z} . \qquad (2.3)$$

Величина $\underline{Z} = R + j(X_L - X_C)$ є *еквівалентним комплексним опором* кола і визначається як алгебраїчна сума комплексних опорів, що утворюють коло. Уявна складова $X_L - X_C = X$ називається *реактивним опором*, який в залежності від величини X_L і X_C може носити індуктивний $(X_L > X_C)$, ємнісний $(X_C > X_L)$ характер або мати нульове значення $(X_L = X_C)$.

На комплексній площі комплексний опір може бути зображеним як вектор, що замикає геометричну суму його складових (рис. 2.2), а сама геометрична сума називається *трикутником опорів*.

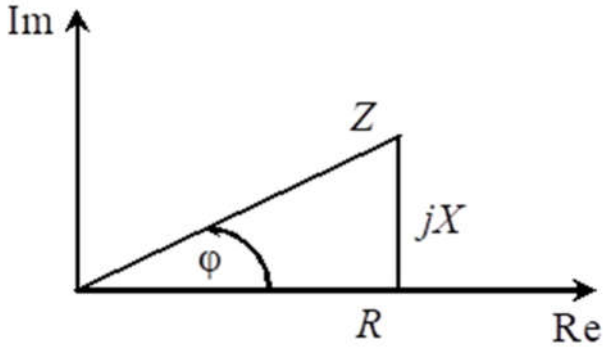


Рис. 2.2 – Зображення комплексного опору як вектора

З трикутника опорів, що представляє собою прямокутний трикутник, витікають наступні співвідношення:

$$\operatorname{tg}\varphi = \frac{X}{R}; \quad \cos\varphi = \frac{R}{Z}; \quad \sin\varphi = \frac{X}{Z};$$

$$Z = \sqrt{R^2 + X^2}; \quad R = Z\cos\varphi; \quad X = Z\sin\varphi; \quad \varphi = \operatorname{arctg}\left(\frac{X}{R}\right).$$

Використовуючи комплексну форму зображення опору $\underline{Z} = Z\,e^{j\varphi}$ та закон Ома, можемо знайти напругу на клемах джерела струму:

$$\dot{U}_m = \dot{I}_m \underline{Z} = I_m e^{j\psi_i} Z e^{j\varphi} = I_m Z e^{j(\psi_i+\varphi)} = U_m e^{j(\psi_i+\varphi)} = U_m e^{j\psi_u}.$$

Як результат цих нескладних перетворень, можемо записати:

$$\dot{U}_m = \dot{I}_m \underline{Z}; \qquad \psi_u = \psi_i + \varphi.$$

Останні формули надають можливість обчислити амплітуду та фазу (по відношенню до реальної осі координат) напруги на клемах джерела.

На рис. 2.3 проведені векторні діаграми напруг при початковій фазі струму $\psi_i = 0$ для трьох випадків: $X_L > X_C$ (Рис. 2.3а); $X_L < X_C$ (Рис. 2.3б);

$X_L = X_C$ (Рис. 2.3в), де $\dot{U}_{\text{р}} = \dot{I}\, j(X_L - X_C)$. З приведених діаграм видно, що кут φ може приймати як позитивні, так і негативні значення.

При $X_L = X_C$ маємо особливий режим, коли геометрична сума напруг на реактивних елементах дорівнює нулю. Такий режим називається *резонансом напруг.* Якщо опори рівні, то напруги теж будуть рівні $\dot{U}_L = \dot{U}_C$, а напруги на елементах будуть такі:

$\dot{U}_R = \dot{I}\, R$; $\dot{U}_L = j\, X_L\, \dot{I}$; $\dot{U}_C = -j\, X_C\, \dot{I}$.

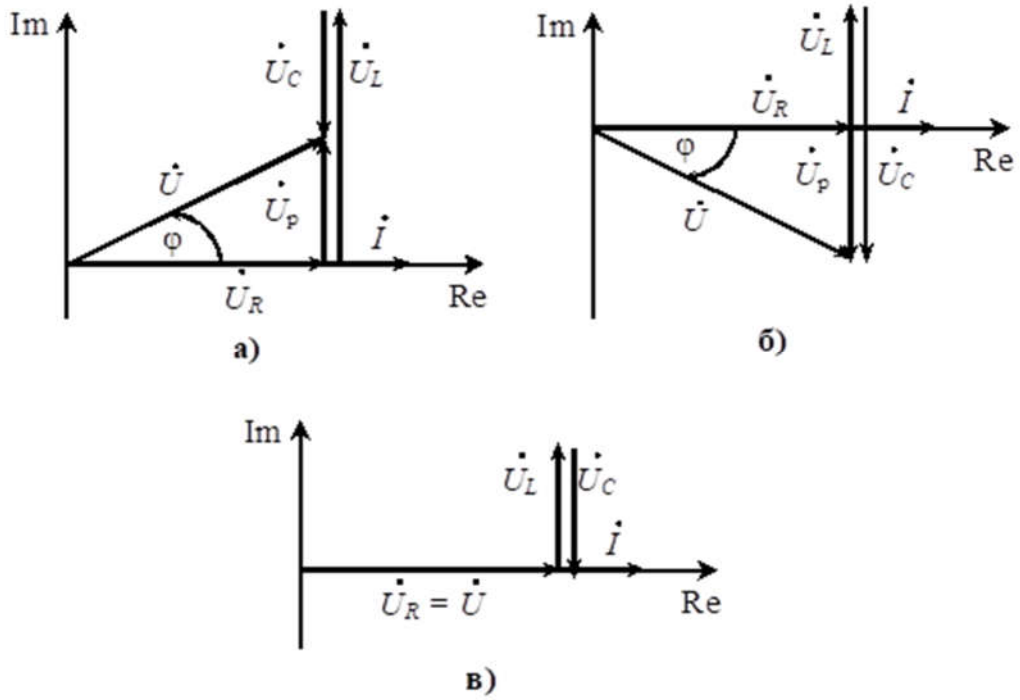


Рис. 2.3 – Векторні діаграми напруг

Виконання роботи

Послідовність проведення дослідів.

1. Скласти електричне коло, що містить в собі послідовно з'єднані джерело змінної напруги та активний опір з параметрами, що задаються в відповідності до варіанту. Заміряти напруги на резисторі та величину струму. Не слід забувати, що прилади повинні бути встановлені в режим вимірювання змінної напруги (АС). Виконати перевірку відповідності розрахункових та експериментальних даних. Підключити осцилограф та зняти часову характеристику.

2. Повторити пункт 1, але замість резистора встановити котушку (рис. 2.4).

3. Повторити пункт 1, але замість резистора встановити конденсатор.

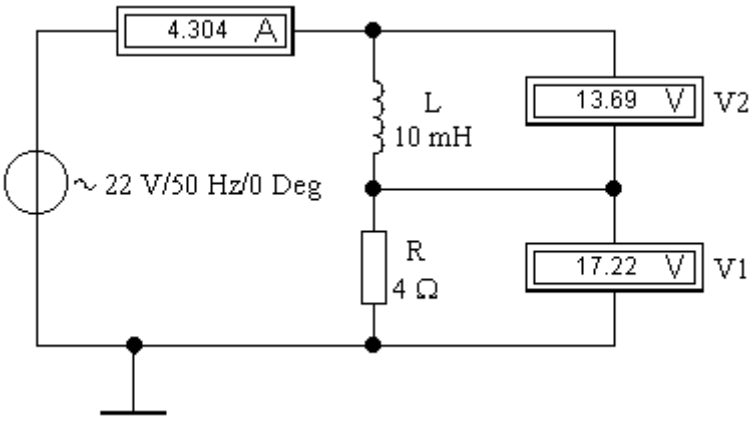


Рис. 2.4 – Електричне коло з котушкою

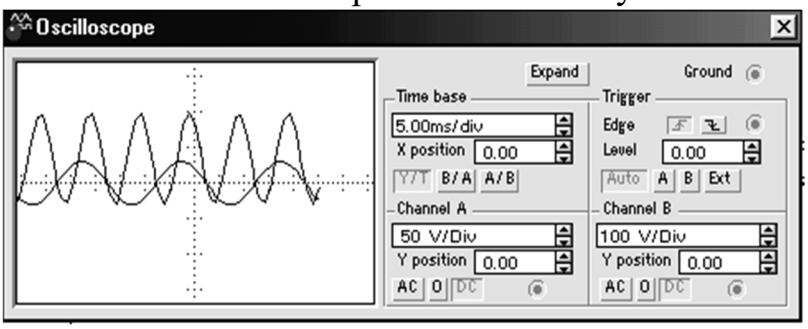


Рис. 2.5 – Покази осцилографа до схеми 2.4

4. Провести дослід з послідовним з’єднанням конденсатора, індуктивності й активного опору з джерелом напруги (Рис. 2.6). Заміряти і записати напруги й струм на елементах схеми, а також зняти осцилограми. Змінюючи частоту джерела, знайти таке її значення, при якому буде мати місце резонанс напруги в електричному колі. Заміряти й записати показання приладів і зняти осцилограми. Резонансну частоту можна визначити за допомогою Bode-plotter (Рис. 2.7).

5. Отримати резонанс напруг змінюючи номінали реактивних елементів.

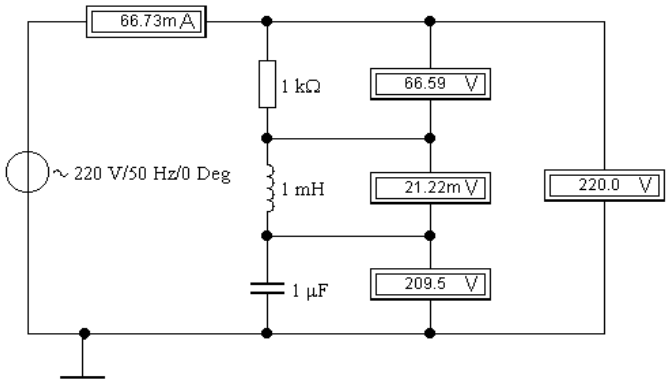


Рис. 2.6 – Схема активного опору з джерелом напруги

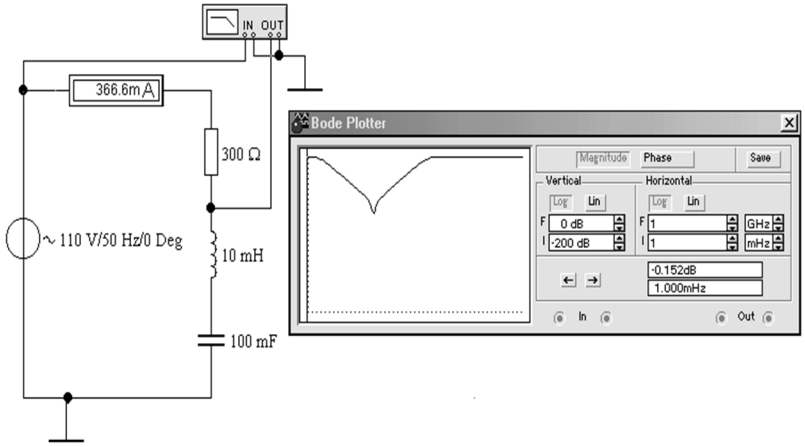


Рис. 2.7 – Резонансна частота

Завдання

1. Детально описати способи вимірювання амплітуди і частоти і фази синусоїдального сигналу й різниці фаз між двома сигналами за допомогою осцилографа. Привести відповідні осцилограми.

2. Привести векторні діаграми для визначення результуючої напруги і її фазового кута.

3. Привести часові діаграми схем з активним опором, індуктивністю, ємністю.

4. У звіті необхідно привести схеми і графіки проведених досліджень.

Номер варіанта	**E,** В	**F,** Гц	**φ, гр.**	**C,** мкФ	**R,** кОм	**L,** Гн
1	10	40	90	1	0.4	0.1
2	20	50	120	2	0.5	0.2
3	30	60	150	3	0.6	0.3
4	40	400	180	4	0.7	0.4
5	45	800	210	5	0.8	0.5
6	50	1к	240	6	0.9	0.6
7	55	2к	270	7	1	0.7
8	60	3к	300	8	2	0.8
9	65	4к	330	9	3	0.9
10	70	5к	0	10	4	1
11	75	6к	30	11	5	1.1
12	80	7к	60	12	6	1.2
13	85	8к	45	13	7	1.3
14	90	9к	80	14	8	1.4
15	95	10к	100	15	9	1.5

Практична робота № 3

Дослідження фільтрів

Основні теоретичні положення

Електричні фільтри – це пасивні чотириполюсники, які призначені для забезпечення необхідних частотних властивостей вихідного сигналу.

Смуга частот вхідного сигналу розділяється фільтром на діапазон зі слабким затуханням, який називається *смугою пропускання* фільтра, і діапазон значного затухання, що називається *смугою затухання*.

Дійсно, починаючи з низьких частот, величина його опору визначатиметься конденсатором і, відповідно, зменшуватиметься зі зростанням частоти до $\omega = \omega_з$. Починаючи з $\omega_з$, опір конденсатора буде мати величину меншу активного опору, і подальше його зменшення не призводитиме до реального зменшення опору двополюсника.

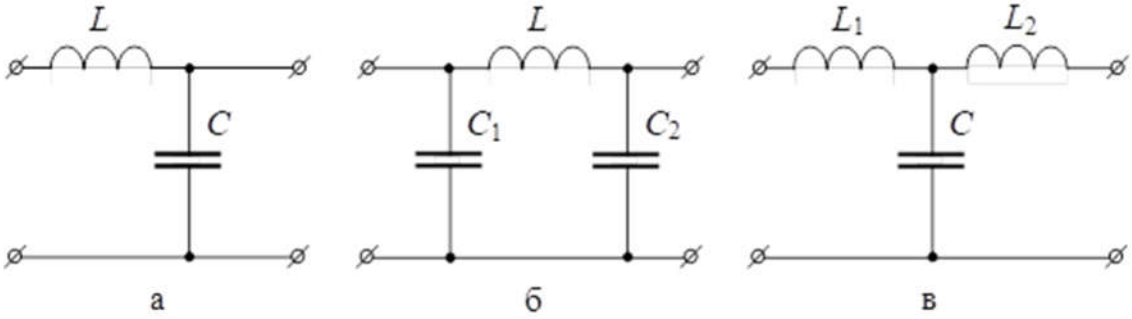


Рис. 3.1 – Фільтри низьких частот

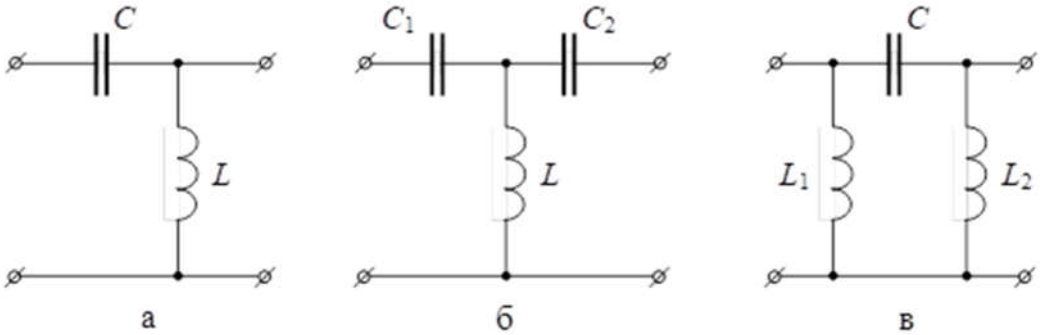


Рис. 3.2 – Фільтри високих частот

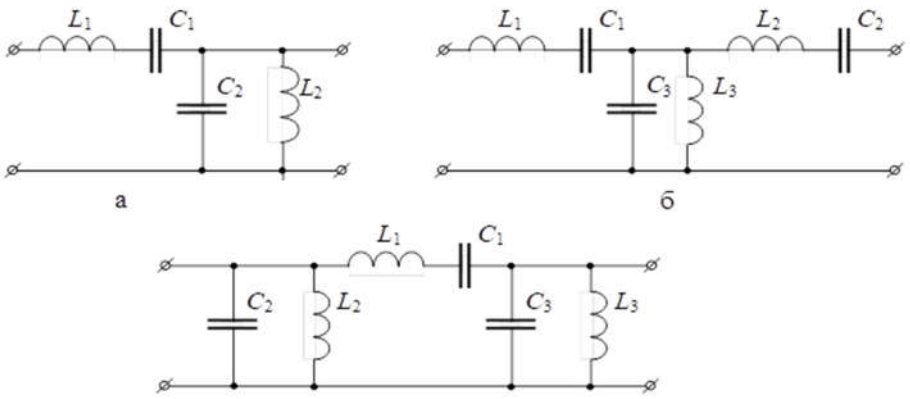


Рис. 3.3 – Смугові фільтри

Розглянемо схеми деяких фільтрів (див. рис. 3.1 – 3.3) .

Фільтри низьких частот (рис. 3.1) характерні тим, що індуктивність, яка стоїть в подовжній гілці, має на низьких частотах малий опір, а конденсатори, що встановлюються в поперечній ланці, – високий.

Фільтри високих частот (рис. 3.2), навпаки, в подовжній гілці містять конденсатор, який на високих частотах має низький опір, а індуктивність поперечної гілки на високих частотах має високий опір.

Смугові фільтри (рис. 3.3) у подовжній гілці мають послідовний резонансний контур, що на резонансній частоті має нульовий реактивний опір, а в поперечній – паралельний резонансний контур з нескінченно великим опором.

Загороджуючі фільтри (рис. 3.4) мають зворотне вмикання резонансних кіл.

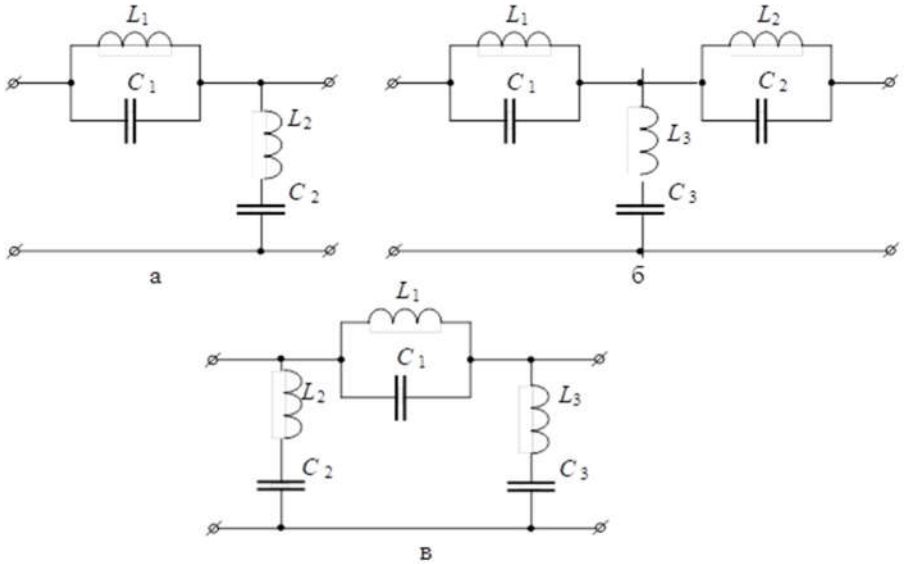


Рис. 3.4 – Загороджуючі фільтри

Особливість наведених схем фільтрів полягає в тому, що двополюсники подовжньої та поперечної гілок є взаємо оберненими. Це означає, що добуток їх комплексних опорів в усьому частотному діапазоні становить

$$\frac{1}{2}\ \underline{Z}_1 \cdot 2\underline{Z}_2 = \underline{Z}_1\ \underline{Z}_2 = K^2,$$

де K – дійсне число, є величиною сталою і не залежить від частоти. Такі фільтри називаються фільтрами типу K.

Без індуктивні фільтри використовуються при побудові фільтрів, що працюють на низьких частотах. Для таких частот індуктивність є занадто громіздкою і замінюється активним опором. Тому такі схеми називаються RC-фільтрами низьких частот (рис. 3.5).

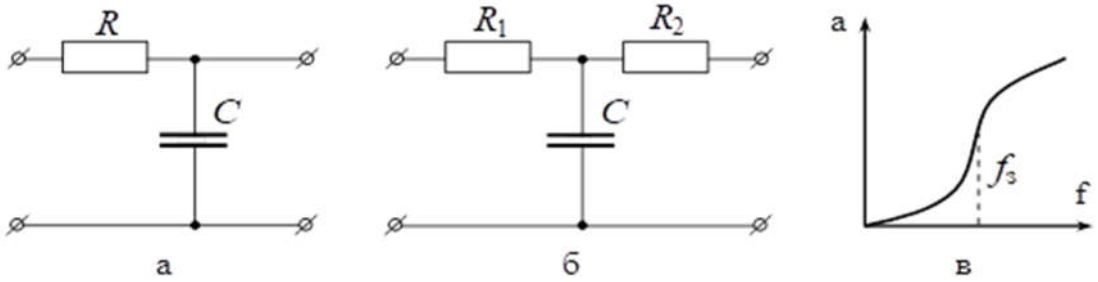


Рис. 3.5 – RC-фільтри низьких частот

Особливість таких фільтрів полягає в низький крутизні затухання, а також у тому, що, на відміну від реактивних фільтрів, *RC*-фільтри не мають області частот, у межах яких власне затухання $a = 0$.

Частотні характеристики *RC*-фільтрів зручніше визначати в логарифмічному масштабі через комплексний коефіцієнт передачі.

Фільтри низьких частот (*low-pass filter*) являють собою пристрої, що видаляють із сигналу середні та високі частоти.

Фільтр високих частот (*high-pass filter*) є пристроєм, що передає без змін сигнали високих частот, а на низьких частотах забезпечує затухання сигналів і випередження їх за фазою відносно вхідних сигналів.

Розглянемо схему простого фільтра високих частот (рис. 3.6).

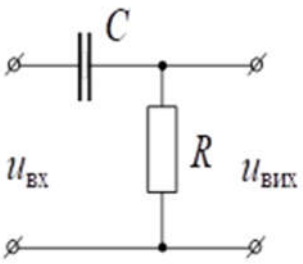


Рис. 3.6 – Простий фільтр високих частот

Комплексний коефіцієнт передачі

$$K(j\omega)=\frac{U_{вих}}{U_{вх}}=\frac{R}{R+\frac{1}{j\omega C}}=\frac{1}{1+\frac{1}{j\omega RC}}=\frac{j\omega RC}{1+j\omega RC}\ .$$

Стала часу $\tau = RC$.

Виконання роботи

1. Зібрати схему згідно таблиці варіантів та зняти амплітудно-частотну характеристику фільтра.

2. Перетворити схему, наприклад; якщо був фільтр низьких частот, внести зміни і отримати фільтр низьких частот та навпаки. Зняти АЧХ.

3. Заміряти частоти зрізу обох фільтрів та перевірити за допомогою формули.

4. Заміряти полосу пропускання фільтрів.

5. Підключити до схеми осцилограф для визначення амплітуди, періоду та зсуву фаз вхідного та вихідного сигналу кожного фільтра.

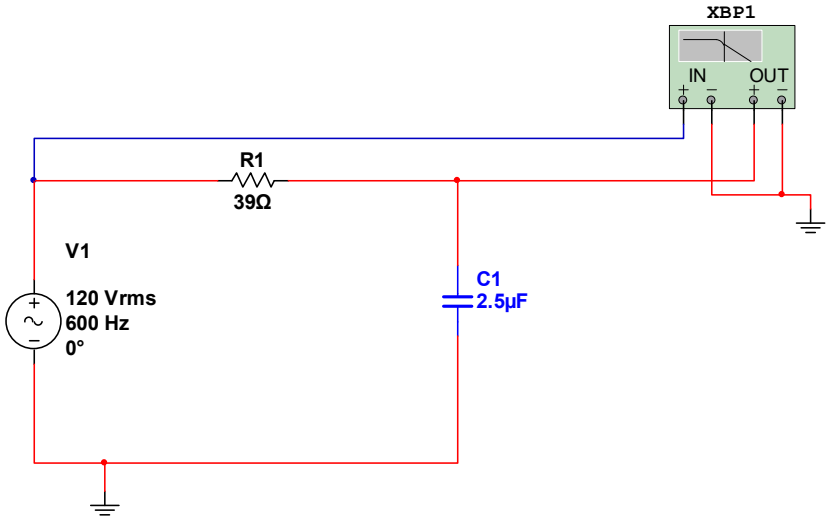


Рис. 3.7 – Приклад схеми

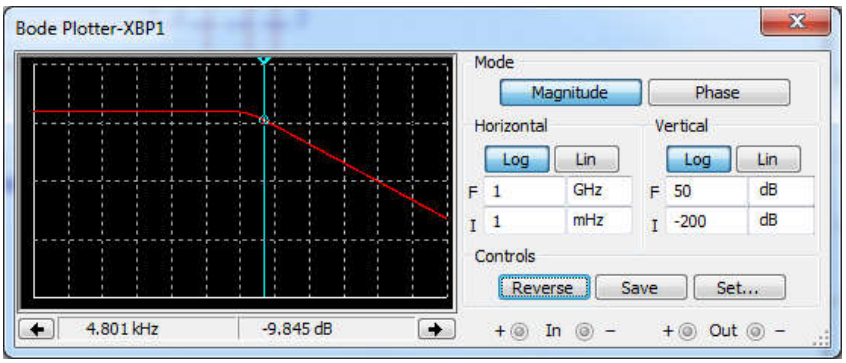


Рис. 3.8 – Осцилограф для визначення амплітуди

Завдання

1. Детально описати способи вимірювання амплітуди і частоти і фази синусоїдального сигналу й різниці фаз між двома сигналами за допомогою осцилографа. Привести відповідні осцилограми.

2. Привести часові діаграми схем з активним опором, індуктивністю, ємністю.

4. У звіті необхідно привести схеми і графіки проведених досліджень.

5. Зробити власні висновки.

Варіанти завдань

№ варіанта	*R–L*	*R–C*	*R*, Ом	*C*, мкФ	*L*, мГ
1	+		500	1	29
2		+	600	2	45
3	+		700	3	24
4		+	800	4	78
5	+		900	5	25
6		+	1000	6	35
7	+		2000	7	49
8		+	3000	8	82
9	+		4000	9	47
10		+	5000	10	19
11	+		6000	11	84
12		+	7000	12	46
13	+		8000	13	28
14		+	9000	14	67
15	+		1000	15	70

Практична робота № 4

Дослідження схем випрямлення змінної напруги

Вивчення особливостей роботи і основних характеристик малопотужних випрямлячів, які використовуються для живлення електронної апаратури при різному характері навантаження.

Основні теоретичні положення

Дослідження двонапівперіодного випрямляча з середньою точкою виконується з використанням схеми, що приведена на рис. 4.1. Ця схема дає можливість заміряти миттєві значення і амплітуди напруг на вході трансформатора та на навантаженні. Діоди, що використовуються в схемі, вибираються з розділу *Diode.* Вибір діода забезпечується бібліотекою діодів. В кожнім з елементів бібліотеки можна редагувати параметри. Для цього необхідно натиснути кнопку *Edit model*, після чого з'являється вікно з параметрами SPICE-моделі діода:

– *Saturation current* (IS) – струм насищення.

– *Ohmic resistance* (RS) – об'ємний опір діода.

– *Zero-bias junction capacitance* (CJO) – бар'єрна ємність *p-n* переходу при нульовій напрузі.

– *Junction potential* (VJ) – контактна різниця потенціалів.

– *Transit time* (Tt) – час переносу зарядів.

– *Grading coefficient* (M) – конструктивний параметр *p-n* переходу.

– *Reverse breakdown voltage* (BV) – максимальна зворотна напруга (для стабілітронів не нормується).

– *Emission coefficient* (N) – коефіцієнт інжекції.

– *Activation energy* (EG) – ширина забороненої зони.

– *Temperature exponent for effect on IS* (XTI) – температурний коефіцієнт струму насищення.

– *Flicker noise coefficient* (KF) – коефіцієнт флікер-шуму.

– *Flicker noise exponent* (AF) – показник ступеня у формулі для флікер-шуму.

– *Coefficient for forward-bias depletion capacitance formula* (FC) – коефіцієнт нелінійності бар'рної ємності прямо зміщеного переходу.

– *Current at reverse breakdown voltage* (IBV) – початковий струм пробою при напрузі BV.

– *Parameter measurement temperature* (TNOM) – температура діоду.

Приведена схема дає можливість вивчити процес перетворення енергії змінного струму в постійний та встановити відповідність між фізичними явищами перетворення енергії і математичними формулами.

В схемі використовується ідеальний трансформатор (*Transformer*) з панелі *Basic.* Для побудови зовнішньої характеристики випрямляча необхідно встановити реальні параметри трансформатора. Для цього треба двічі натиснути лівою кнопкою миші по ньому, у вікні, що відчиниться, натиснути кнопку *Edit,* у наступному вікні виставити коефіцієнт перетворення (*Primary-to-Secondary turns ratio*) – 1, а опір вторинної обмотки (*Secondary winding resistance*) – 1 Ом.

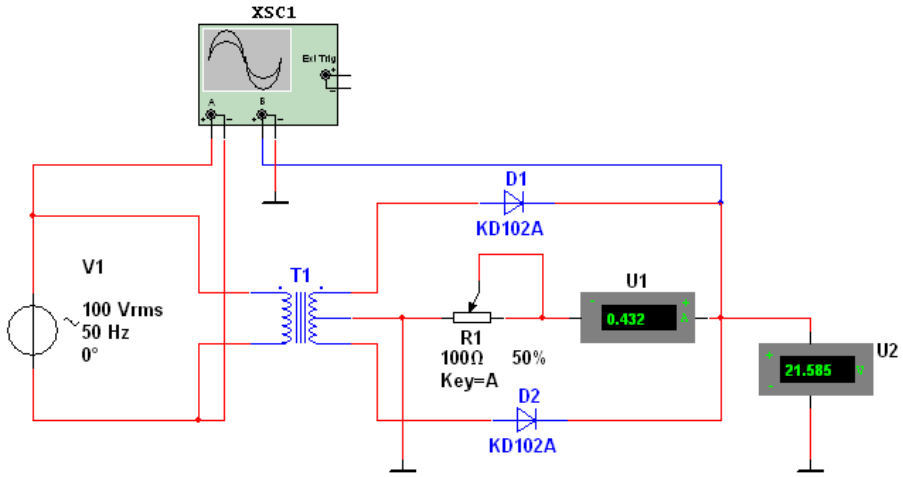


Рис. 4.1 – Двонапівперіодний випрямляч з середньою точкою

Осцилограф покаже форму вхідної та вихідної напруги. Щоб розрізняти вхідний та вихідний сигнали рекомендується провідники, які приєднують осцилограф до схеми, виділити різними кольорами. В цьому випадку сигнали осцилографа будуть відображені тим кольором, який має провідник цього

сигналу. Натиснувши правою кнопкою миші по провіднику, та вибрав опцію *Color Segment…*, можна змінити колір провідника. Часові діаграми вхідної і вихідної напруги можна отримати завдяки *Transient Analysis* з меню *Analysis.* Для побудови зовнішньої характеристики випрямляча можна використовувати потенціометр (Potentiometer) з розділу Basic. Вікно Potentiometer Properties має такі опції:

– *Key* – ключ для регулюювання установкою.

– *Resistance* (R) – початковий опір.

– *Increment* – шаг зміни установки (в відсотках).

Вивчення роботи однофазного мостового випрямляча забезпечується за допомогою схеми, що приведена на рис. 1.1. Вона дає можливість як встановити залежності між напругами, що мають місце в процесі перетворення, оцінити вплив конденсатора на величину і вигляд випрямленої напруги, так і провести необхідні заміри для побудови зовнішньої характеристики випрямляча. В неї використовується саме той трансформатор, що і в попередній схемі.

Виконання роботи

Досліджується схема однофазного однонапівперіодного випрямляча.

Досліджується схема однофазного двонапівперіодного випрямляча з середньою точкою.

Збирається схема за рис. 4.1, вмикається в роботу і виконуються наступні заміри за допомогою приладів:

- максимальне і мінімальне значення випрямленої напруги;
- середнє значення випрямленої напруги;
- середнє значення струму, що протікає через навантаження.

Замір максимального та мінімального значення випрямленої напруги виконується по часовій діаграмі, що отримується за допомогою *Transient Analysis* з меню *Analysis.* Зняття середніх значень виконується по вольтметру та амперметру, які встановлюються в режим DC.

Проводиться спектральний аналіз (*Фур'є-аналіз)* вихідної напруги випрямляча за допомогою *Fourier Analysis* з меню *Analysis* і записується значення параметра, який називається *Total harmonic distortion* (коефіцієнт гармонік).

Паралельно до навантаження встановлюється конденсатор і виконуються заміри максимального і мінімального значення випрямленої напруги (слід перед цим встановити опір обмотки трансформатора в нуль і діоди замінити на ідеальні). Підбирається потрібна ємність. Доцільніше буде зробити це за допомогою *Parameter Sweep* з меню *Analysis.* По отриманим сімействам перехідних характеристик зробити 4 - 5 замірів максимального і мінімального значень випрямленої напруги при різних значеннях індуктивності. Провести спектральний аналіз вихідної напруги випрямляча і записати значення коефіцієнта гармонік.

Для декількох значень ємності, за допомогою *Fourier Analysis* з меню *Analysis*, провести спектральний аналіз струму в навантаженні і записати значення параметра, який називається *Total harmonic distortion* (коефіцієнт гармонік).

Перейти до схеми мостового випрямляча з активно-ємнісним навантаженням. За допомогою *Parameter Sweep* з меню *Analysis*, збільшувати ємність конденсатора до величини, коли змінна складова не буде перевищувати 10% від середнього значення випрямленої напруги і зафіксувати параметри спектру вихідної напруги.

Для отриманого значення активно-ємнісного навантаження провести заміри в 4 - 6 точках зовнішньої характеристики випрямляча. Зовнішня характеристика знімається так само, як для двонапівперіодного випрямляча.

Зібрати схему подвоєння напруги і, збільшуючи ємність конденсаторів, виконати заміри середнього значення випрямленої напруги.

Завдання

1. Побудувати часові діаграми випрямленої напруги при активному, навантаженні випрямляча.

2. Дати приклади спектральних діаграм випрямленої напруги.

3. Розрахувати коефіцієнти пульсацій випрямленої напруги, використовуючи формулу:

$$K_{П} = \frac{U_{MAX} - U_{MIN}}{2U_{d}}$$

Побудувати графічні залежності параметра $K_П$ від параметрів реактивних елементів в навантаженні випрямляча.

Таблиця варіантів до схеми, що приведена на рис. 4.1.

Номер варіанту	**U, В**	**R, Ом**	**С, мкФ**
1	100	2	4
2	110	3	6
3	120	4	8
4	130	5	10
5	140	6	12
6	150	7	14
7	160	8	16
8	170	9	18
9	180	10	20
10	190	11	22
11	200	12	24
12	210	13	26
13	230	15	30
14	240	16	32
15	250	17	34

Практична робота № 5

Дослідження схем параметричних стабілізаторів та обмежувачів змінної напруги

Вивчення особливостей роботи і використання параметричних стабілізаторів і обмежувачів напруги.

Основні теоретичні положення

Для дослідження роботи схеми параметричного стабілізатора готується схема відповідно до рис. 5.1. Параметри стабілітронів включають наступні опції (додатково до загальних для діодів):

– *Zener test current* (IZT), A – номінальний струм стабілізації (від одиниць до десятків мА).

– *Zener test voltage at IZT* (VZT), В – напруга стабілізації при номінальному струмі стабілізації.

– *Reverse breakdown voltage* (BV), В – для стабілітронів не нормується.

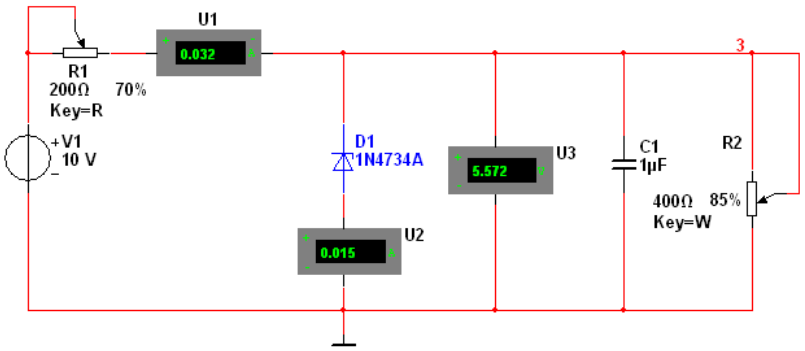


Рис. 5.1 – Параметричний стабілізатор

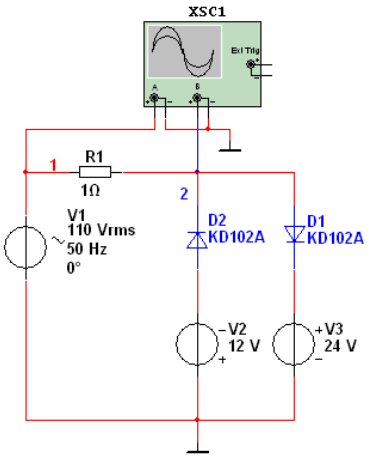


Рис. 5.2 – Обмежувач напруги

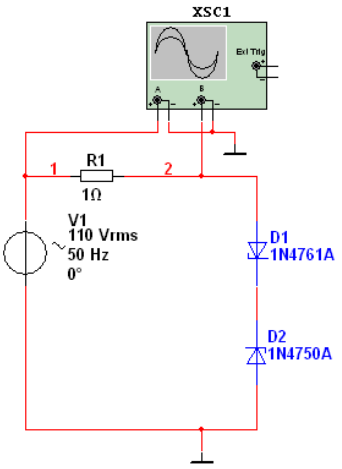


Рис. 5.3 – Обмежувач напруги

Виконання роботи

Готується схема параметричного стабілізатора напруги. Вибирається напруга стабілізації згідно варіанту. Вона може бути встановлена шляхом відповідного вибору стабілітрона.

Підбирається баластний опір. Він встановлюється таким чином, щоб напруга на вольтметрі дорівнювала напрузі стабілізації.

Обчислюється діапазон зміни напруги на навантаженні.

Готується схема обмежувача напруги.

Завдання

1. Привести пояснення роботи кожної з схем, що досліджувалась.

Таблиця варіантів до схем, які приведені на рис. 5.1-5.3.

Номер варіанту	**Рис. 5.1**			**Рис. 5.2, рис. 5.3**	
	U, В	**$U_{ст}$, В**	**$I_н$, мА**	**U, В**	**$U_{об}$, В**
1	10	6	10	5	1
2	12	7	20	8	2
3	14	8	30	11	3
4	16	9	40	14	5
5	18	10	45	17	10
6	20	11	50	20	10
7	22	12	55	5	2
8	25	12	60	8	4
9	22	11	50	11	6
10	20	10	55	14	10
11	18	9	40	17	15
12	16	8	45	20	15
13	14	7	30	5	3
14	12	6	35	8	6
15	10	6	20	11	9

Практична робота № 6

Арифметичні цифрові пристрої

З’ясування принципів роботи суматорів, компараторів, арифметико-логічних пристроїв (АЛП), пристроїв контролю парності, які входять до складу комп’ютерної техніки. Побудова більш складних арифметичних модулів на їх основі.

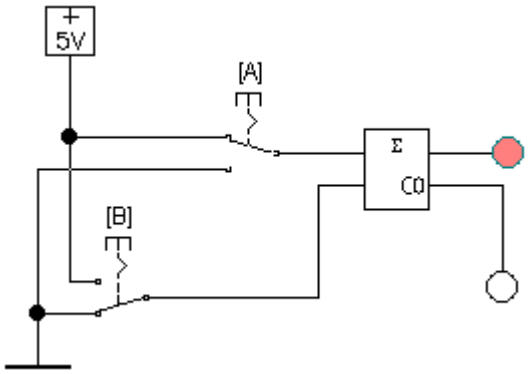
Основні теоретичні положення

Суматори. *Суматором* називається цифровий електронний пристрій, який виконує операцію арифметичного додавання кодів двох чисел. Суматори можуть також застосовуватись для виконання операції віднімання, але для цього необхідно здійснити додаткові перетворення кодів чисел. Суматори є складовою частиною арифметико-логічних пристроїв (АЛП) мікропроцесорів. Також вони використовуються для формування фізичної адреси комірок пам’яті у мікропроцесорах з сегментною організацією пам’яті.

Напівсуматор характеризується наявністю двох входів, на які подаються однорозрядні числа, і двох виходів, на одному з яких (Σ) реалізується арифметична сума у даному розряді, а на другому (СО) – перенос у наступний (більш старший) розряд. Отже, напівсуматор виконує лише частину операції підсумовування, оскільки не враховує ще однієї вхідної величини – переносу з сусіднього молодшого розряду даних, а тому може використовуватись для знаходження суми лише у наймолодшому розряді.

Повні однорозрядні двійкові суматори характеризуються наявністю трьох входів, на два з яких подаються однорозрядні числа, а на третій (СІ) – перенос із попереднього (більш молодшого) розряду, і двома виходами, на одному з яких (Σ) формується арифметична сума у даному розряді, а на другому (СО) – сигнал переносу в наступний (більш старший) розряд.

Схеми включення напівсуматора і повного однорозрядного суматора показані відповідно на рис.6.1 - а і б.



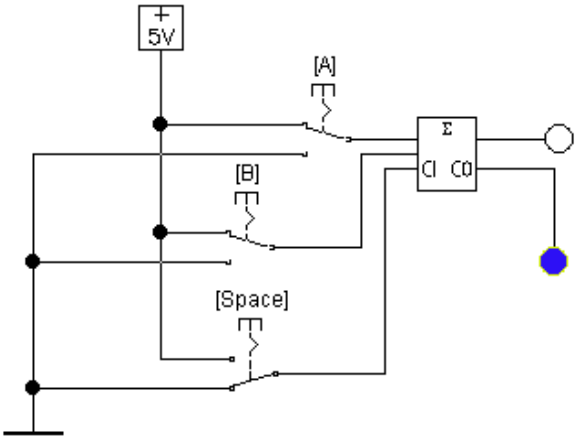


Рис.6.1(а) – Напівсуматор

Рис.6.1(б) – Однорозрядний суматор

Багаторозрядний суматор можна створити на базі одного напівсуматора і кількох повних суматорів. Як приклад, на рис. 6.2 приведена структура трьохрозрядного суматора з переносом. За допомогою ключів [A], [B], [C] та [X], [Y], [Z] подаються перший і другий доданки відповідно; з виходів S0, S1, S2 знімається результат підсумовування; з виходу CO подається сигнал переносу у старший розряд.

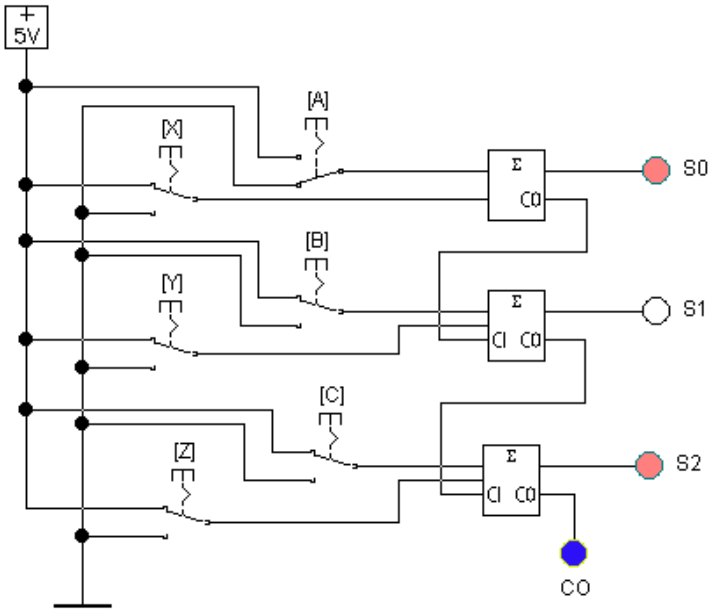


Рис.6.2 – Структура трьохрозрядного суматора з переносом

У бібліотеці присутня модель реальної мікросхеми чотирьохрозрядного повного суматора. Її зовнішній вигляд показаний на рис. 6.3.

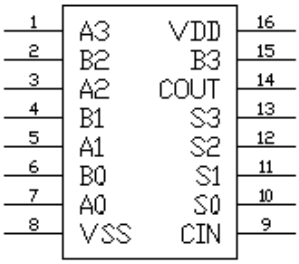


Рис.6.3 – Реальна мікросхема чотирьохрозрядного повного суматора

Призначення виводів такі: A3, A2, A1, A0 – розрядні входи першого додатку; B3, B2, B1, B0 – розрядні входи другого доданку; CIN – вхід переносу з попереднього (молодшого) розряду; S3, S2, S1, S0 – розрядні виходи суми; COUT – вихід переносу до наступного (старшого) розряду; VDD – вивід живлення; VSS – загальний. Схема включення мікросхеми приведена на рис. 6.4.

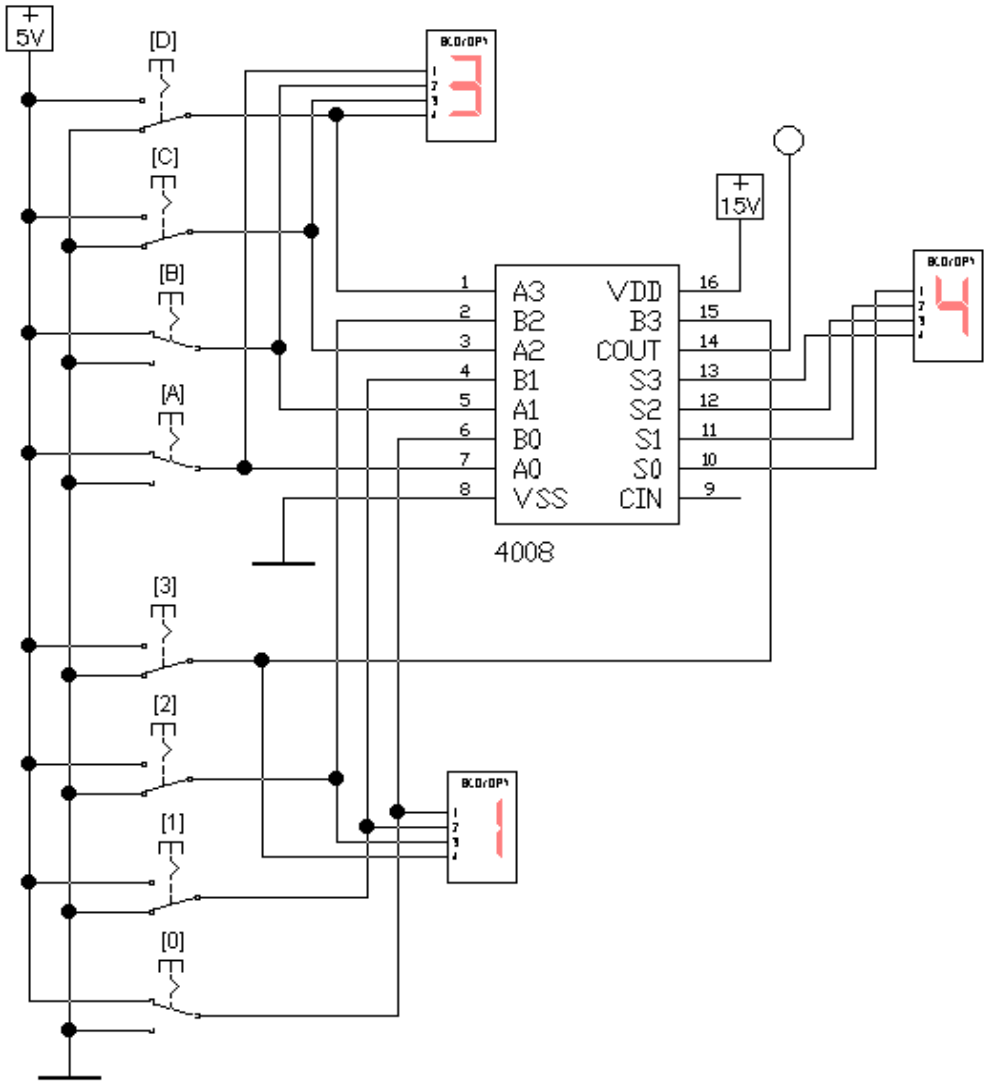


Рис. 6.4 – Схема включення мікросхеми

Компаратори. *Цифрові компаратори* (від англ. to compare – порівнювати) виконують порівняння двох чисел A і B однакової розрядності, заданих у війковому коді. В залежності від схемного виконання, компаратори можуть визначати рівність $A = B$ або нерівності $A < B$, $A > B$. Результат порівняння відображається у вигляді логічного сигналу на однойменних виходах. Цифрові компаратори застосовуються для виявлення потрібного числа (слова) у цифрових послідовностях, для відмітки часу в часових приладах, для здійснення умовних переходів в обчислювальних пристроях.

Схема однорозрядного компаратора приведена на рис. 6.5. Вона складається з двох елементів **НІ**, чотирьох елементів **І** та одного елементу **АБО**.

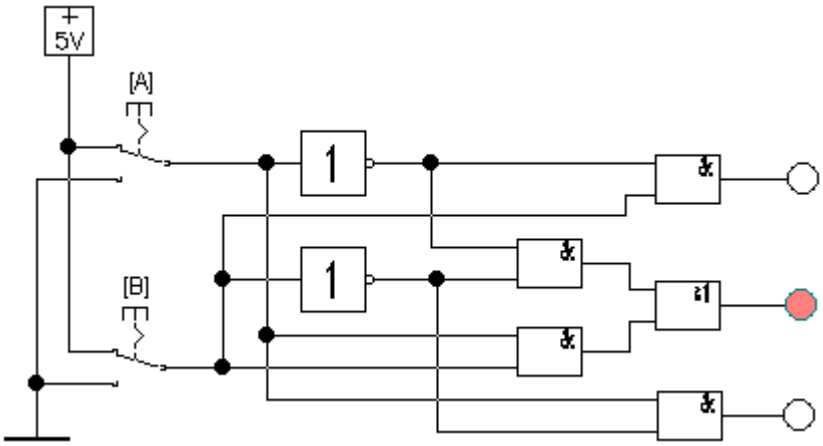


Рис. 6.5 – Схема однорозрядного компаратора

Виконання роботи

1. Готується і досліджується схема трьох- (чотирьох) розрядного суматора з переносом. На вхід подається число за варіантом.

2. Збирається схема цифрового однорозрядного компаратора. Проводиться її дослідження.

Завдання

У звіті повинні бути приведені:

- схеми включення повного суматора;
- схема однорозрядного цифрового компаратора, таблиця істинності, логічні вирази для кожного з режимів роботи;
- реальна мікросхема компаратора.

Практична робота № 7

Кодуючі пристрої

Вивчення алгоритмів роботи та принципів використання кодуючих та декодуючих пристроїв.

Основні теоретичні положення

Шифратори (кодери) і дешифратори (декодери) відносяться до пристроїв перетворення кодів. З операцією шифрації пов'язане поняття про стиснення інформації, а з операцією дешифрації – зворотне її перетворення.

Дешифратори. *Дешифратором* називається комбінаційний цифровий пристрій, призначений для перетворення паралельного двійкового коду в унітарний код. При подачі на його вхід паралельного двійкового коду активний сигнал з'явиться на тому його виході, номер якого відповідає десятковому еквіваленту вхідного коду; у будь-який момент часу вихідний сигнал має місце лише на одному виході дешифратора. В залежності від типу дешифратора, цей сигнал може мати як високий рівень (при цьому на всіх інших виходах встановлюються низькі рівні), так і низький рівень (при цьому на всіх інших виходах високі рівні сигналу). В дешифраторах кожній вихідній функції відповідає лише один мінтерм, а кількість функцій визначається кількістю розрядів двійкового числа. Якщо дешифратор реалізує усі мінтерми вхідних змінних, то він називається *повним*; інакше – *неповним* (в якості прикладу неповного дешифратора можна привести дешифратор двійково-десяткових чисел). Дешифратори широко використовуються для організації звернення до тих чи інших пристроїв, наприклад, до конкретних пристроїв пам'яті.

Інформацію про функціонування кожної конкретної мікросхеми можна отримати з таблиці істинності. Якщо цієї інформації недостатньо, необхідно скористатися довідковою літературою.

Загальний алгоритм роботи дешифратора можна зрозуміти, якщо дослідити модель ідеального дешифратора "3 на 8" (*Generic 3-to-8 Dec*), умовне зображення якого представлене на рис. 7.1 (а-б).

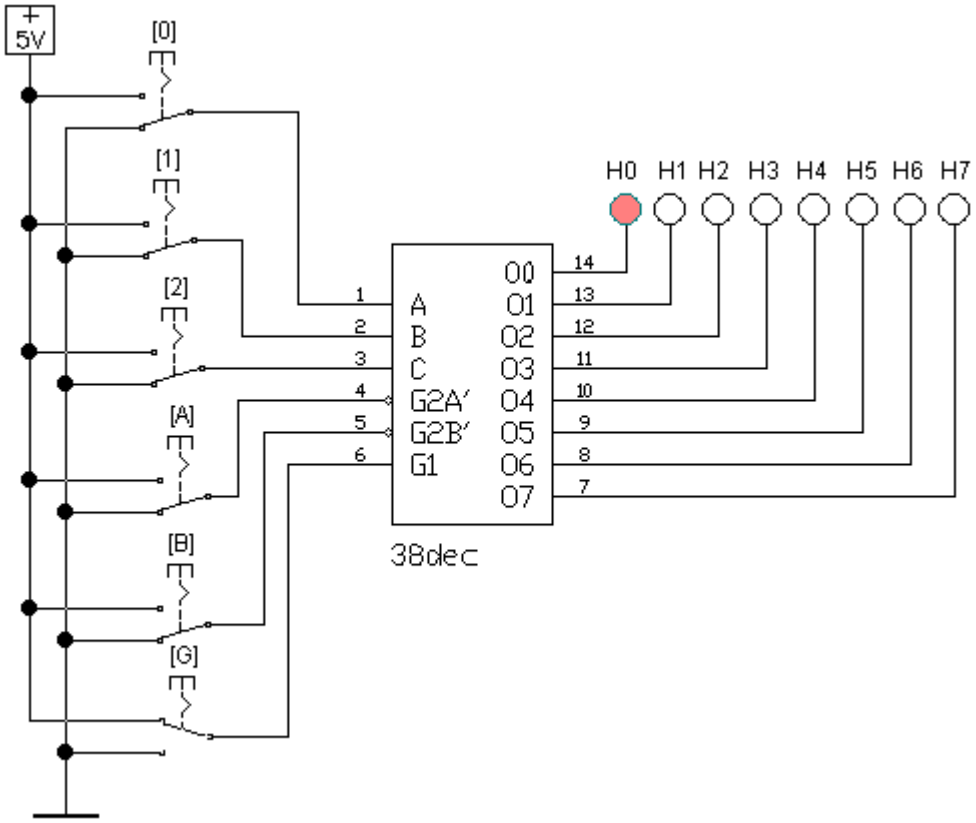


Рис. 7.1а – Модель ідеального дешифратора

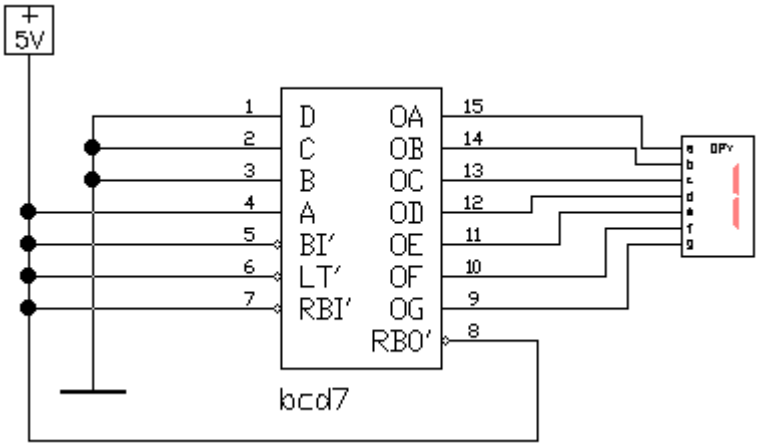


Рис. 7.1б – Модель ідеального дешифратора

Ось як описаний алгоритм роботи ідеального дешифратора у підказці програми: “Ідеальний дешифратор декодує одну з восьми ліній, номер якої визначають три керуючі входи ABC, якщо це дозволено сигналами на входах G1, G2A' і G2B'. Якщо сигнал на G1 має високий рівень, а G2A' і G2B' – низький, то на вихід передається десятковий еквівалент вхідного двійкового слова. Якщо рівень сигналу на G1 низький або на G2A' і G2B' – високий, то усі виходи неактивні.” У правильності цього алгоритму дозволяє пересвідчитись

схема (рис. 7.1), у якій вхідні перемикачі задають керуюче двійкове слово [0], [1], [2] і сигнали дозволу [A], [B], [G], а логічні пробники H0…H7 дозволяють фіксувати вихідні значення.

Шифратори. *Шифратором* називається комбінаційний цифровий пристрій, який здійснює перетворення позиційного коду в n-розрядний двійковий код, тобто виконує функцію, протилежну тій, яку виконують дешифратори. Шифратори можуть використовуватись для перетворення десяткових чисел у двійковий або двійково-десятковий код – наприклад, у мікрокалькуляторах, де натискання десяткової клавіші відповідає генерації відповідного двійкового коду. Часто у шифраторах використовується принцип пріоритету старшого розряду, і такі шифратори називаються *пріоритетними*. Пріоритетні шифратори знаходять використання для визначення номеру пристрою, що подає сигнал запиту на обслуговування в мікропроцесорних системах, визначення станів та номерів вимикачів у схемах автоматики і т. ін.

Виконання роботи

1. Дослідження роботи ідеального дешифратора. Вибирається ідеальний дешифратор, збирається схема і складається таблиця істинності пристрою.

2. Вивчення алгоритму роботи ідеального шифратора. Вибирається ідеальний шифратор і збирається схема для його дослідження. Складається таблиця істинності шифратора.

3. Вивчення роботи реальної мікросхеми шифратора/дешифратора. Вибирається одна з мікросхем і аналізується.

4. Дослідження роботи дешифратора семисегментного індикатора.

Завдання

У звіті повинні бути приведені:

- схеми кодових перетворювачів, а саме: шифраторів, дешифраторів (в тому числі, дешифраторів семисегментного індікатора);
- пояснення призначення виводів мікросхем;
- таблиці істинності досліджуваних цифрових пристроїв.

Практична робота № 8

Операційні та керуючі пристрої

Проектування операційних пристроїв

Основні теоретичні положення

Поняття мікропрограмного керування Переважна більшість цифрових пристроїв, в тому числі й комп'ютери, переробляють інформацію, виконуючи над нею деякі операції. Для виконання операцій над інформацією використовуються операційні пристрої – процесори, канали введення-виведення, пристрої управління зовнішніми пристроями. Функцією операційного пристрою є виконання заданої пристроїв визначається алгоритмом виконання операції $F=\{f_1,...,f_G\}$ над вхідними словами $D=\{d_1,...,d_H\}$ з метою обчислення слів $R=\{r_1,...,r_Q\}$, які подають результати операцій $R=f_g(D)$, де $g=1,2,...,G$.

Функціональна і структурна організація операційних пристроїв базується на принципі мікропрограмного управління, який полягає в такому:

1. Будь-яка операція $f_g(g=1,...,G)$, що реалізується пристроєм, розглядається як складна дія, яка розділяється на послідовність елементарних дій над словами інформації. Ці елементарні дії називаються мікрооперaціями.

2. Для управління порядком проходження мікрооперацій використовуються логічні умови, які залежно від значень слів, що перетворюються мікрооперaціями, набувають значення "істина" або "брехня", („true" або „false"), (1 або 0).

3. Процес виконання операцій в пристрої описується у формі алгоритму, який представляється в термінах мікрооперацій і логічних умов і називається мікропрограмою. Мікропрограма визначає порядок перевірки значень логічних умов і проходження мікрооперацій, необхідний для отримання потрібних результатів.

4. Мікропрограма використовується як форма подання функції пристрою, на основі якої визначається структура і порядок функціонування пристрою в часі.

Таким чином, із принципу мікропрограмного управління випливає, що структура і порядок функціонування операційних пристроїв визначається алгоритмом виконання операції $F=\{f_1,...,f_G\}$.

До елементарних дій над словами інформації, тобто мікрооперацій, належать: передача інформації з одного регістра в інший, утворення оберненого коду, зсув, додавання тощо.

Поняття операційного та керуючого автоматів. Як показав академік В.М. Глушков, у будь-якому пристрої обробки цифрової інформації можна виділити два основні блоки – операційний автомат (ОА) і керуючий автомат (КА) (рис. 8.1).

Операційний автомат (ОА) служить для зберігання слів інформації, виконання набору мікрооперацій і обчислення значень логічних умов, тобто операційний автомат є структурою, організованою для виконання дій над інформацією. Мікрооперації, що виконуються ОА, задаються множиною керуючих сигналів $Y=\{y_1,....,y_M\}$, з кожним з яких ототожнюється певна мікрооперація.

Значення логічних умов, що обчислюються в операційному автоматі, відображаються множиною інформаційних сигналів $X=\{x_1,...,x_L\}$, кожний з яких ототожнюється з певною логічною умовою.

Керуючий автомат (КА) генерує послідовність керуючих сигналів, вказану мікропрограмою і відповідну значенням логічних умов. Інакше кажучи, керуючий автомат задає порядок виконання дій в ОА, що визначається з алгоритму виконання операцій. Найменування операції, яку необхідно виконати в пристрої, визначається кодом g операції, що надходить у КА ззовні. По відношенню до КА сигнали $g_1,...,g_h$, за допомогою яких кодується найменування операції та інформаційні сигнали $x_1,...,x_L$, сформовані в операційному автоматі, відіграють однакову роль: вони впливають на порядок вироблення керуючих сигналів Y. Тому сигнали $g_1,...,g_h$ і $x_1,...,x_L$ належать до одного класу – до класу інформаційних сигналів, що надходять на вхід КА.

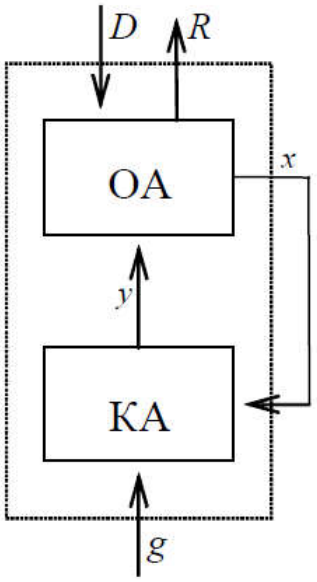


Рис. 8.1 – Декомпозиція цифрового пристрою на керуючий автомат (КА) та операційний автомат (ОА)

Таким чином, будь-який операційний пристрій – процесор, канал введення-виведення тощо – є композицією операційного та керуючого автоматів. Операційний автомат, реалізовуючи дії над словами інформації, є виконавчою частиною пристрою, роботою якого управляє керуючий автомат, що генерує необхідні послідовності керуючих сигналів.

Операційний та керуючий автомати можуть бути визначені своїми функціями – переліком виконуваних ними дій. Функція ОА визначається такою сукупністю відомостей:

1) множиною вхідних слів $D=\{d_1,...,d_H\}$, що вводяться в автомат як операнди;

2) множиною вихідних слів $R=\{r_1,...,r_Q\}$, що являють собою результати операцій;

3) множиною внутрішніх слів $S=\{s_1,...,s_N\}$, що використовуються для подання інформації в процесі виконання операцій. Можна вважати, що вхідні і вихідні слова збігаються з певними внутрішніми $D\subseteq S$, $R\subseteq S$;

4) множиною мікроопераций $Y=\{y_m\}$, що реалізують перетворення $S=\varphi_m(s)$ над словами інформації, де φ_m – обчислювана функція;

5) множиною логічних умов $X=\{x_i\}$, де $x_i=\psi_i(s_i)$, ψ_i – булева функція.

Отже, функція ОА задана, якщо задана (визначена) множина *D, R, S, Y, X*. Час не є аргументом функції ОА. Функція встановлює список дій-мікрооперацій і логічних умов, які може виконувати автомат, але ніяк не визначає порядок проходження цих дій у часі. Тобто функція ОА характеризує засоби, які можуть бути використані для обчислень, але не сам обчислювальний процес.

Порядок виконання дій у часі визначається у формі функцій керуючого автомата.

Функція керуючого автомата – це операторна схема алгоритму (мікропрограми), функціональними операторами якої є символи $y_1...,y_m$, ототожнювані з мікрооперaціями, і як логічні умови використовуються булеві змінні $x_1...,x_L$. Операторна схема алгоритму найбільш часто зображується у вигляді граф-схеми алгоритму (ГСА). ГСА визначає обчислювальний процес послідовно в часі, встановлюючи порядок перевірки логічних умов $x_1 - x_L$ і порядок проходження мікрооперацій $y_1 - y_m$.

Операційні елементи. Як уже зазначалося вище, згідно із принципом мікропрограмного керування, будь-яка складна операція розпадається на ряд мікрооперацій, які виконуються операційним автоматом (ОА). Різні мікрооперації виконуються елементарними ОА – так званими операційними елементами (ОЕ), які є складовими частинами основного ОА.

Під операційним елементом розуміють пристрій, що реалізує одну з нижченаведених функцій або їх довільну комбінацію:

- зберігання слова інформації;

- виконання мікрооперацій, в результаті яких обчислюється нове значення слова інформації;

- обчислення логічної умови, яка залежить від слова інформації;

Таким чином, функція ОЕ визначена, якщо задані:

- опис слова, що зберігається або обчислюється;

- опис множини мікрооперацій, що виконуються цим елементом;

- опис обчислюваних цим елементом логічних умов.

Для побудови ОА операційні елементи з’єднуються між собою за допомогою ланцюгів передачі слів інформації від виходів одних елементів до входів інших. Залежно від виконуваних мікрооперацій ОЕ діляться на різновиди: шина, регістр, лічильник, суматор, схема порівняння, дешифратор, шифратор тощо.

Виконання роботи

Реалізуємо цифровий операційний пристрій, який являє собою арифметично-логічний блок для виконання операцій додавання, віднімання, множення та ділення двох восьмирозрядних цілих чисел без знака (див. алгоритми двійкової арифметики). Структура пристрою буде являти собою композицію операційного та керуючого автоматів. Керуючий автомат побудуємо у вигляді мікропрограмного автомата із програмованою логікою (це буде зроблено у практичній роботі № 12). В рамках даної лабораторної роботи ми спроектуємо загальну структуру операційного пристрою та схему операційного автомата.

На рис. 8.2 показано загальний вигляд схеми операційного пристрою. Тут виділено операційний автомат (підсхема SUB1), керуючий автомат (підсхема SUB3), регістри операндів B і A (ініційовані для прикладу числами $B2_{(16)}$ та $93_{(16)}$) відповідно), допоміжні буферні схеми, схема запуску операції та набір логічних констант для задання адреси початку мікропрограми виконання тієї чи іншої операції. Так, після виконання операції множення вміст регістрів B і A набуде значення 66(16) та 36(16) відповідно, оскільки $B2_{(16)}$ x $93_{(16)}$ = $6636_{(16)}$. Арифметичні операції будуть розпочинатися при натисканні кнопки „Запуск операції”. Результат буде висвітлюватися практично миттєво за рахунок високої тактової частоти роботи схеми (10 кГц). Схему операційного автомата зображено на рис. 8.3.

Операційний пристрій, який буде реалізовано в цій і наступній лабораторних роботах (рис. 8.2), має три логічні умови, в залежності від значень яких відбуваються або ігноруються умовні переходи, та ознаку безумовного переходу 11.

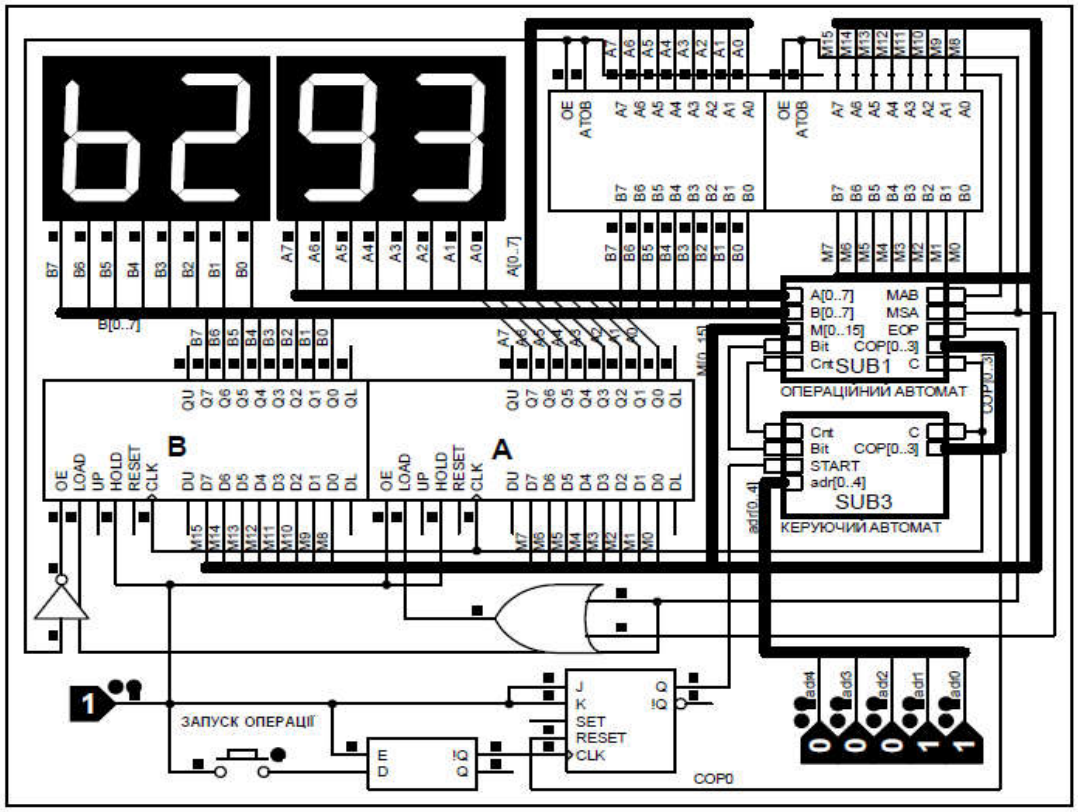


Рис. 8.2 - Загальний вигляд схеми операційного пристрою

У таблиці 8.1 наведено опис кодів логічних умов для команд умовних переходів. Ця інформація буде використана при проектуванні мікропрограмного автомата. В рамках даної лабораторної роботи нам необхідно виділити множину мікрооперацій, які б забезпечували виконання арифметичних операцій додавання, віднімання, множення та ділення двійкових цілих беззнакових чисел. Після визначення множини мікрооперацій необхідно синтезувати структуру операційного автомата. У таблиці 8.2 наведено опис кодів операцій, які подаються на дешифратор мікрооперацій в операційну схему. Таким чином, тут реалізовано вертикальне кодування операційних мікрокоманд (див. л. р. № 12).

Таблиця 8.1

Опис кодів логічних умов для команд умовних переходів мікропрограмного керуючого автомата

№	x_1x_0	Назва	Призначення
0	00	START	Перевірка натискання кнопки START для запуску операції
1	01	Bit	Аналіз значення тригера Bit
2	10	Cnt	Аналіз вмісту лічильника тактів
3	11	Логічна 1	Безумовний перехід

Таблиця 8.2

Опис кодів операцій, які подаються на дешифратор в операційному автоматі

№	$y_3y_2y_1y_0$	Назва	Призначення
0	0000	ONE	Заборонити виведення результату операції
1	0001	LAB	Завантажити в регістри A і B операнди (із зовнішніх регістрів A і B)
2	0010	L_S	Додавання до регістра-суматора вмісту регістра B
3	0011	SH_R	Зсув вправо вмісту регістра A та регістра-суматора
4	0100	R_CA	Скидання лічильника та регістра-суматора
5	0101	INC	Збільшення на одиницю (інкремент) вмісту лічильника
6	0110	OE	Дозволити виведення результату операції
7	0111	SHLA	Зсув вліво вмісту регістра A
8	1000	SH_L	Зсув вліво вмісту регістра A та регістра-суматора
9	1001	S_L	Віднімання від регістра-суматора вмісту регістра B
10	1010	W_1	Встановити в 1 тригер для запису в молодший біт A
11	1011	W_0	Скинути в 0 тригер для запису в молодший біт A
12	1100	SUB	Порівняти вміст регістра-суматора і регістра B шляхом віднімання та занести знак результату у тригер Bit
13	1101	L_B	Встановити тригер Bit відповідно до молодшого розряду регістра A
14	1110	MAB	Запис із зовнішнього регістра A у внутрішній B
15	1111	MSA	Запис із внутрішнього регістра B у зовнішній A

На рис. 8.4 наведено схему операційного автомата. Тут, окрім вищезгаданого дешифратора мікрооперацій, можна виділити: подвоєний регістр A із регістром-суматором, вхід якого підключено до 8-розрядного суматора (рис. 8.3), регістр B, лічильник тактів, тригер заборони/дозволу виведення результатів, тригер для запису логічних значень у молодший біт регістра A, тригер-прапорець Bit зберігання логічної умови для мікропрограмного керуючого автомата та ряд допоміжних логічних елементів.

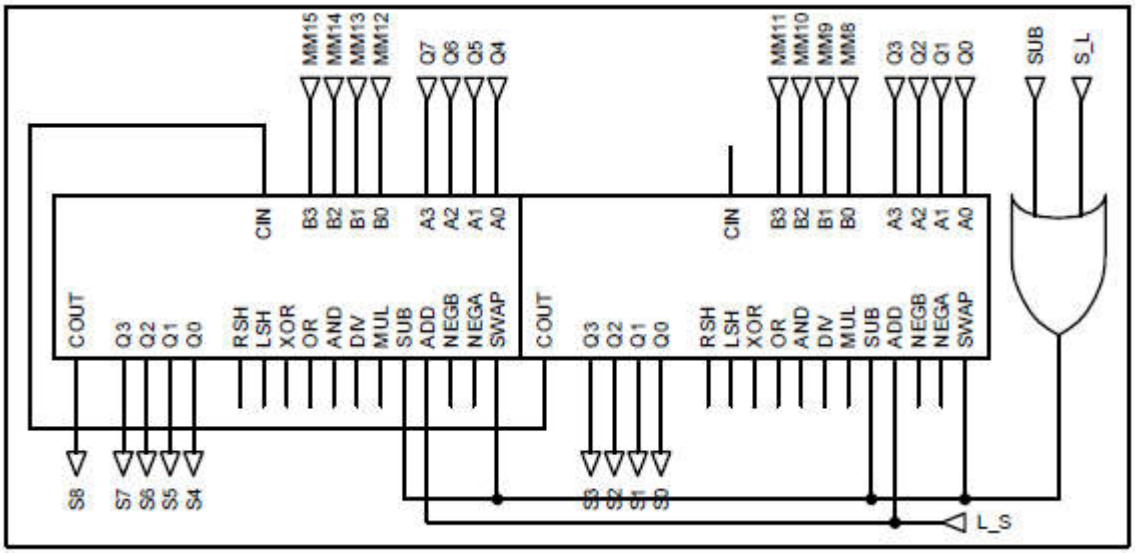


Рис. 8.3 – Вміст підсхеми SUB2 на рис. 4.3 – 8-розрядний суматор

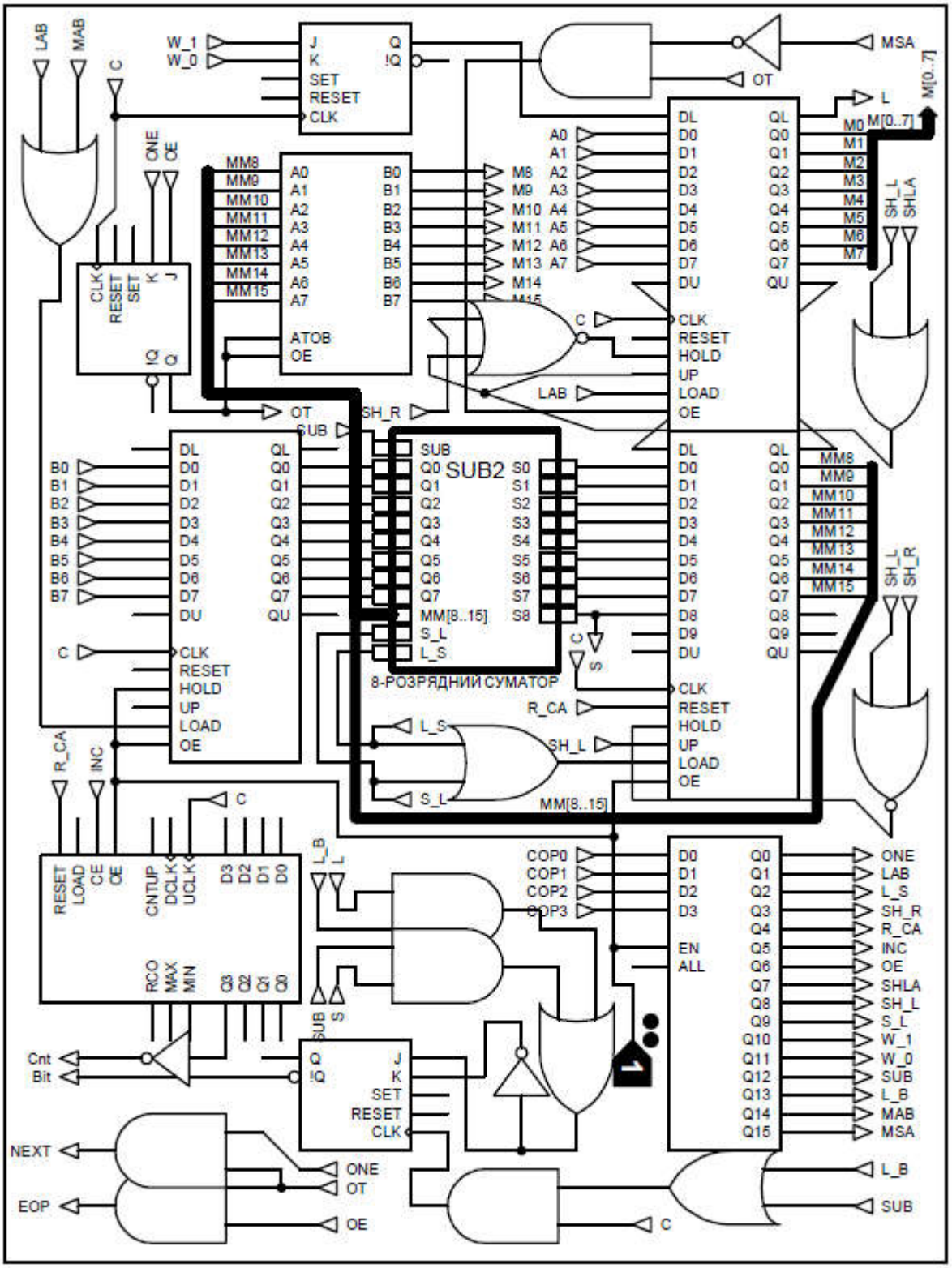


Рис. 8.4 – Схема операційного автомата

Практична робота № 9

Вивчення особливостей роботи і використання мультиплексорів

Вивчення особливостей роботи і використання мультиплексорів в задачах комп'ютерної техніки.

Основні теоретичні положення

Мультиплексори, їх використання та принципи нарощування. *Мультиплексором* (від англ. multiplex – багатократний) називається комбінаційний цифровий пристрій, призначенням якого є комутація у заданому порядку сигналів, які поступають з декількох вхідних шин, на одну вихідну. Мультиплексори набули широкого використання в обчислювальній техніці в якості комутаторів цифрових сигналів. Вони застосовуються в комп'ютерах і мікропроцесорних контролерах для комутації адресних входів динамічних оперативних запам'ятовуючих пристроїв, у вузлах об'єднання або розгалуження шин і т. д. На базі мультиплексорів можна будувати різноманітні комбінаційні пристрої з мінімальною кількістю додаткових елементів логіки. У мікропроцесорних системах керування мультиплексори встановлюють на віддалених об'єктах для забезпечення можливості передачі інформації по одній лінії від кількох встановлених на них датчиків. Слід відмітити, що мультиплексори, які виготовлені за технологією КМОН, можуть комутувати як цифрові, так і аналогові сигнали.

На рис. 9.1 приведена схема двохканального мультиплексора, що складається з елементу **АБО**, двох елементів **НІ** і двох елементів **І**.

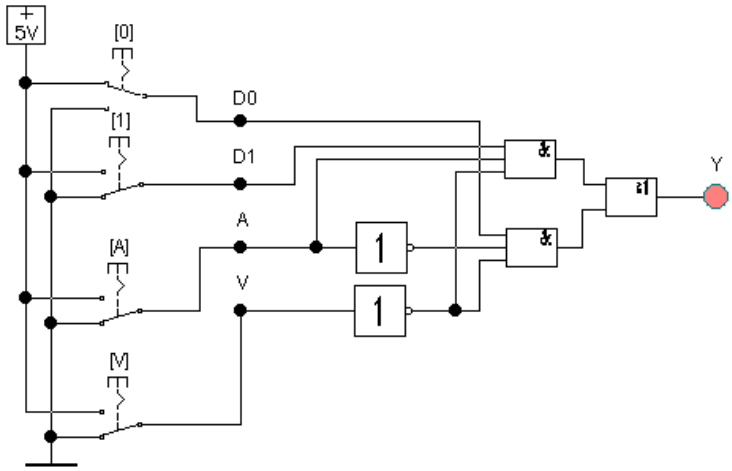


Рис. 9.1 – Двоканальний мультиплексора

Входи D0, D1 – інформаційні, на них знаходяться розряди двійкового слова, заданого у паралельному форматі. Вхід A – адресний, який дозволяє звертатись до одного з інформаційних входів і передавати його сигнал на вихід Y. Вхід V – дозволяючий.

Нижче приводиться приклад дослідження функціональних властивостей і прикладів використання мікросхеми мультиплексора, що зображена на рисунку під назвою **MUXER** (рис. 9.2**)**.

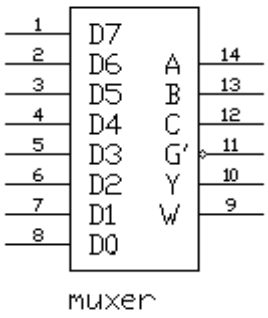


Рис. 9.2 – Мікросхема мультиплексора

Принципова схема, для проведення досліджень повинна демонструвати роботу мультиплексора при всіх можливих станах, які забезпечуються інверсним входом дозволу роботи **G** а також адресними входами **A, B, C**.

На принциповій схемі (рис. 9.3**)** проводяться дослідження статичних режимів мультиплексора для встановлення залежності між сигналами на адресних входах і відповідним бітом вхідної шини даних **D0–D7.**

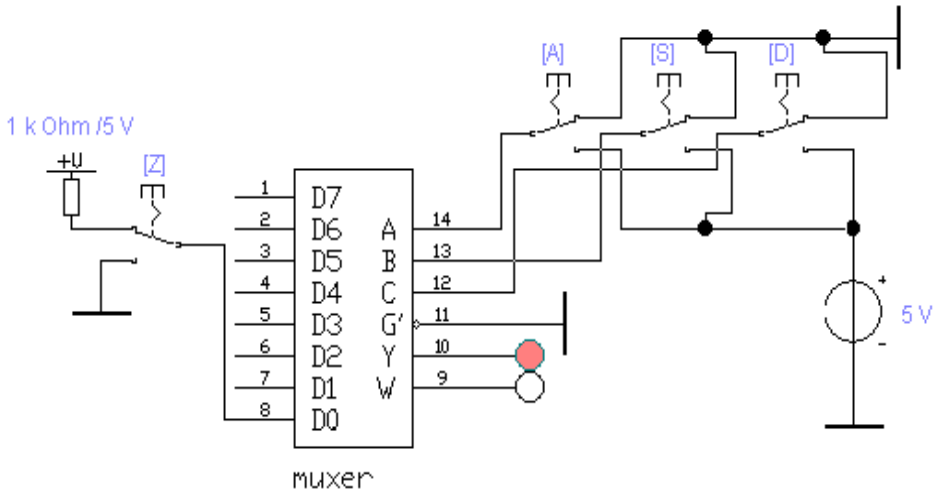


Рис. 9.3 – Дослідження статичних режимів мультиплексора

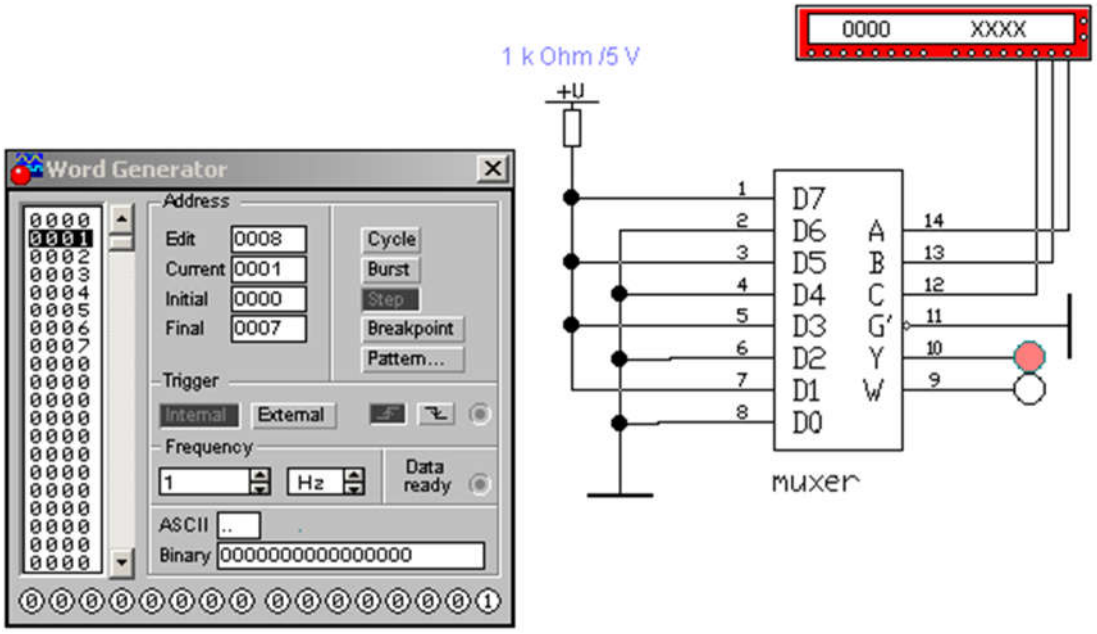


Рис. 9.4 – Word Generator

Рисунок 9.4 демонструє використання Word Generator для визначення логічної функції яка реалізується з допомогою мультиплексора. Генератор слів в режимі Pause послідовно формує адреси в двійковому форматі від 0000 до 0007, які відображаються на нижній горизонтальній строчці генератора і подаються на адресні входи мультиплексора. Логічні сигнали, що подані на біти шини даних послідовно будуть відображатись на виході Y в прямому коді, а на виході W- в зворотному. Рисунок демонструє режим, при якому адресний сигнал 0001 приєднує вхід D1 до виходу Y мультиплексора, внаслідок чого сигнал логічної 1 з його входу передається на вихід Y.

Таблиця станів мультиплексора одержана шляхом дослідження його роботи і має вигляд:

№ пп	C	B	A	Y
0	0	0	0	D_0
1	0	0	1	D_1
2	0	1	0	D_2
3	0	1	1	D_3
4	1	0	0	D_4
5	1	0	1	D_5
6	1	1	0	D_6
7	1	1	1	D_7

Логічна функція, що реалізується на попередньому рисунку за допомогою мультиплексора Y = ∑ 1, 3, 5, 7 = C’B’A +C’BA +CB’A + CBA.

У пакеті *Multisim* присутні як ідеальні пристрої, так і реальні мікросхеми. Інформацію про функціонування кожної конкретної мікросхеми можна отримати з її таблиці істинності. Для отримання більш детальної інформації про використання мікросхем можна звернутися до довідкової літератури.

Схема включення ідеального мультиплексора для дослідження приведена на рис. 9.5. Призначення виводів пристрою наступні: D7…D0 – вхідна інформаційна шина; A, B, C – входи адресації; G' – дозволяючий вхід; Y – прямий вихід, а W – інверсний. При подачі на адресні входи мультиплексора керуючого двійкового слова ABC (старший розряд C) на вихід пристрою буде переданий сигнал з того входу, номер якого відповідає десятковому еквіваленту цього слова. Керуючий сигнал G' своїм низьким вхідним рівнем дозволяє мультиплексору виконувати закладену в нього функцію, а високим – забороняє його роботу.

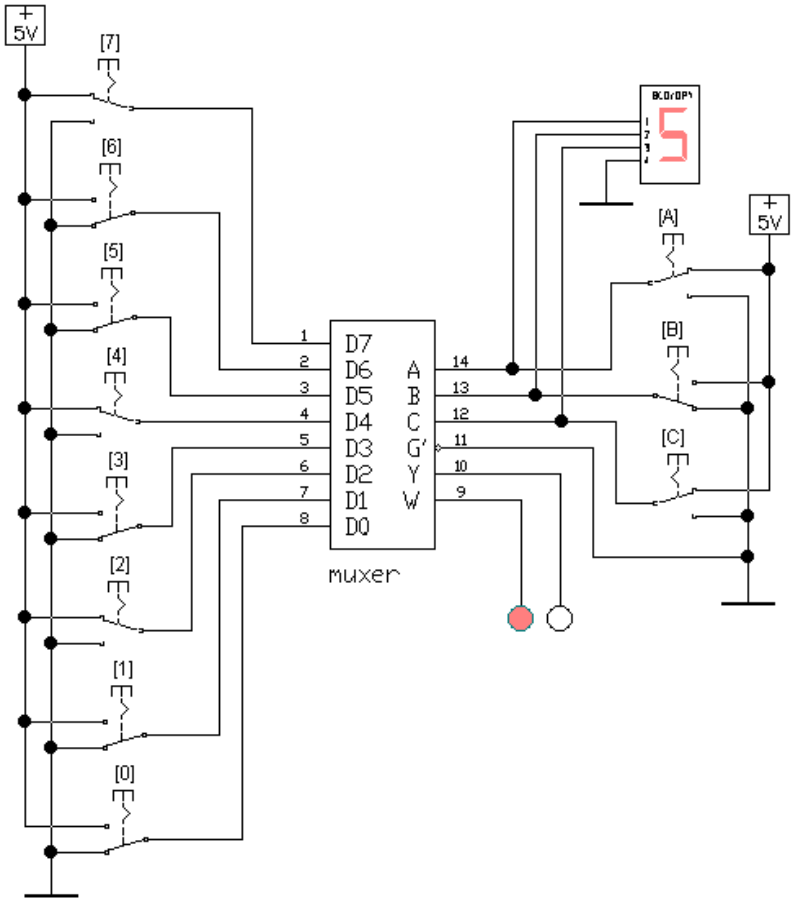


Рис. 9.5 – Схема включення ідеального мультиплексора

Завдяки наявності дозволяючого входу існує можливість синхронізації роботи мультиплексора з іншими пристроями, а також – можливість нарощування розрядності адресного простору. Приклад такого нарощування мультиплексора приведений на рис. 9.6. Вхід дозволу, об'єднаний в обох мультиплексорах через інвертор, виступає старшим розрядом D адресної шини. При D = 0 зміна сигналів на A, B, C дозволяє вибирати входи шини даних першого мультиплексора і через елемент **АБО** передавати на прямий вихід; другий мультиплексор при цьому відключений високим рівнем сигналу на вході G'. При D = 1 зміна сигналів A, B, C забезпечує комутацію входів другого мультиплексора і передачу на вихід через елемент **ВИКЛ. АБО**, у той час як відключений перший мультиплексор. При наявності в мультиплексорах інверсних виходів їх об'єднання забезпечується елементом **ВИКЛ. АБО-НІ**.

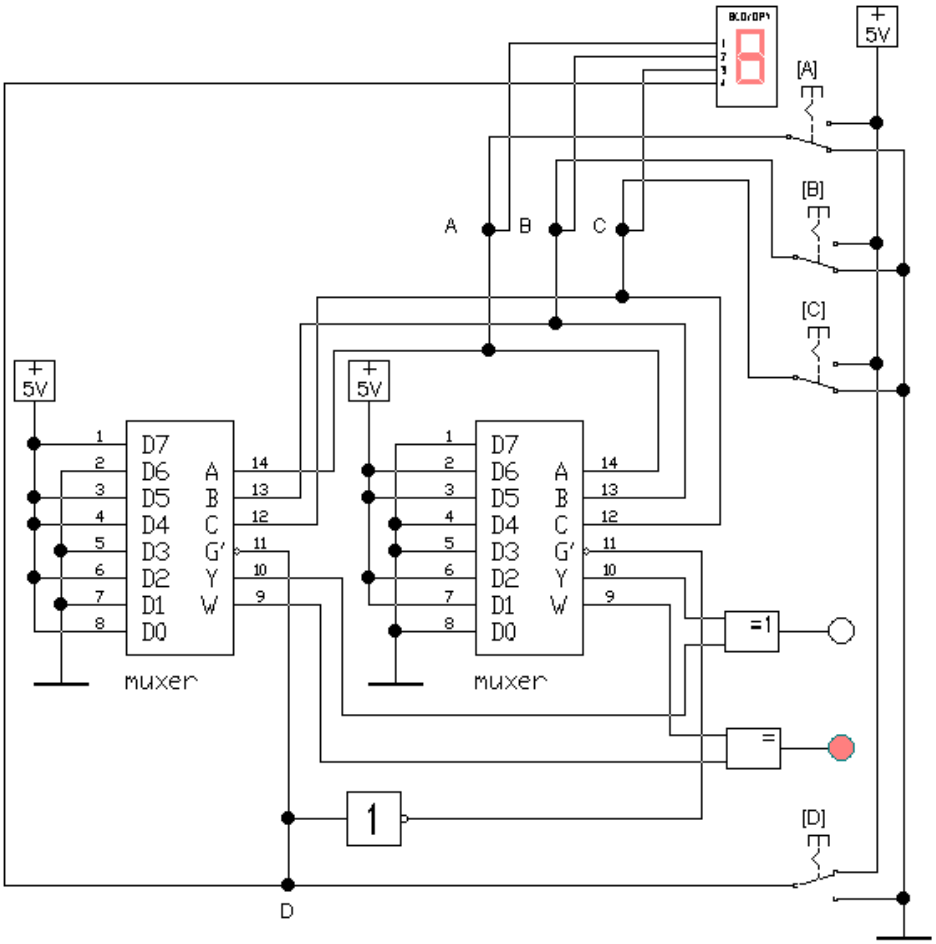


Рис. 9.6 – Нарощування мультиплексора

При необхідності суттєвого нарощування інформаційного простору використовується пірамідальний спосіб нарощування. Адресні входи

мультиплексорів нижнього рівня з’єднуються паралельно і керуються молодшими розрядами адресного простору. Їх кількість визначається тільки кількістю інформаційних входів кожного мультиплексора. Мультиплексор верхнього ступеню задає старші розряди адресного простору, завдяки яким він забезпечує комутацію виходу кожного з мультиплексорів на загальний вихід.

Виконання роботи

1. Досліджується схема реального мультиплексора.
2. Готується і досліджується схема, яка ілюструє нарощування адресного простору мультиплексорів (використовуються ідеальні мультиплексори).
3. Пропонується розробити і дослідити схему нарощування інформаційного простору мультиплексора, побудовану за пірамідальним способом (використовуються ідеальні мультиплексори).

Завдання

У звіті повинні бути приведені:

- схеми мультиплексорів, таблиці істинності та логічні вирази;
- детальний опис виводів мікросхеми, що досліджується і детальне пояснення її роботи.

Практична робота № 10

Регістри

Побудова регістрів на основі різних видів тригерів. Засвоєння алгоритмів їх функціонування та принципів використання.

Основні теоретичні положення

Регістрами називаються послідовнісні цифрові пристрої (ПЦП), які використовується для зберігання і виконання деяких логічних перетворень над вхідним словом. Вони представляють собою впорядковану послідовність тригерів, кількість яких відповідає кількості розрядів у слові, що обробляється, у сукупності з логічними елементами. Регістри можуть виконувати наступні мікрооперації: прийом слова з іншого ПЦП, передача слова в інший ПЦП, порозрядні логічні операції, зсув слова вліво або вправо на задану кількість розрядів, перетворення послідовного коду у паралельний і навпаки, установка регістра у початковий стан. Усі регістри в залежності від функціональних властивостей класифікують на дві категорії – накопичувальні (регістри пам’яті, зберігання) і зсувні. В свою чергу, зсувні регістри діляться за способом введення та виведення інформації на паралельні, послідовні та комбіновані (паралельно-послідовні та послідовно-паралельні); за напрямком передачі (зсуву) інформації – на однонаправлені та реверсивні.

Регістри є одними з найпоширеніших пристроїв цифрової техніки. При своїй порівняній простоті вони мають великі функціональні можливості, і тому застосовуються в якості керуючих і запам’ятовуючих пристроїв, генераторів, перетворювачів кодів, лічильників, дільників частоти, вузлів організації часових затримок.

Регістри пам’яті. Найпростішими з регістрів можна вважати регістри пам’яті (або паралельні регістри, регістри зберігання). Їх призначенням є зберігання двійкової інформації невеликого обсягу протягом нетривалого проміжку часу. Зміна інформації, яка зберігається у такому регістрі, відбувається після установки на вхідній шині нових цифрових даних при подачі певного рівня або фронту синхронізуючого сигналу (синхроімпульсу)

на вхід С регістра. При записі інформації у паралельний регістр усі біти вихідного слова записуються одночасно. В якості розрядних тригерів регістра пам’яті використовуються синхронізовані рівнем або фронтом тригери. Регістри пам’яті можуть бути реалізовані на D-тригерах, якщо інформація поступає на входи регістра у вигляді однофазних сигналів, і на RSC-тригерах, якщо інформація поступає у вигляді парафазних сигналів.

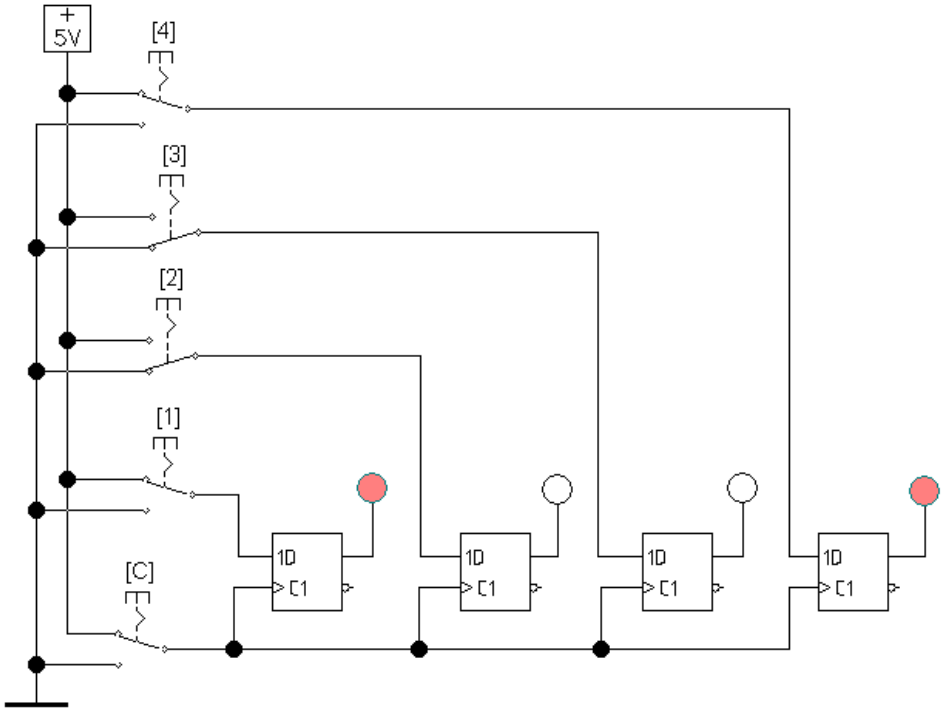


Рис. 10.1а – Чотирьохрозрядний регістр пам’яті на D-тригерах

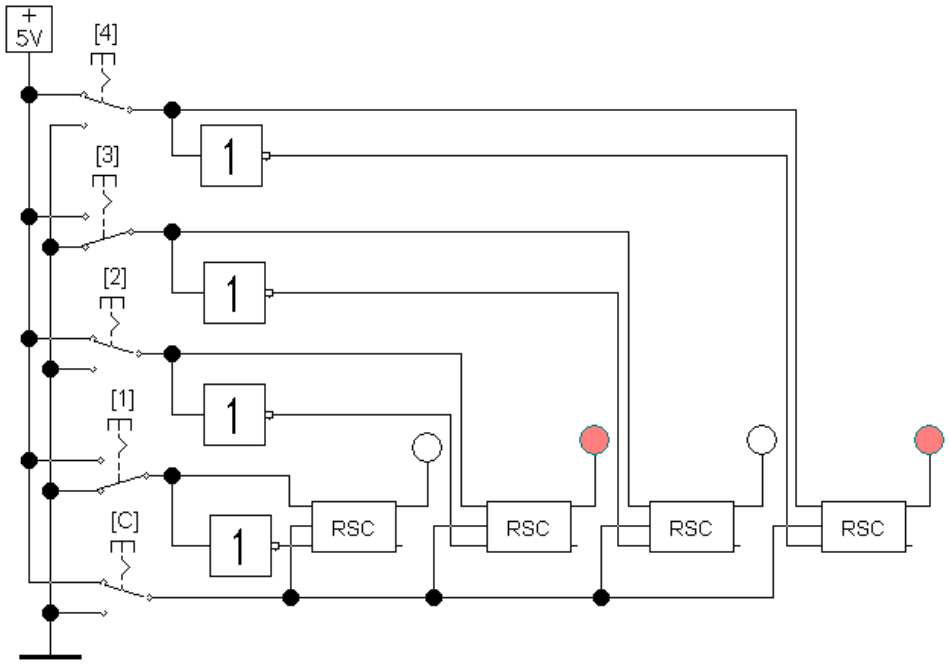


Рис. 10.1б – Чотирьохрозрядний регістр пам’яті на RSC-тригерах

На рис. 10.1 (а, б) приведені схеми чотирьохрозрядних регістрів пам’яті відповідно на D-тригерах і RSC-тригерах, синхронізованих рівнем і фронтом синхроімпульсів (часто саме чотири тригери об’єднані в одному корпусі ІМС).

Регістри пам’яті представлені реальними мікросхемами. Однією з таких є мікросхема 74173.

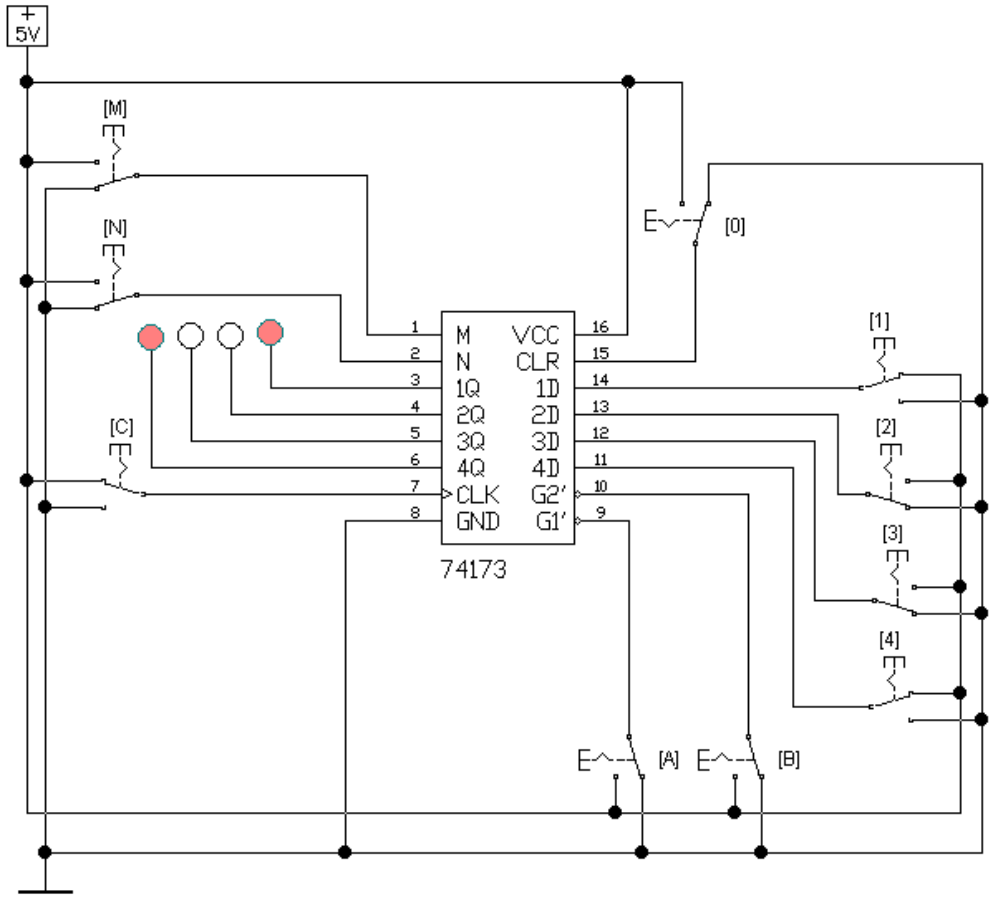


Рис. 10.2 – Мікросхема 74173

Мікросхема 74173 (аналог ІР15) – послідовний чотирьохрозрядний регістр, який здатний обслуговувати безпосередньо шину даних цифрової системи. Схема включення пристрою приведена на рис. 10.2. Мікросхема має виходи 1Q…4Q, які при високому рівні сигналів на входах G2', G1' можуть бути переведені у Z-стан. Пристрій оснащений логічними елементами дозволу запису шляхом подачі низького рівня сигналу на інверсні входи M, N (в пакеті EWB вони помилково показані прямими). Завантаження інформації в регістр відбувається синхронно з позитивним перепадом тактового імпульсу, якщо на входах M, N присутні сигналу низького рівня. Якщо на одному з цих входів наявний сигнал високого рівня, то після приходу позитивного тактового

перепаду в регістрі повинні залишатись попередні дані. Вхід очищення регістра CLR має високий активний рівень. Якщо на входи G2', G1' подані сигнали активного низького рівня, то дані, що містяться в регістрі, відображаються на виходах 1Q…4Q. Присутність хоча б одного сигналу високого рівня на входах дозволу G2', G1' викликає перехід у Z-стан (розмикання) вихідних ліній; при цьому дані з регістрів на шину даних не проходять, виходи регістра не впливають на роботу інших аналогічних виходів, під'єднаних до провідників шини. На роботу входів очищення регістра CLR і тактового C зміна рівнів сигналу на входах дозволу не впливає.

Регістри зсуву. Регістри зсуву (або послідовні регістри), окрім операції зберігання даних, здатні здійснювати перетворення інформації (наприклад, послідовного двійкового коду у паралельний і навпаки), виконувати арифметичні та логічні операції, виступати в якості елементів часової затримки. Вони можуть бути використані для побудови помножувачів і подільників двійкових чисел, адже зсув двійкового числа вліво на один розряд відповідає множенню його на **2**, а зсув вправо – діленню на **2**.

Усі регістри зсуву будуються на основі двоступінчастих тригерів або тригерів, синхронізованих фронтом синхронізуючих імпульсів. З приходом кожного тактового імпульсу відбувається перезапис (зсув) вмісту тригера кожного розряду в сусідній розряд без зміни порядку слідування одиниць і нулів. При зсуві інформації вправо після кожного тактового імпульсу біт з більш старшого розряду зсувається у молодший, а при зсуві вліво – навпаки.

Приклад схеми регістра зсуву приведений на рис. 10.3.

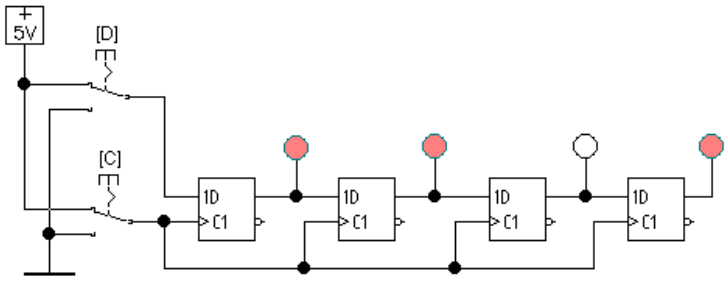


Рис. 10.3 – Схема регістра зсуву

Регістри зсуву використовуються для побудови засобів відображення інформації типу “рухомий рядок”.

Виконання роботи

1. У Multisim готується і досліджується схема регістра пам’яті на базі D-тригерів (рис. 10.1а).

2. Створюється підсхема RSC-тригера, як було показано в “Тригери”, і на її основі будується схема регістра пам’яті (рис. 10.1б). Виконується дослідження цієї схеми.

3. Досліджується один з бібліотечних регістрів пам’яті.

4. Збирається та досліджується схема регістра зсуву (рис. 10.3).

5. Досліджується один з бібліотечних регістрів зсуву.

Завдання

У звіті повинні бути приведені:

- схеми всіх досліджених регістрів;
- таблиці станів;
- короткий опис роботи схем;
- для реальних мікросхем регістрів – призначення виводів;

Практична робота № 11

Лічильники імпульсів

Вивчення принципів побудови, функціонування та використання лічильників імпульсів.

Основні теоретичні положення

Лічильниками імпульсів називають цифрові пристрої, які визначають кількість імпульсів, що поступають на їх вхід, і відображають результат підрахунку у певному коді. Лічильники імпульсів є невід'ємними вузлами мікропроцесорів, калькуляторів, електронних годинників, таймерів, частотомірів та багатьох інших пристроїв цифрової техніки. В ЕОМ лічильники використовуються для утворення послідовності адрес команд, для підрахунку кількості циклів виконання операцій і т. д.

Найпростішим однорозрядним лічильником імпульсів можна вважати JK-тригер або D-тригер, який працює у лічильному режимі (режимі Т-тригера) і рахує вхідні імпульси за **mod 2** (кожний імпульс перемикає тригер у протилежний стан). Один тригер рахує до одного, два послідовно з'єднаних тригерів рахують до трьох, n тригерів – до $(2^n - 1)$ імпульсів. Результат підрахунку формується у заданому коді, який може зберігатись у пам'яті лічильника або бути зчитаним іншим пристроєм цифрової техніки (наприклад, дешифратором для виведення числа на семисегментний індикатор). Лічильники характеризуються таким параметром, як коефіцієнт (модуль) перерахунку М, який характеризує максимальну кількість імпульсів, яка може бути подана на лічильник, щоб привести його до початкового стану.

Лічильник імпульсів, у якому при надходженні вхідного імпульсу перемикаючий перепад передається від попереднього тригера наступному, називають *лічильником з послідовним переносом*, а коли перемикаючий перепад надходить на всі розряди одночасно (чи майже одночасно) – *лічильником з паралельним переносом*.

Розглянемо двійковий лічильник з послідовним переносом. Будучи реалізований на базі лічильного тригера, перший розряд лічильника

перемикається за кожним вхідним імпульсом. Кожний наступний розряд лічильника отримує перемикаючий перепад (1/0 або 0/1) від попереднього розряду – перемикаючий перепад розповсюджується уздовж тригерної ланки лічильника послідовно.

Якщо з надходженням кожного вхідного імпульсу число в лічильнику інкрементується (збільшується на одиницю), такий лічильник називається *додаючим*. Схема додаючого чотирьохрозрядного двійкового лічильника імпульсів з послідовним переносом і коефіцієнтом рахунку $K = 2^4 = 16$ приведена на рис. 11.1а.

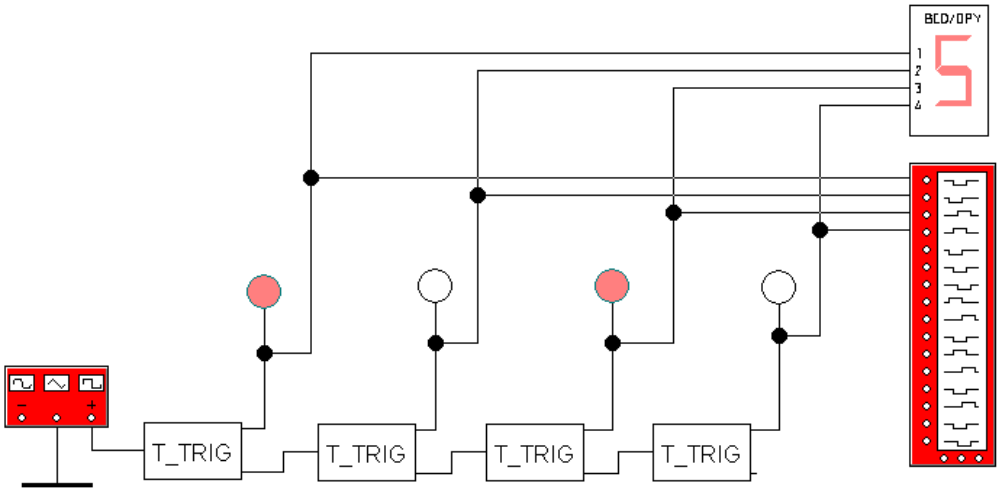


Рис. 11.1а – Схема додаючого чотирьохрозрядного двійкового лічильника

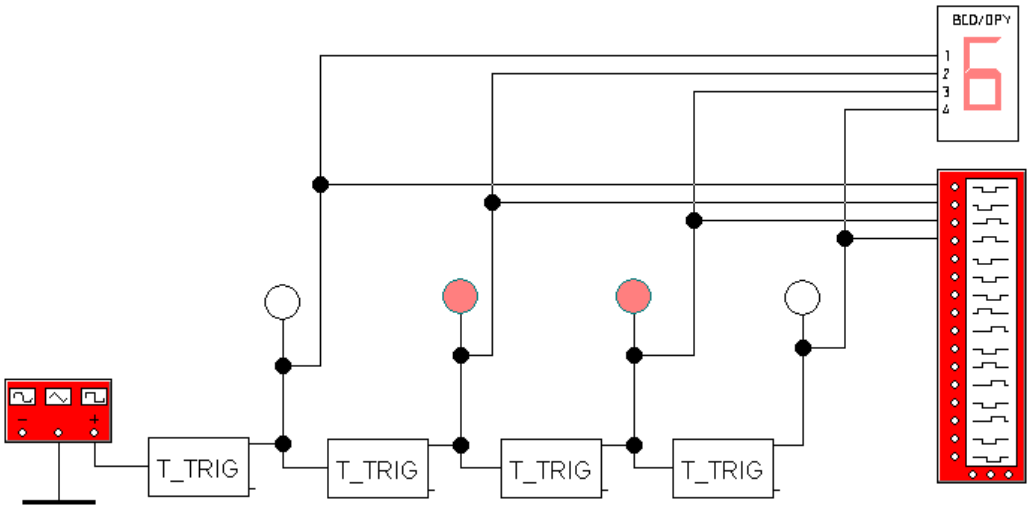


Рис. 11.1б – Схема віднімаючого чотирьохрозрядного двійкового лічильника

Якщо на лічильний вхід кожного наступного тригера подавати сигнал з прямого виходу попереднього тригера, то з надходженням кожного вхідного імпульсу число в лічильнику декрементується (зменшується на одиницю), а такий лічильник називається *віднімаючим* (рис. 11.1б).

На рис. 11.1в приведена внутрішня структура підсхеми **T_TRIG** на базі D-тригера.

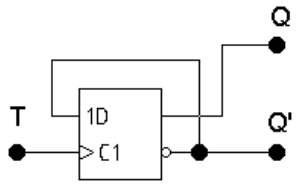


Рис. 11.1в – Структура підсхеми **T_TRIG**

Лічильники, які здатні виконувати функції додавання і віднімання в залежності від значення керуючого сигналу [M], називаються *реверсивними*. Схема такого лічильника приведена на рис. 11.2а. Внутрішня структура підсхеми **LOGIC** розкрита на рис. 11.2б, а підсхема **T_TRIG** може бути зібрана на рис. 11.2в.

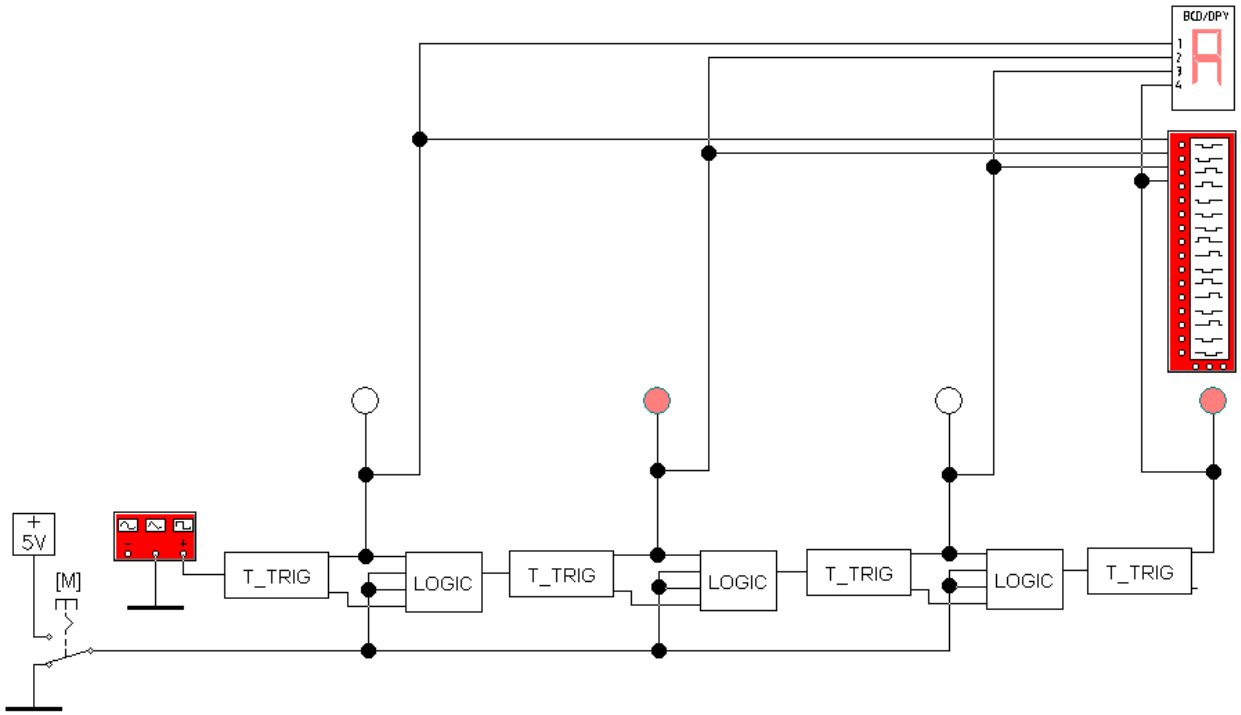


Рис. 11.2а – Схема реверсивного лічильника

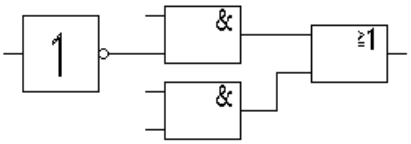


Рис. 11.2б – Внутрішня структура підсхеми LOGIC

Рис. 11.2в – Підсхема **T_TRIG**

Підсхема **LOGIC** представляє собою мультиплексор **2:1**, що забезпечує керовану комутацію одного з двох виходів попереднього Т-тригера на вхід наступного в залежності від значення керуючого сигналу [M]. Таким чином, вхід [M] задає напрямок рахунку – прямий (**лог. 1**) чи зворотний (**лог. 0**).

Реалізація лічильників з довільним коефіцієнтом рахунку. Часто при вирішенні деяких практичних задач виникає необхідність побудови лічильників з довільним модулем рахунку, що відрізняється від цілого ступеня числа **2**.

Наприклад, електронний секундомір повинен мати коефіцієнт рахунку **60**, а не **64** ($\mathbf{2^6}$). У цьому випадку застосовується метод примусового обнуління вмісту лічильнику, суть якого полягає в тому, що із загальної кількості станів виключаються ті, значення яких перевищують необхідний модуль перерахунку M = 60. Іншими словами, дорахувавши до необхідного значення, лічильник повинен бути очищений і почати відлік спочатку. Отже, для вирішення поставленої задачі спочатку необхідно спроектувати лічильник з коефіцієнтом рахунку 64 (з'єднати між собою 6 Т-тригерів), а потім за допомогою зовнішніх елементів логіки забезпечити умову його обнуління у потрібний момент. Перетворивши число 60_{10} у двійковий код, отримаємо 111100_2. Тоді, враховуючи те, що обнуління вмісту лічильника здійснюється низьким рівнем сигналу $\overline{R}$, стає зрозумілим, що, об'єднавши чотири старші розряди лічильника через елемент **І-НІ**, ми забезпечимо вказану умову. На рис. 11.3а зображена схема електронного секундоміра, а на рис. 11.3б – вміст підсхеми **TRIG**.

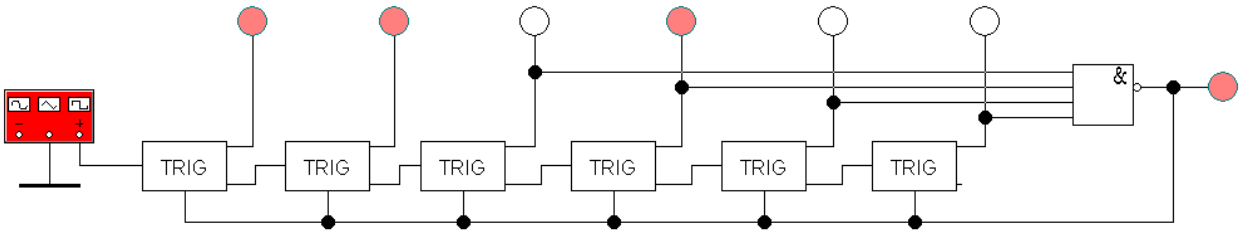


Рис. 11.3а – Схема електронного секундоміра

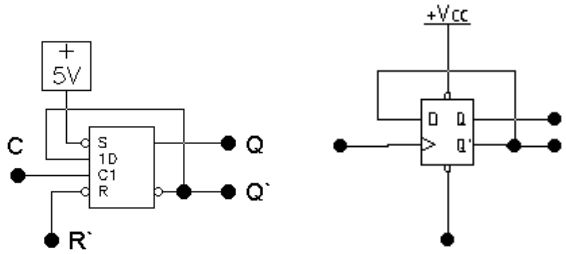


Рис. 11.3б – Вміст підсхеми TRIG.

В усіх розглянутих вище схемах підраховувані імпульси подаються на лічильник пристроєм Function Generator, який необхідно налаштувати наступним чином:

- форма сигналу – прямокутні імпульси ;
- частота (Frequency) – 1 Hz (одна зміна стану за секунду);
- скважність (Duty cycle) – 50%;
- амплітуда (Amplitude) – 5V.

У параметрах пристрою Logic Analyzer (логічний аналізатор) потрібно змінити частоту аналізу, щоб часові діаграми були синхронізовані з частотою генерації сигналу. Це можна зробити після натиску кнопки Set… підгрупи Clock, виставивши в полі вводу Internal clock rate число 1, а в найменуванні одиниць виміру вказавши Hz. У такий спосіб отримаємо частоту аналізу 1 Гц, яка відповідає частоті генерації синхроімпульсів.

Виконання роботи

1. У Multisim готуються і досліджуються схеми:

- чотирьохрозрядного двійкового лічильника імпульсів прямого рахунку (рис. 11.1а);
- чотирьохрозрядного двійкового віднімаючого лічильника імпульсів (рис. 11.1б);

– чотирьохрозрядного двійкового реверсивного лічильника імпульсів (рис. 11.2);

– лічильників з модулями рахунку 60 (повторюється схема рис. 11.3а) та згідно варіанту.

2. Досліджується бібліотечний елемент Generic 4-bit Binary Counter (ідеальний чотирьохбітний двійковий лічильник) з меню Digital/CNT (за бажанням).

3. Досліджується одна з реальних мікросхем лічильників (на вибір).

Завдання

У звіті повинні бути приведені:

– схеми усіх досліджених лічильників;

– часові діаграми;

– короткий опис схем і аналіз часових діаграм.

Таблиця варіантів

№ варіанту	Мікросхема лічильника для досліджень	Коефіцієнт перерахунку
1	4040	18
2	7469	19
3	7490	20
4	7492	21
5	7493	22
6	74160	23
7	74162	24
8	74163	25
9	74169	26
10	74190	27
11	74191	28
12	74192	29
13	74290	23
14	74293	31

Відповідність зарубіжних мікросхем лічильників імпульсів, моделі яких присутні в бібліотеці пакету EWB, і вітчизняних аналогів.

Номер по EWB	Вітчизняний аналог	Призначення (англ.)	Переклад, деталізація
Серія 74xx			
7469	—	Dual 4-bit Binary Counter	Здвоєний 4-бітний двійковий лічильник
7490	1533ИЕ2	Decade Counter	Двійково-десятковий лічильник
7492	—	Divide-By-Twelve Counter	Лічильник-дільник на 12
7493	1533ИЕ5	Dual 4-bit Binary Counter	Здвоєний 4-бітний двійковий лічильник
Серія 741xx			
74160	1533ИЕ9	Sync 4-bit Decade Counter (clr)	Синхронний 4-бітний двійково-десятковий лічильник з асинхронною установкою у стан логічного нуля
74162	1533ИЕ11	Sync 4-bit Decade Counter	Синхронний 4-бітний двійково-десятковий лічильник з синхронною установкою у стан логічного нуля
74163	1554ИЕ18	Sync 4-bit Binary Counter	Синхронний 4-бітний двійковий лічильник з синхронними передустановкою і скидом
74169	ИЕ17	Sync 4-bit Up/Down Counter	Синхронний 4-бітний реверсивний лічильник
74190	1533ИЕ12	Sync BCD Up/Down Counter	Синхронний 4-бітний синхронний реверсивний десятковий лічильник
74191	1533ИЕ13	Sync 4-bit Up/Down Counter	Синхронний 4-бітний синхронний реверсивний двійковий лічильник

74192	1533ИЕ6	Sync BCD Up/Down Counter	Синхронний 4-бітний синхронний реверсивний десятковий лічильник
Серія 742xx			
74290	—	Decade Counter	Двійково-десятковий лічильник
74293	—	4-bit Binary Counter	4-бітний двійковий лічильник
Серія 743xx			
74393	1533ИЕ19	Dual 4-bit Binary Counter	Здвоєний 4-бітний двійковий лічильник з індивідуальною синхронізацією і скидом

Серія 4xxx			
4017	1561ИЕ8 1561ИЕ9	5-stage Johnson Counter	Лічильник Джонсона з п’ятьма станами
4024	ИЕ1	7-stage Binary Counter	Двійковий лічильник з сімома станами
4040	1561ИЕ20	12-stage Binary Counter	Двійковий лічильник з дванадцятьма станами
4510		BCD Up/Down Counter	Десятковий реверсивний лічильник
4516	1561ИЕ11	Binary Up/Down Counter	Двійковий реверсивний лічильник
4518		Dual BCD Counter	Здвоєний десятковий лічильник
4520	1561ИЕ10	Dual Binary Counter	Здвоєний двійковий лічильник

Практична робота № 12

Проектування пристроїв керування на основі мікропрограмних автоматів

Особливості організації керуючих автоматів у складі операційних пристроїв.

Основні теоретичні положення

Виконання будь-якої команди в операційних пристроях, наприклад у мікропроцесорах (МП), можна розбити на два цикли: циклу вибірки команди і циклу виконання команд. У циклі вибірки команди пристрій керування (ПК) проводить зчитування коду операції і його декодування. У циклі виконання команди, відповідно до типу реалізованої команди, здійснюється визначення адрес операндів, що беруть участь в операції, безпосередньо виконання команди і збереження результатів операції.

Будь-яка команда, що виконується операційним блоком, описується деякою мікропрограмою і реалізується за певну кількість тактів, в кожному з яких виконується одна або декілька мікрооперацій. При виконанні мікропрограми на відповідні керуючі лінії операційного блока подається певним чином розподілена в часі послідовність керуючих функціональних сигналів (мікрооперацій). Порядок виконання мікрооперацій може змінюватися в залежності від ознак операції, що виробляються в арифметично-логічному пристрої (АЛП) і є вхідними сигналами ПК.

Отже, пристрій керування формує розподілену в часі і просторі послідовність зовнішніх і внутрішніх керуючих сигналів (ВКС), що забезпечують вибірку і виконання команди.

Однією з найважливіших характеристик ПК є можливість зміни послідовності керуючих слів (мікрооперацій). За цим критерієм ПК підрозділяються на ПК із «жорсткою» логікою, або спеціалізовані ПК, і на універсальні, або мікропрограмні ПК (із «гнучкою» логікою). На рис. 12.1 зображена структура ПК із «жорсткою» логікою.

Вхідною інформацією для ПК є вміст регістра команд, що визначає тип

виконуваної команди, тобто код операції (КОП), і ознаки операцій, що виробляються АЛП.

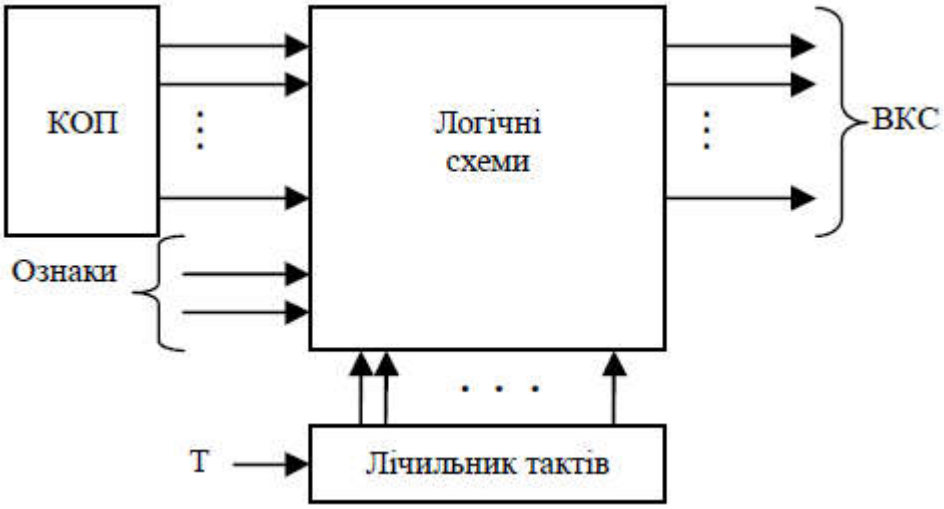


Рис. 12.1 - Пристрій керування з «жорсткою» логікою

Вихідна інформація є сукупністю керуючих сигналів, що виробляються ПК відповідно до заданої мікропрограми. Інтервал часу, що відводиться на виконання мікрооперації, називається робочим тактом або тактом операційного пристрою. Тривалість такту встановлюється по найтривалішій мікрооперації. Синхронізація ПК здійснюється за допомогою лічильника тактів, керованого зовнішнім генератором тактових імпульсів Т.

До складу ПК входять запам'ятовуючі і комбінаційні схеми, що виконують функції запам'ятовування поточного стану, визначають сукупність ВКС і формують наступний стан відповідно до вхідних ознак. Мікропрограма в такому ПК задається взаємозв'язками між елементами логічних схем, основу яких складають лічильники, регістри, дешифратори. Зміна мікропрограми у ПК призводить до задання нових взаємозв'язків між елементами, що рівносильно проектуванню нової логічної схеми ПК. Таким чином, основною властивістю ПК із «жорсткою» логікою є фіксований набір системи команд, який заданий на етапі проектування. ПК із «жорсткою» логікою використовуються в спеціалізованих і однокристальних МП.

Структурна схема мікропрограмного ПК (МПК) – пристрою керування з логікою, що зберігається в пам'яті («гнучка» логіка), показана на рис. 12.2.

До складу пристрою входять контролер послідовності мікрокоманд

(КПМК), регістр адреси мікрокоманд (РАМК), пам’ять мікрокоманд (ПМК), регістр мікрокоманди (РМК) і дешифратор мікрооперацій (ДШ МО).

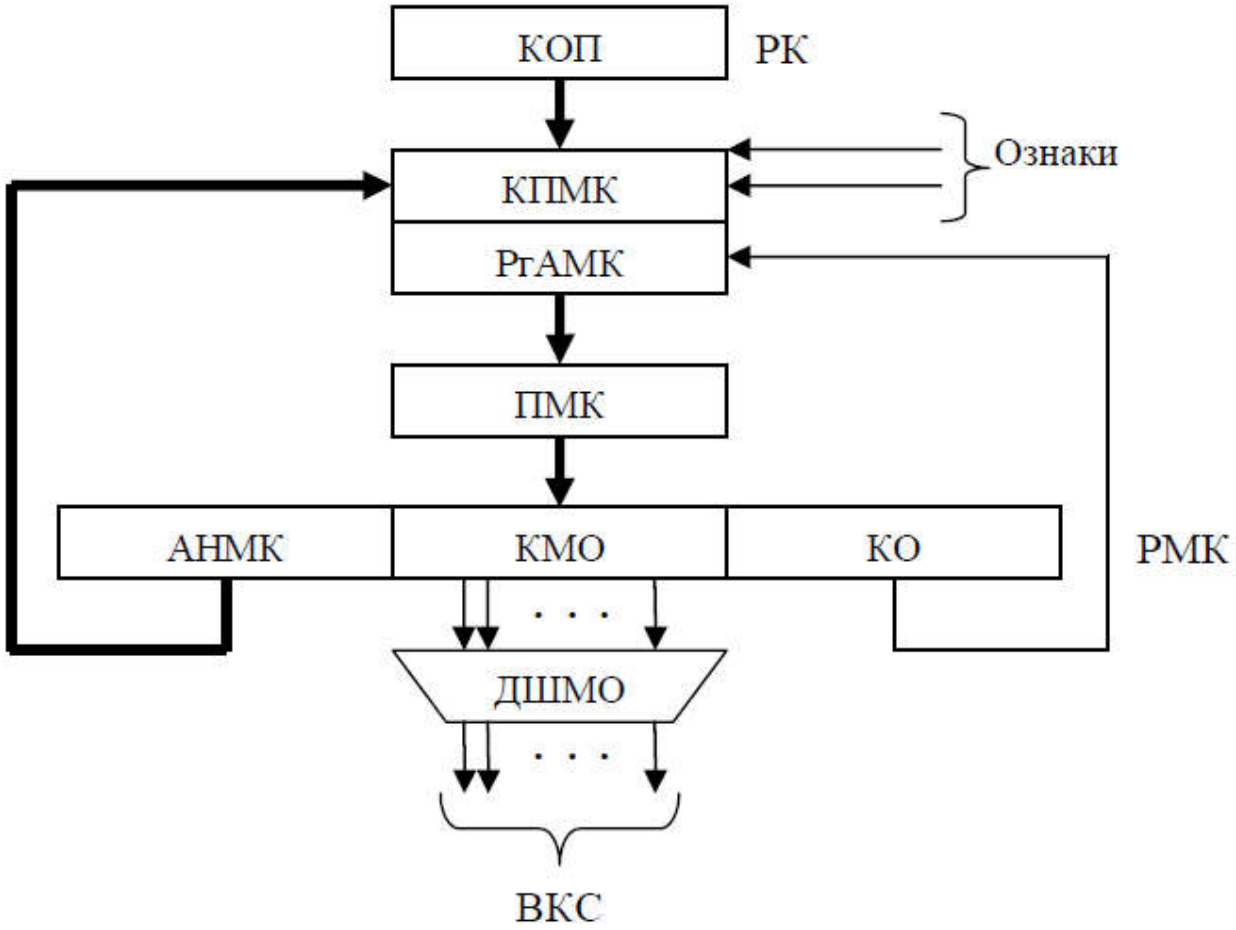


Рис. 12.2 - Мікропрограмний керуючий пристрій

Основним призначенням КПМК є реалізація алгоритмічних керуючих структур, що зустрічаються в мікропрограмах: лінійної послідовності, алгоритмічної структури типу «якщо Р, то Х, інакше Y» і алгоритмічної структури типу «поки Р, роби Х». При цьому контролер реалізує такі функції:

- проводить дешифрацію коду операції команди (КОП) для звернення до першої мікрокоманди мікропрограми, що інтерпретує дану команду;

- формує адреси наступних мікрокоманд по вказаних вище трьох керуючих структурах;

- зберігає ознаки переходів, що надходять з операційного блока та використовуються при виконанні мікрокоманд умовного переходу;

- здійснює управління перериваннями на мікропрограмному рівні.

Пам’ять мікрокоманд призначена для зберігання мікрокоманд, її

місткість і розрядність однозначно визначаються набором реалізовуваних мікропрограм. Шляхом зміни набору мікропрограм можна гнучко міняти систему команд мікропроцесора і тим самим орієнтувати його функціональну спрямованість.

Регістр мікрокоманди призначений для зберігання мікрокоманди при виконанні поточного мікрокомандного циклу. Мікрокоманда містить три основні поля: код мікрооперації КМО, адресу наступної мікрокоманди АНМК, поле коду ознак КО, в якому указується, яку ознаку розгалуження в мікропрограмі необхідно аналізувати КПМК.

Дешифратор мікрооперацій служить для декодування коду мікрооперації і формування керуючих сигналів, що ініціюють виконання відповідних мікрооперацій в операційному блоці.

Мікропрограмний пристрій керування функціонує так. КОП із регістра команд надходить на вхід КПМК. КПМК дешифрує його і на виході регістра адреси мікрокоманд контролера формується адреса першої мікрокоманди виконуваної мікропрограми. Мікрокоманда, що підлягає реалізації в поточному мікрокомандному циклі, зчитується з пам’яті у регістр мікрокоманд. ДШМО декодує код мікрооперації і формує керуючі слова, які ініціюють виконання мікрокоманди в операційному блоці. АНМК може вказуватися в мікрокоманді явно або формуватися природно, як це має місце при вибірці команд. Після виконання вибраної мікрокоманди мікрокомандний цикл повторюється.

Основними питаннями при проектуванні мікропрограмного ПК, які доводиться вирішувати з метою досягнення оптимальних параметрів ПК, є:

1) аналіз способів формування наступної адреси мікрокоманди;

2) вибір способу кодування мікрокоманд;

3) визначення суміщення в часі циклів вибірки і виконання мікрокоманд;

4) вибір типу синхронізації при формуванні мікрооперації.

Спосіб адресації мікрокоманд визначає правило формування адреси наступної мікрокоманди. У мікропрограмних автоматах використовується два

основні способи адресації: примусовий і природний методи.

Примусова адресація зводиться до вказівки в кожній мікрокоманді адреси наступної мікрокоманди, а при природній адресації – адреса наступної мікрокоманди утворюється шляхом приросту адреси попередньої, як це має місце при формуванні наступної команди. Природна адресація дозволяє за рахунок виключення поля адреси з операційних мікрокоманд зменшити розрядність пам’яті.

Щодо способів кодування мікрокоманд у мікропрограмних автоматах, то можна виділити такі три способи: горизонтальне, вертикальне та змішане кодування.

Горизонтальний спосіб кодування є простим варіантом кодування, при якому кожен розряд поля коду мікрооперації однозначно визначає керуючий сигнал для виконання мікрооперації. У випадку великого набору мікрооперацій (10 – 100) горизонтальне кодування може вимагати великої розрядності пам’яті мікрокоманд. З іншого боку, за рахунок обмежень на сумісність мікрооперацій у більшості мікрокоманд лише невелика кількість розрядів у полі КМО міститиме одиниці, в основному ж мікрокоманда складатиметься з нулів. Це призводить до неефективного використання пам’яті мікропрограм.

Перевагою горизонтального кодування є можливість паралельної роботи декількох операційних блоків, що дозволяє істотно підвищити швидкодію і зумовлює високий ступінь завантаження устаткування, а також простоту формування ВКС.

При вертикальному кодуванні мікрооперація визначається не станом одного з розрядів мікрокоманди, а двійковим кодом, що міститься в операційній частині мікрокоманди. Кількість розрядів операційної частини мікрокоманди визначається як: m=]log2 n[. Звідси видно, що основною перевагою є невелика довжина мікрокоманди, що спричинює скорочення ємності ПМК. Проте в цьому випадку потрібні складні дешифратори на велику кількість мікрооперацій, збільшуються часові витрати на дешифрацію, а

головне – в кожній мікрокоманді вказується лише одна мікрооперація, що призводить до збільшення довжини мікропрограм у порівнянні з горизонтальним кодуванням.

Розвитком способів кодування мікрокоманд з метою усунення основних недоліків, властивих горизонтальному і вертикальному способам, є горизонтально-вертикальне, або змішане, кодування мікрокоманд.

Виконання роботи

Реалізуємо керуючий автомат цифрового операційного пристрою (арифметично-логічного). Керуючий автомат побудуємо у вигляді мікропрограмного автомата із програмованою логікою (природна адресація мікрокоманд та вертикальне кодування мікрооперацій).

Для цього необхідно, перш за все, визначити формати мікрокоманд. Оскільки ми проектуємо мікропрограмний автомат із природною адресацією, то слід виділити операційні та адресні мікрокоманди. Для реалізації пристрою оберемо 8-розрядну мікросхему пам’яті мікрокоманд. Формати мікрокоманд зображено на рис. 12.3, а схему керуючого автомата – на рис. 12.4.

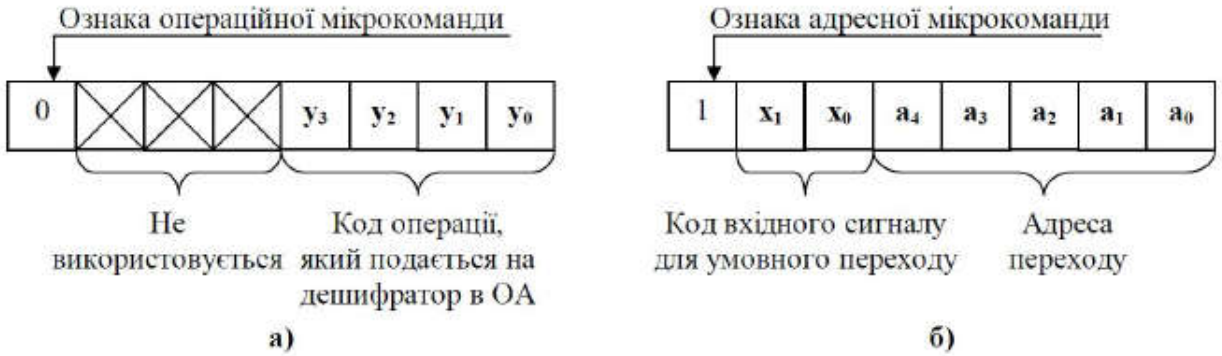


Рис. 12.3. Формати мікрокоманд: а – операційна мікрокоманда; б – адресна мікрокоманда

Правильність роботи спроектованої нами схеми цифрового операційного пристрою перевіримо на прикладі виконання операцій множення та віднімання двох цілих двійкових чисел. Насамперед складемо блок-схему алгоритму множення (рис. 12.5). Використовуючи опис кодів логічних умов та мікрооперацій, наведені у таблицях 12.1 і 12.2, ми можемо подати алгоритм множення у вигляді мікропрограми для керуючого автомата,

доповнивши її мікропрограмою віднімання.

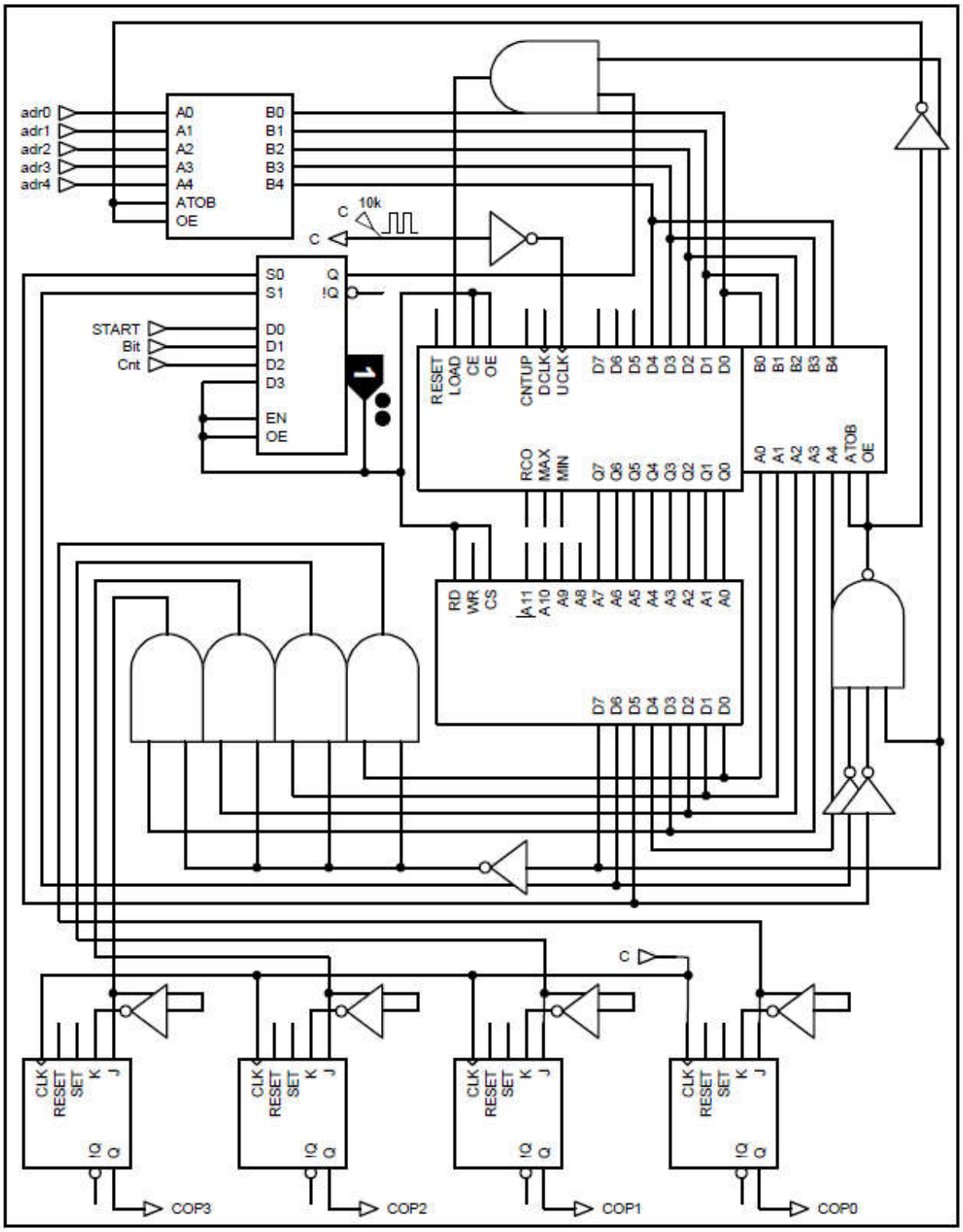


Рис. 12.4 - Схема керуючого мікропрограмного автомата

У таблиці 12.3 показано вміст пам’яті мікрокоманд (мікропрограма) у двійковому форматі подання кодів мікрокоманд. Дана мікропрограма керує всіма процесами, які відбуваються у схемі цифрового операційного пристрою,

тобто визначає логіку його роботи.

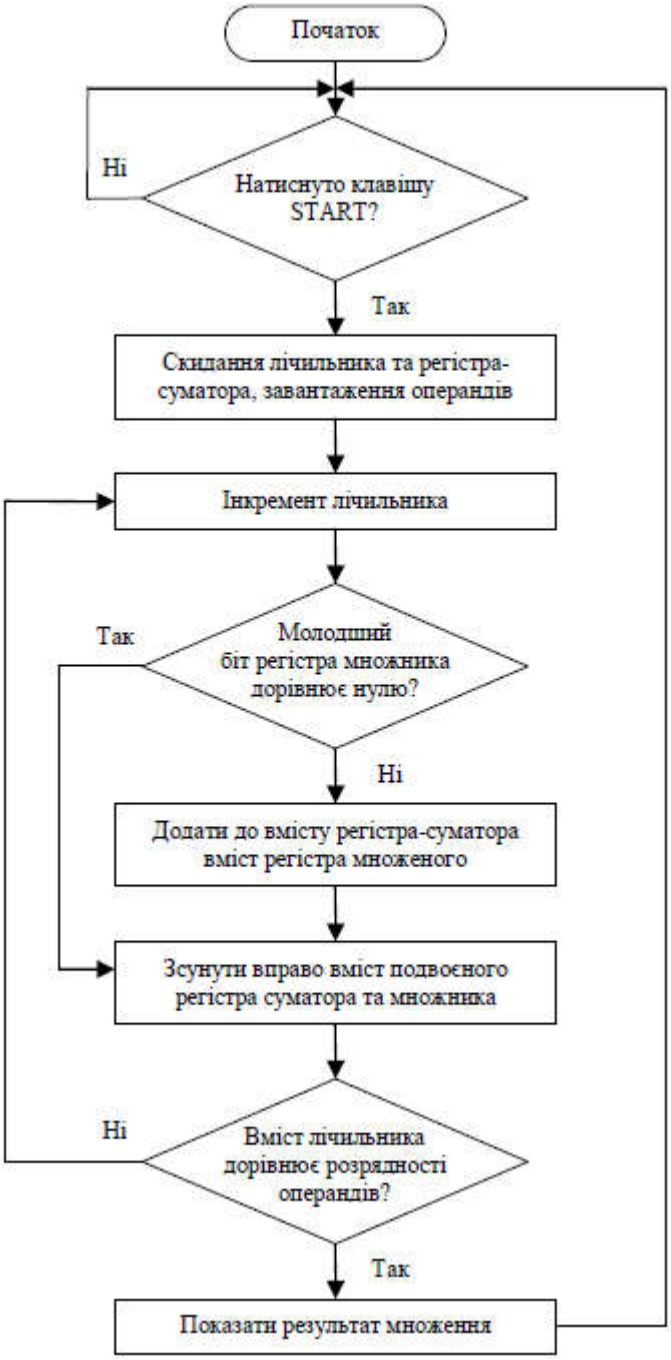


Рис. 12.5 - Блок-схема алгоритму роботи пристрою множення

Таблиця 12.1

Опис кодів логічних умов для команд умовних переходів мікропрограмного керуючого автомата

№	x_1x_0	Назва	Призначення
0	00	START	Перевірка натискання кнопки START для запуску операції
1	01	Bit	Аналіз значення тригера Bit
2	10	Cnt	Аналіз вмісту лічильника тактів
3	11	Логічна 1	Безумовний перехід

Таблиця 12.2

Опис кодів операцій, які подаються на дешифратор в операційному автоматі

№	$y_3y_2y_1y_0$	Назва	Призначення
0	0000	ONE	Заборонити виведення результату операції
1	0001	LAB	Завантажити в регістри A і B операнди (із зовнішніх регістрів A і B)
2	0010	L_S	Додавання до регістра-суматора вмісту регістра B
3	0011	SH_R	Зсув вправо вмісту регістра A та регістра-суматора
4	0100	R_CA	Скидання лічильника та регістра-суматора
5	0101	INC	Збільшення на одиницю (інкремент) вмісту лічильника
6	0110	OE	Дозволити виведення результату операції
7	0111	SHLA	Зсув вліво вмісту регістра A
8	1000	SH_L	Зсув вліво вмісту регістра A та регістра-суматора
9	1001	S_L	Віднімання від регістра-суматора вмісту регістра B
10	1010	W_1	Встановити в 1 тригер для запису в молодший біт A
11	1011	W_0	Скинути в 0 тригер для запису в молодший біт A
12	1100	SUB	Порівняти вміст регістра-суматора і регістра B шляхом віднімання та занести знак результату у тригер Bit
13	1101	L_B	Встановити тригер Bit відповідно до молодшого розряду регістра A
14	1110	MAB	Запис із зовнішнього регістра A у внутрішній B
15	1111	MSA	Запис із внутрішнього регістра B у зовнішній A

Таблиця 12.3

Мікропрограми операцій множення (3 – 13) та віднімання A=B–A (14 – 22) для мікропрограмного керуючого автомата

№	Код мікро-команди	Призначення мікрокоманди
0	00000000	Немає операції. Заборонити виведення результату операції
1	10000000	Завантаження адреси переходу, якщо натиснуто кнопку START
2	11100000	Безумовний перехід на адресу 0
3	00000001	Завантажити в регістри A і B операнди
4	00000100	Скидання лічильника та регістра-суматора
5	00000101	Збільшення на одиницю (інкремент) вмісту лічильника
6	00001101	Встановити Bit відповідно до молодшого розряду регістра A
7	10101001	Перехід на адресу 9, якщо мол. біт рег. множника дорівнює 0
8	00000010	Додавання до регістра-суматора вміст регістра множеного B
9	00000011	Зсув вправо вмісту регістра A та регістра-суматора
10	11000101	Перехід на адресу 5, якщо вміст лічильника не дорівнює 8
11	00000110	Дозволити виведення результату множення
12	00000110	Вивести результат множення
13	11100000	Безумовний перехід на адресу 0
14	00000000	Немає операції. Заборонити виведення результату операції
15	00000001	Завантажити в регістри A і B операнди
16	00000100	Скидання регістра-суматора
17	00000010	Додавання до регістра-суматора вміст регістра B
18	00001110	Запис операнда із зовнішнього регістра A у внутрішній B
19	00001001	Віднімання від регістра-суматора вміст регістра B
20	00000110	Дозволити виведення результату операції віднімання
21	00001111	Запис із внутрішнього регістра B у зовнішній A
22	11100000	Безумовний перехід на адресу 0

Обирати, яка саме операція буде виконана (множення чи віднімання), слід набором логічних констант для задання адреси початку мікропрограми adr[0..4] (див. рис. 12.6).

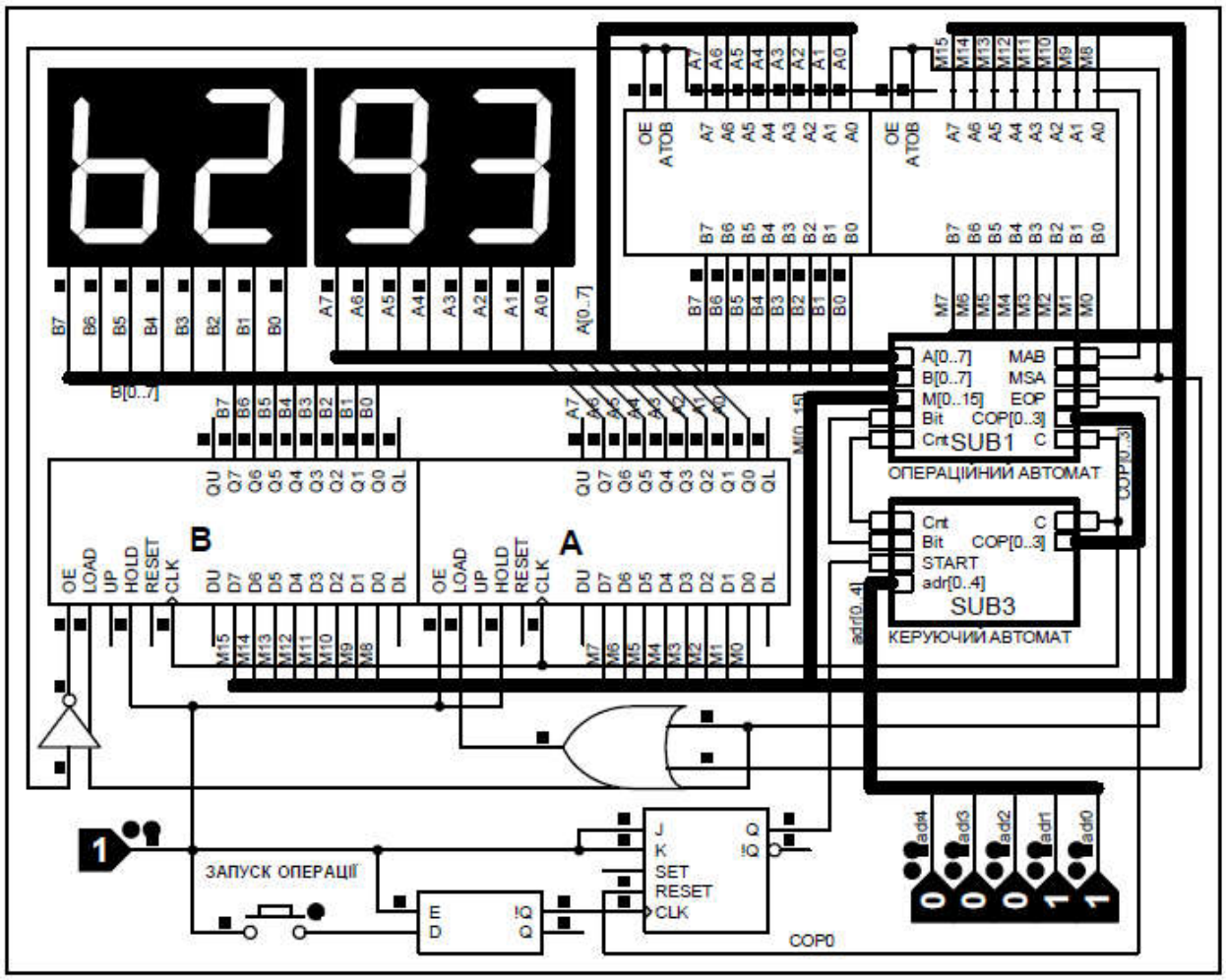


Рис. 12.6 – Загальний вигляд схеми операційного пристрою

При встановленому значенні 3 (0011(2)) буде виконуватись мікропрограма множення, а при встановленні значення 14 (1110(2)) – операція віднімання.

Практична робота № 13

Проектування елементарної мікропроцесорної системи нейманівської архітектури

Принципи побудови та функціонування комп'ютерів

Основні теоретичні положення

Основні принципи побудови комп'ютерів виклали в 1946 р. американські математики Дж. фон Нейман, К. Голдстайн і А. Беркс. Сукупність цих принципів породила класичну нейманівську архітектуру, яка зберігає актуальність і сьогодні.

Загалом нейманівська архітектура має такі основні ознаки:

- наявність одного обчислювача, що має процесор, пам'ять, засоби введення-виведення інформації та керування;

- використання двійкової системи числення як для подання інформації, так і для виконання арифметико-логічних операцій;

- розміщення в єдиній спільній пам'яті команд і чисел фіксованої довжини;

- лінійну структуру адресації комірок пам'яті, що вимагає наявності в процесорі лічильника команд;

- централізоване автоматичне послідовне зчитування команд із пам'яті та інтерпретацію їх процесором; дані обробляються паралельно – одночасно над усіма розрядами машинного слова; - низький рівень машинної мови.

Комп'ютер класичної архітектури вміщує (рис. 13.1):

- арифметико-логічний пристрій (АЛП); - оперативну пам'ять (ОП);

- засоби зберігання і введення-виведення інформації: зовнішні запам'ятовуючі пристрої (ЗЗП); пристрої введення інформації (ПВв); пристрої виведення інформації (ПВив); усі ці пристрої називають зовнішніми або периферійними (ПП);

- пристрій керування (ПК). Разом з АЛП він утворює процесор. При наявності в машині декількох процесорів виділяють центральний (ЦП).

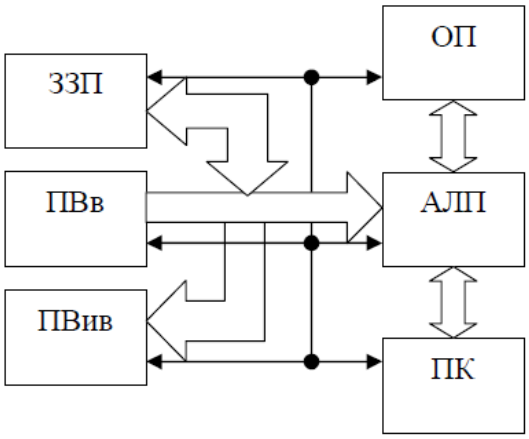


Рис. 13.1 – Спрощена структура комп’ютера

Арифметично-логічний пристрій призначений для виконання арифметичних і логічних операцій, передбачених системою команд даного комп’ютера. До складу АЛП входять регістри і комбінаційні схеми. Дані для обробки в АЛП надходять з ОП і називаються операндами. Результати операцій пересилаються в ОП або тимчасово зберігаються в регістрах.

Пристрій керування зчитує і дешифрує у відповідній послідовності команди, формує й подає керуючі сигнали для інших пристроїв комп’ютера.

Оперативна пам’ять призначена для тимчасового зберігання програм і даних, в ній виконуються операції записування і читання інформації. Крім ОП, використовують також постійну пам’ять, в якій здійснюються тільки операції читання. Оперативну (ОЗП) і постійну пам’ять (ПЗП) та регістри АЛП називають внутрішньою пам’яттю. Процесор і ОП разом створюють ядро комп’ютера.

Операції введення-виведення – це обмін інформацією між ядром машини і ПП. Операція введення передає інформацію з ПП в ядро комп’ютера, а операція виведення – навпаки.

Зовнішні запам’ятовуючі пристрої призначені для тривалого і енергонезалежного зберігання великих об’ємів інформації. Фізично її реалізують у вигляді накопичувачів.

Пристрої введення і виведення інформації (ПВВ) розглядають як єдину функціональну частину комп’ютера. Різні за своїми функціями, принципами побудови та характеристиками ПВВ і ЗЗП разом створюють групу дуже

різноманітних зовнішніх або периферійних пристроїв.

Зв’язок між функціональними частинами машини здійснюють за допомогою інтерфейсу – сукупності шин, сигналів, допоміжних мікросхем та алгоритмів, призначених для обміну інформацією між пристроями комп’ютера. Виділяють три шини (рис. 13.2):

- адреси (ША), призначена для передачі адреси комірок ОП і регістрів ПП;

- даних (ШД), призначена для передачі даних;

- керування (ШК), призначена для передачі керуючих сигналів від процесора до пристроїв і навпаки.

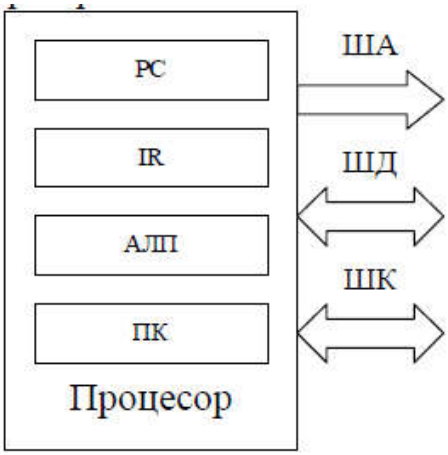


Рис. 13.2 - Спрощена структура процесора

Принцип програмного керування. У комп’ютері реалізують принцип програмного керування, суть якого така. Для розв’язання кожної задачі розробляють алгоритм на основі числових методів обчислення. Алгоритм переводиться на мову, властиву даній машині, у вигляді програми – мовної конструкції, яка є впорядкованою послідовністю описів і команд, призначених для обробки інформації. Кожна команда визначає дії комп’ютера щодо виконання будь-якої операції, які реалізують апаратні (технічні) і програмні засоби. Програма записується в ОП у вигляді машинних слів, які кодуються цифрами 0 і 1 та розрізняються тільки способом використання. Код операції надходить у регістр команд IR (instruction register) і потім дешифрується, а дані – в регістри АЛП (рис. 13.2).

Команди програми розміщені в ОП лінійно (одна за одною) і

виконуються послідовно. Номер команди в ОП визначається програмним лічильником PC (program counter), який іноді називають вказівником команд IP (instruction pointer). Пристрій керування (ПК) виробляє множину керуючих сигналів, які подаються на всі пристрої машини. Регістр команд, програмний лічильник і керуючий автомат входять до складу ПК. Послідовне керування зумовлене наявністю одного процесора. Команди умовного й безумовного розгалуження змінюють лінійний порядок зчитування і виконання команд.

Множина всіх операцій, що реалізуються в комп'ютері, складає його операційні ресурси. Комп'ютери, операційні ресурси яких забезпечують виконання будь-якого алгоритму обробки інформації, називають універсальними. Для цього теоретично достатньо мати в операційних ресурсах тільки чотири операції: пересилку слова між будь-якими комірками ОП, додавання (віднімання) одиниці до слова, умовний перехід за збігом слів та безумовну зупинку комп'ютера. Проте в комп'ютерах операційні ресурси складаються з десятків і сотень команд, що спрощує програмування.

Загалом у комп'ютерах використовують список команд, що забезпечує виконання таких груп операцій:

- пересилки даних між регістрами АЛП, регістрами і ОП;

- арифметичних операцій над двійковими числами із фіксованою та плаваючою комою: додавання, віднімання, знакового і беззнакового множення і ділення;

- логічних операцій заперечення, диз'юнкції, кон'юнкції, додавання за модулем два;

- установлення відношень – більше, менше, нерівно, більше-рівно та ін.;

- зсуву вліво чи вправо – арифметичного, логічного, циклічного;

- керування програмою: умовними та безумовними переходами та викликами процедур, безумовними та умовними поверненнями з процедур, перериванням програми; деякі комп'ютери мають спеціальні команди для організації циклів; - введення-виведення даних між ядром машини і ПП;

- спеціальних операцій для машин із співпроцесорами (математичними

розширювачами): обчислення квадратного кореня, синуса, косинуса, логарифмічні та ін.;

- перетворення з одного формату в інший (наприклад, із 8- бітного в 16-бітний або в двійково-десятковий формат);

- системних операцій – завантаження службових регістрів, захист пам’яті;

- мультимедійних операцій для виконання дій зі звуком, графікою, зображенням.

Зі зростанням продуктивності процесора збільшується і кількість команд.

Виконання роботи

Спроектуємо елементарну 8-розрядну мікропроцесорну систему (МПС), яка буде містити арифметично-логічний пристрій, розроблений у практичних роботах № 8, 12. У своєму складі МПС буде містити:

1. Оперативний запам’ятовуючий пристрій, в якому буде розміщена деяка програма.

2. Мікропроцесорну частину, яка буде складатися із:

а) програмного лічильника;

б) регістра коду команди;

в) блока синхронізації;

г) блока дешифрації коду команди;

д) блока керування виконанням команди;

е) арифметично-логічного блока з регістрами A і B;

ж) блоку додаткових регістрів C і D для лічби та індексації.

3. Набір семисегментних світлодіодних індикаторів, приєднаних до регістрів для візуалізації процесу роботи МПС.

Система буде мати можливість виконання таких команд: NOP (0); MOV register, data (1); MOV register, [address] (2); MOV [address], register (2); MOV register, [address + D] (3); MOV [address + D], register (3); INC register (4); DEC register (5); ADD register1, register2 (6); SUB register1, register2 (7); MUL (8);

DIV (9); JMP address (10); JZ address (11); JS address (12).

Для простоти побудови схеми МПС та незмінності структури АЛП, розробленої у попередніх лабораторних роботах, накладемо деякі обмеження на функціональні можливості команд:

1) команди пересилки MOV працюють зі всіма регістрами;

2) команди INC, DEC працюють лише з регістрами C і D;

3) арифметичні команди працюють лише з регістрами A і B;

4) команда JZ аналізує лише вміст регістра лічби C;

5) команда JS аналізує лише біт знака регістра-акумулятора A.

Відзначимо, що команди в нашому випадку можуть бути як однобайтні, наприклад ADD A, B або INC A, так і двобайтні, наприклад JMP address або MOV A, data, у яких другий байт команди – це або адреса, або деяке числове значення.

Оберемо такий формат коду команди (рис 13.3): молодші 4 розряди (0 – 3) міститимуть код операції (номери, наведені у круглих дужках системи команд), розряд 4 визначатиме напрямок передачі даних (наприклад, 0 – зчитування, 1 – запис), розряди 5 – 6 визначатимуть регістр (наприклад, 00 – A, 01 – D, 10 – C, 11 – B), старший розряд 7 визначатиме довжину команди (наприклад, 0 – однобайтна, 1 – двобайтна команда).

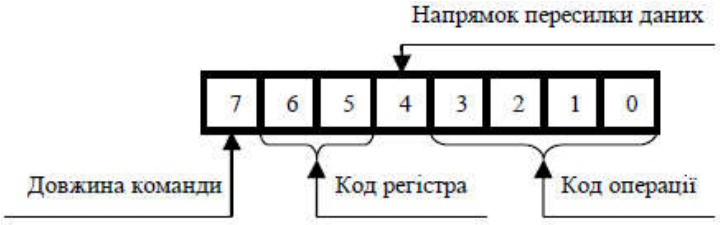


Рис. 13.3 - Формат коду команди

Для аналізу працездатності спроектованої мікропроцесорної системи складемо деяку програму (використовуючи набір команд МПС). Для цього складену програму, закодовану відповідно до таблиці кодування команд, буде занесено в оперативний запам’ятовуючий пристрій. Іншими словами, компіляцію програми виконаємо вручну.

На рис. 13.4 зображено загальну схему проектованої мікропроцесорної

системи. Блок керування мікропроцесора (підсхема SUB4, рис. 13.5) зчитує та виконує команди програми, що міститься в оперативному запам'ятовуючому пристрої (справа від SUB4). Стан мікропроцесора (вміст регістрів загального призначення) висвітлюється індикаторами, що дозволяє візуально спостерігати за роботою мікропроцесорної системи.

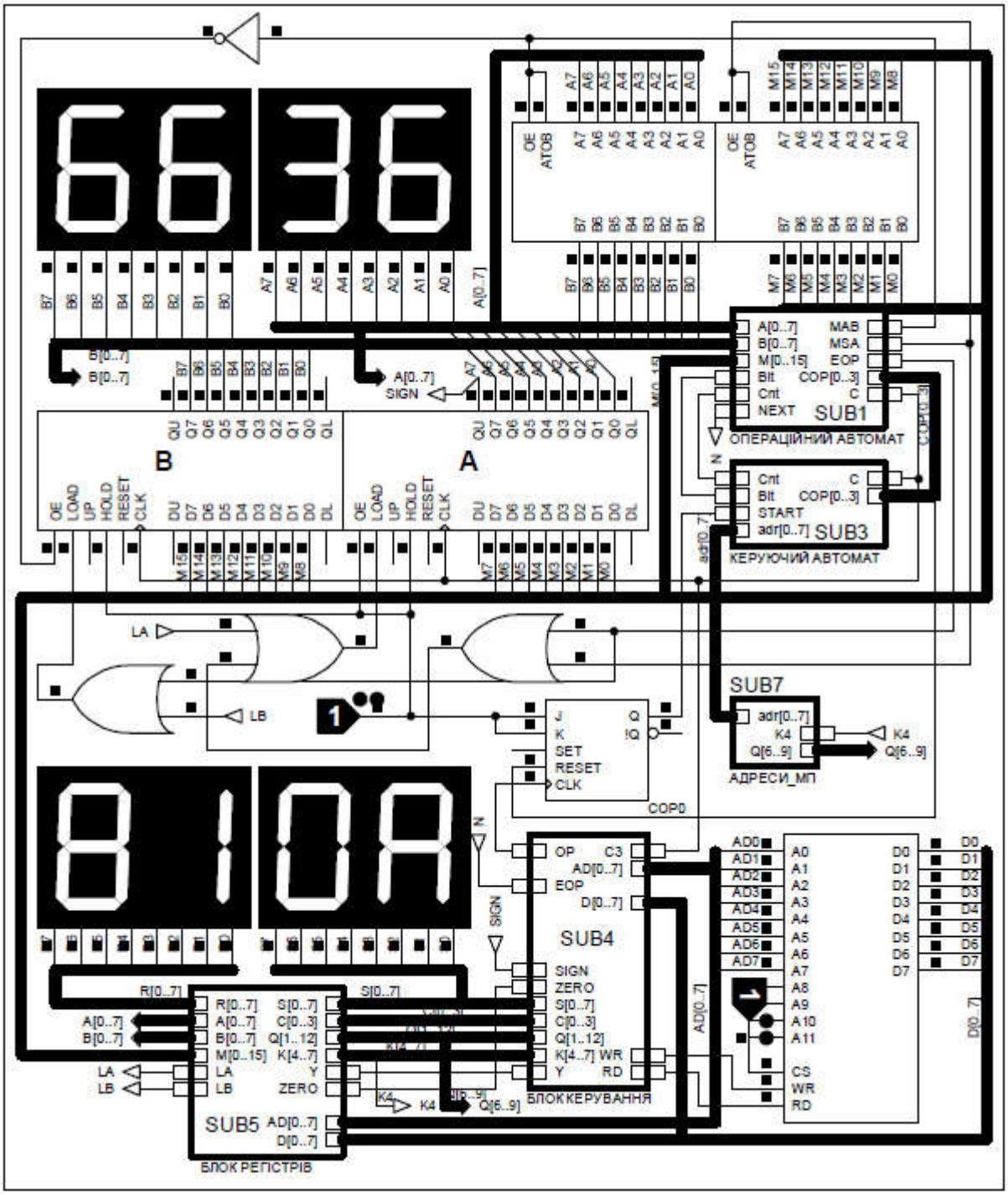


Рис. 13.4 - Загальний вигляд схеми мікропроцесорної системи

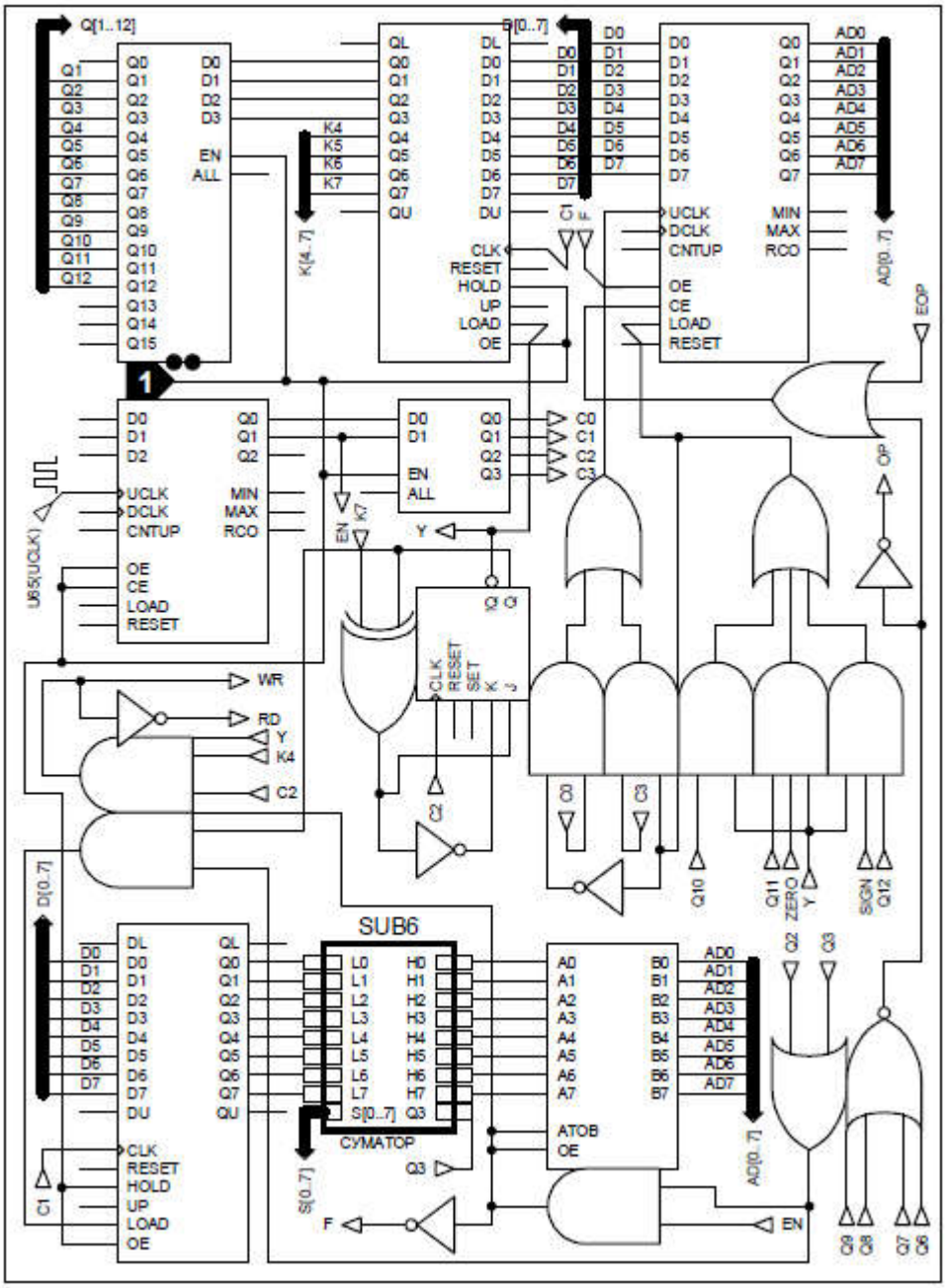


Рис. 13.5 - Вміст SUB4 – блок керування МПС

(генератор синхроімпульсів перенесено сюди з керуючого автомата)

На рис. 13.5 зображена схема блоку керування проектованої МПС. Тут можна виділити: блок генерації синхросигналів (лічильник із дешифратором із виходами C0 – C3), блок формування керуючих сигналів (дешифратор із виходами Q0 – Q12 та схема керування виконанням команд із виходом Y), програмний лічильник (схема з виходами AD0 – AD7), регістр коду команди (схема з виходами K0 – K7), блок формування адреси операндів з опосередкованою адресацією (схема, що містить допоміжний регістр,

приєднаний до суматора SUB6).

На рис. 13.6 зображено вміст підсхеми SUB6 – суматор для формування адреси операндів при опосередкованій адресації. Тут при виконанні команд MOV register, [address + D] або MOV [address + D], register (при активному сигналі Q3) вміст індексного регістра D додається до деякої базової адреси, записаної у допоміжний регістр (відносно-індексна адресація).

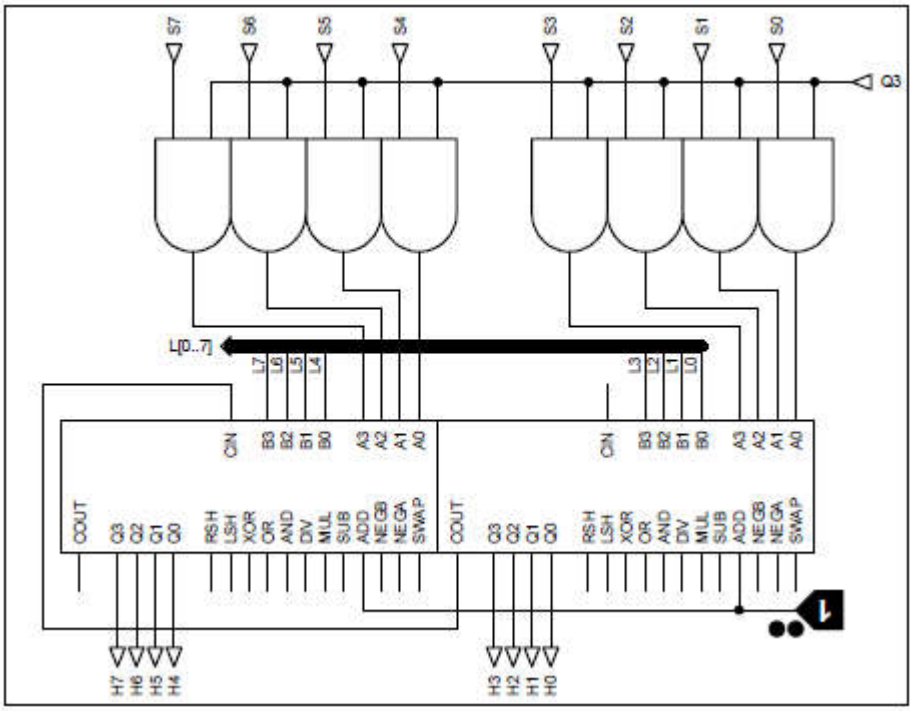


Рис. 13.6 – Вміст SUB6 – суматор для формування адреси операндів при опосередкованій адресації

Зображена на рисунку 13.5 організація блока синхронізації зумовлена необхідністю конвеєрного (послідовного) функціонування частин МПС під час виконання команд. Призначення синхросигналів С0 – С3 таке (рис. 13.7):

С0 – приріст програмного лічильника (Program Counter – PC);

С1 – запис у регістр команд (Instruction Register – IR), якщо скинуто тригер дозволу запису (Y = 1); запис адреси операнда у допоміжний регістр у випадку опосередкованої адресації при встановленому тригері дозволу запису (Y = 0);

С2 – встановлення тригера дозволу запису, якщо в IR записалася двобайтова команда (заборона запису в IR наступному машинному циклі) або скидання тригера дозволу запису, якщо в попередньому машинному циклі він

був встановлений; виконання команд запису операндів у пам’ять;

С3 – виконання команди, якщо скинуто тригер дозволу.

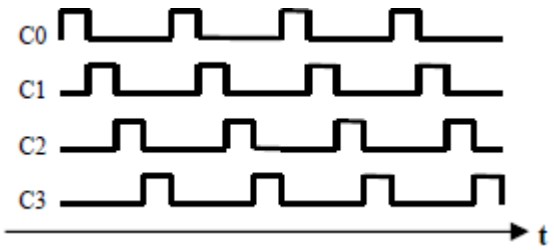


Рис. 13.7 – Часові діаграми синхроімпулсів

На рис. 13.7 зображено часові діаграми сигналів синхронізації. З рисунка видно, що на чотирьох окремих лініях синхронізації (С0 – С3), приєднаних до виходів дешифратора (рис. 13.5), почергово з’являються активні сигнали. Таким чином, у кожен момент часу лише одна лінія синхронізації активна, тобто виконується лише одна специфічна мікрооперація.

Опишемо блок керування процесом виконання команд, що містить тригер дозволу запису із виходом Y. Код команди (перший байт у двобайтових командах) записується у регістр команд (IR) і запам’ятовується там упродовж усього циклу виконання команди. Сигнал Y керує процесом дозволу/заборони запису даних у регістр IR, що забезпечує запис у цей регістр лише коду команди, оскільки запис у цей регістр будь-якої іншої інформації призведе до неправильної її дешифрації і до збою в роботі мікропроцесорної системи.

Однобайтні команди виконуються за один машинний цикл, а для двобайтних необхідні два машинні цикли, причому на першому машинному циклі відбувається завантаження коду команди у регістр команд (IR) та її дешифрація, а на другому машинному циклі – власне виконання команди з використанням операндів або адрес, що з’являються на шині даних. Під час виконання двобайтних команд відповідні керуючі лінії на виході дешифратора коду операції активні протягом двох машинних циклів. Отже, виникає необхідність у проектуванні блока керування процесом виконання команд. Даний блок повинен забороняти виконання двобайтних команд на першому машинному циклі та дозволяти на другому. Це виконується за допомогою генерації керуючого сигналу Y.

Згідно з описаною логікою роботи блока керування процесом виконання команд можна спроектувати його схему, яка являє собою тригер, охоплений зворотним зв’язком у вигляді елемента XOR, на другий вхід якого подано старший розряд коду команди, тобто біт, що визначає кількість байтів у команді (див. рис. 13.5).

Арифметичні команди ADD, SUB, MUL та DIV є однобайтними, але час їх виконання визначається декількома машинними циклами, упродовж яких виконуються відповідні мікропрограми у арифметично-логічному блоці. Під час виконання арифметичних команд приріст програмного лічильника блокується сигналом OP (operation). По завершенні виконання мікропрограми генерується сигнал EOP (end operation), програмний лічильник розблоковується та розпочинається виконання наступної команди.

На рис. 13.8 показано вміст підсхеми SUB5 – блок допоміжних регістрів (регістра-лічильника C та індексного регістра D), що містить також допоміжні буферні схеми, які служать для передачі операндів між пам’яттю та регістрами арифметично-логічного блока A і B.

Рис. 13.9 демонструє вміст підсхеми SUB7 – блок формування адреси мікропрограми для відповідної арифметичної команди.

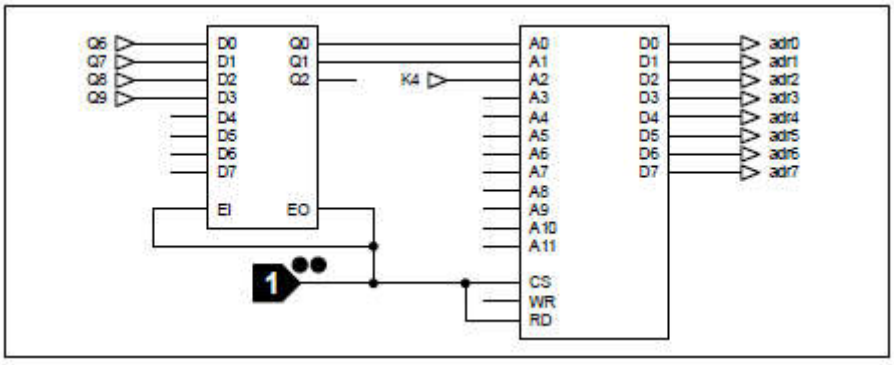


Рис. 13.9 - Вміст SUB7 – блок формування адреси мікропрограми АЛП за кодом відповідної арифметичної команди

Для аналізу працездатності спроектованої мікропроцесорної системи складемо деяку програму, яка б сортувала елементи масиву згідно з ознакою парності, причому по закінченні роботи програми, на початку масиву повинні бути розташовані непарні числа, а в кінці – парні.

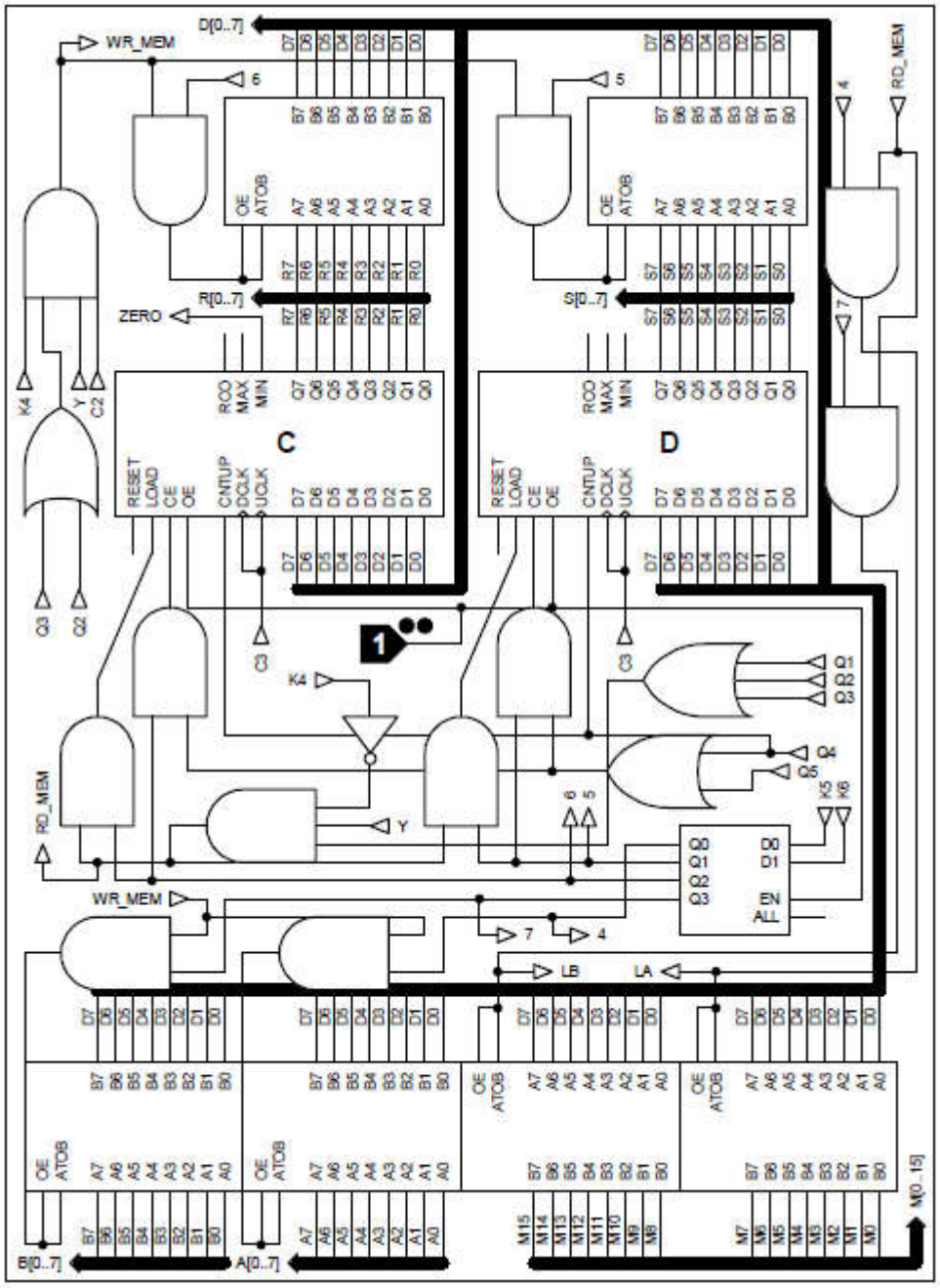


Рис. 13.8 – Вміст SUB5 – блок допоміжних регістрів

(регістра-лічильника C та індексного регістра D), що містить також допоміжні буферні схеми (у регістрах-лічильниках C і D встановіть Use CNTUP Direction Control)

Програма має такий вигляд:

Мнемонічний запис програми	**Адреси**	**Двійкове зображення команд**	**Коментарі**
JMP begin	0	10001010	Безумовний перехід
array DB 3,6,9,2,8,4,5,7,1,3	1	00001101	Адреса переходу
	2	00000011	Масив чисел: 3
	3	00000110	6
	4	00001001	9

	5	00000010	2
	6	00001000	8
	7	00000100	4
	8	00000101	5
	9	00000111	7
	10	00000001	1
	11	00000011	3
help_var DB 0	12	00000000	Допоміжна комірка пам’яті
begin: MOV C, 9	13	11000001	Занесення у регістр С числа 9
	14	00001001	(к-ть елементів array мінус 1)
MOV D, 0	15	10100001	Занесення у регістр D
	16	00000000	числа 0
next: MOV A, array[D]	17	10000011	Занесення у регістр А вмісту
	18	00000010	комірки за адресою [array+D]
MOV help_var, A	19	10010010	Занесення у допоміжну
	20	00001100	комірку пам’яті вмісту рег. А
MOV B, 2	21	11100001	Занесення у регістр В числа 2
	22	00000010	(для визначення А mod 2)
DIV	23	00001001	Ділення вмісту А на В
MOV A, 128	24	10000001	Занесення у регістр А числа
	25	10000000	128 (одиниця у 7-му розряді)
MUL	26	00001000	Множення вмісту А на В
JS n_xch	27	10001100	Перехід, якщо SIGN = 1
	28	00110101	Адреса переходу
MOV A, array[D+1]	29	10000011	Занесення у А вмісту комірки
	30	00000011	за адресою [array + D + 1]
MOV B, 2	31	11100001	Занесення у регістр В числа 2
	32	00000010	(для визначення А mod 2)
DIV	33	00001001	Ділення вмісту А на В
MOV A, 128	34	10000001	Занесення у регістр А числа
	35	10000000	128 (одиниця у 7-му розряді)
MUL	36	00001000	Множення вмісту А на В
JS ljns	37	10001100	Перехід, якщо SIGN = 1
	38	00101001	Адреса переходу
JMP n_xch	39	10001010	Безумовний перехід
	40	00110101	Адреса переходу
ljns: MOV A, array[D+1]	41	10000011	Занесення у А вмісту комірки
	42	00000011	за адресою [array + D + 1]
MOV array[D], A	43	10010011	Занесення вмісту регістра А у
	44	00000010	комірку за адресою [array+D]
MOV A, help_var	45	10000010	Занесення у регістр А вмісту
	46	00001100	допоміжної комірки пам’яті
MOV array[D+1], A	47	10010011	Занесення вмісту А у комірку
	48	00000011	за адресою [array + D + 1]
INC C	49	01000100	Збільшення на одиницю С
DEC D	50	00100101	Зменшення на одиницю D
JMP next	51	10001010	Безумовний перехід
	52	00010001	Адреса переходу
n_xch: INC D	53	00100100	Збільшення на одиницю D
DEC C	54	01000101	Зменшення на одиницю С
JZ end	55	10001011	Перехід, якщо ZERO=1 (C=0)
	56	00111011	Адреса переходу
JMP next	57	10001010	Безумовний перехід
	58	00010001	Адреса переходу
end: MOV help_var, C	59	11010010	Занесення у допоміжну
	60	00001100	комірку пам’яті вмісту С (0)
fin: NOP	61	00000000	Порожня операція

JMP fin	62 63	10001010 00111101	Безумовний перехід Адреса переходу

Наведена програма сортує елементи масиву array, який являє собою сукупність із десяти чисел, а саме: 3, 6, 9, 2, 8, 4, 5, 7, 1, 3. Переконайтеся у правильності роботи складеної програми. Коли процесор виконуватиме порожній цикл (fin: NOP JMP fin) елементи масиву *array* будуть розташовані в такому порядку: 3, 9, 5, 7, 1, 3, 6, 2, 8, 4, тобто перші шість елементів – непарні числа, а останні чотири – парні, як показано на рис. 13.10.

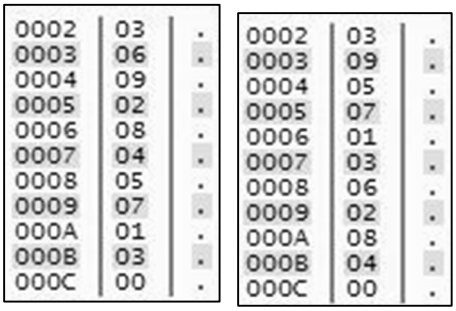


Рис. 13.10 – Розташування елементів масиву в пам’яті до (зліва) і після (справа) роботи програми

Практична робота № 14

Складання схем пам'яті з адресним способом доступу до даних та дослідження особливостей їх роботи

Способи доступу до даних у напівпровідниковій пам'яті

Основні теоретичні положення

У напівпровідникових запам'ятовуючих пристроях (ЗП) виділяють адресні, послідовні й асоціативні способи доступу до даних.

В адресних ЗП усі комірки пам'яті в момент звернення рівнодоступні. До адресних ЗП відносять: RAM (Random Access Memory) – ОЗП (оперативний ЗП) або ЗПДВ (ЗП із довільною вибіркою) та ROM (Read Only Memory) – ПЗП (постійний ЗП).

Оперативні ЗП зберігають дані, необхідні при виконанні поточної програми; вони можуть бути змінені в будь-який момент часу. Оперативні ЗП здебільшого енергонезалежні. У постійних ЗП вміст комірок або взагалі не змінюється, або змінюється рідко в спеціальних режимах.

Запам'ятовуючі пристрої RAM поділяються на статичні SRAM (Static RAM) і динамічні DRAM (Dynamic RAM). У статичних RAM елементами пам'яті є тригери. Вони зберігають свій стан, поки схема має напругу живлення і нові дані не записуються. У динамічних RAM дані зберігаються у вигляді зарядів конденсаторів, створюваних компонентами МОН-транзисторів. Саморозряд конденсаторів веде до руйнування даних, тому вони періодично (кожні 2 – 30 мс) мають регенеруватися. Але щільність упакування динамічних елементів пам'яті в 4 – 5 разів перевищує такий самий показник для статичних RAM. Регенерація даних здійснюється за допомогою спеціальних контролерів. Розроблені також DRAM із внутрішніми схемами регенерації; такі ЗП називаються квазістатичними.

Постійна пам'ять типу ROM(M) програмується при виготовленні за допомогою масок, тому такі ПЗП називають масковими. В подальших різновидах ROM у позначеннях є буква P (від Programmable). Це – пам'ять, що одноразово програмується користувачем PROM (ППЗП – програмовані ПЗП)

і багаторазово – EPROM, EEPROM. Пам’ять типу Flash елементами пам’яті подібна до EEPROM, але має структурні та технологічні особливості, які дозволяють виділити її в окремий тип.

У ЗП із послідовним доступом дані, що записуються, створюють чергу. Зчитування виконується слово за словом у порядку записів або навпаки. Прямий порядок зчитування використовується в буферах FIFO з дисципліною „перший прийшов – перший вийшов (First In – First Out)”, а також у файлових та циклічних ЗП.

Різниця між пам’яттю FIFO та файловим ЗП полягає в тому, що у FIFO записування в порожній буфер зразу доступне для читання (тобто надходить у кінець ланцюга), а у файлових ЗП доступ до читання можливий лише після деякої кількості звернень, яке дорівнює кількості елементів у ланцюзі.

У циклічних ЗП слова доступні одне за одним із постійним періодом, який визначається ємністю пам’яті. До них належить відеопам’ять VRAM.

Зчитування у зворотному порядку властиве стековим ЗП із дисципліною „останнім прийшов – першим вийшов”. Такі ЗП називаються буферами LIFO (Last In – First Out).

Час доступу до конкретної одиниці інформації, що зберігається в послідовних ЗП, є випадковою величиною. В найгіршому випадку для такого доступу треба переглянути весь об’єм інформації, що зберігається у цій пам’яті.

Асоціативний доступ реалізує пошук інформації за деякою ознакою, а не за адресою. В найбільш повній версії всі слова, які зберігаються в пам’яті, можуть одночасно перевірятися на відповідність ознаці, наприклад, на збіг визначених полів слів – тегів (від tag) за ознакою, яку задає вхідне слово (тегова адреса). На вихід передаються слова, які задовольняють ознаку. Дисципліна видавання слів, якщо тегу задовольняє кілька слів, та дисципліна записування нових даних можуть бути різними. Основна область використання асоціативної пам’яті в комп’ютерах – кешування даних.

Основні структури напівпровідникової пам’яті. Елементний базис

пам’яті сучасних комп’ютерів складають мікросхеми різного ступеня інтеграції. Основою будь-якого запам’ятовуючого пристрою (ЗП) є елемент пам’яті (ЕП) статичного або динамічного типу, призначений для записування, зберігання і зчитування одного біта інформації – цифри 0 або 1. Сукупність ЕП, які утворюють п-розрядне слово, називають коміркою пам’яті (КП). Множина КП утворює запам’ятовуючий масив, який називається матрицею М елементів пам’яті.

Кожна матриця М у пристрої пам’яті має систему адресних і розрядних ліній (провідників). Адресні (словникові) лінії служать для виділення за адресою будь-якої КП. Сукупність різних адресних кодів утворює адресний простір пам’яті. Розрядні лінії записування (ЛЗП) служать для введення в кожний розряд вибраної КП цифри 0 або 1 відповідно до вхідної інформації. Розрядні лінії зчитування (ЛЗЧ) служать для знімання інформації, яка зберігається, з розряду вибраної КП. Часто використовують спільну лінію записування-зчитування (ЛЗЗ). Адресні та розрядні лінії разом називаються лініями вибірки. Якщо довжина адресного коду дорівнює k, то кількість слів N, які зберігаються в пам’яті як окремі одиниці даних, визначаються зі співвідношення $N=2^k$.

Структуру пам’яті визначає спосіб розподілу КП між адресними та розрядними лініями. За цією ознакою виділяють такі структури пам’яті: 2D, 3D, 2,5D і модифіковану – 2DM (D від Dimension – розмірність).

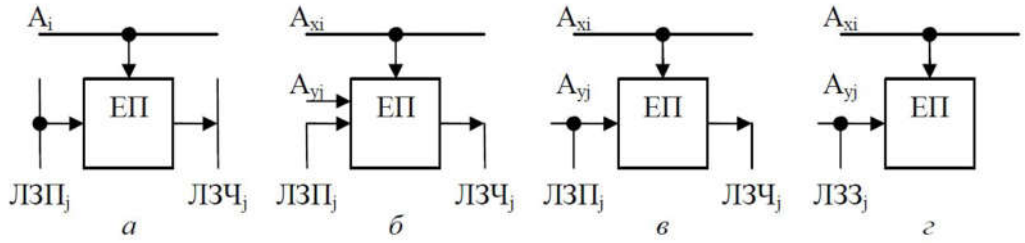


Рис. 14.1. Узагальнене поняття структури пам’яті: а – 2D, б – 3D, в – 2,5D, г – 2DM

У системі 2D кожний ЕП має одну адресну лінію Ai (одне D), лінії записування $ЛЗП_j$ і зчитування $ЛЗЧ_j$, які спільно утворюють друге D (рис. 14.1а). У структурі 3D адресу розділяють на дві частини: старша A_x визначає

адреси рядків, а молодша A_y – адреси стовпців; разом вони утворюють 2D. Лінії записування і зчитування утворюють третє D (рис. 14.1б). У структурі 2,5D одна з ЛЗП або ЛЗЧ суміщена з напівадресою A_{xi} або A_{yi} (рис. 14.1в). У модифікованій системі 2DM використовується спільна лінія $ЛЗЗ_j$, яка поєднується з адресною лінією A_{yi} мультиплексорами.

Розглянуті структури характерні для статичних ОЗП і пам'яті типу ROM. Структури динамічних ОЗП мають свою специфіку. До складу мікросхем пам'яті входять:

- матриця елементів пам'яті М, яка містить N рядків і m стовпців (за кількістю розрядів слова);

- буфер і дешифратор k-розрядної адреси з кількістю виходів $N = 2^k$;

- буферні формувачі вхідних і вихідних інформаційних сигналів у режимах записування і зчитування й буфер даних;

- блок місцевого керування.

Розглянемо коротко організацію матриць пам'яті, що мають різну структуру. Матриця М пам'яті зі структурою 2D організована так, що при звертанні до пам'яті вибираються ЕП, які розміщені на збудженому виході дешифратора адреси. Недоліком структури 2D є складність побудови дешифратора адреси з кількістю виходів N, яке дорівнює кількості слів, що зберігаються у пам'яті. Тому структура типу 2D використовується в ЗП малої ємності.

У пам'яті зі структурою 3D адресний код розділяється на дві частини A_x і A_y, кожна з яких декодується окремими дешифраторами. Матриця М складається з m підматриць за кількістю розрядів слова. Кожна матриця зберігає значення свого і-го розряду всіх N слів. При звертанні до пам'яті вибирають m запам'ятовуючих елементів (по одному з кожної підматриці), які знаходяться на перетині рядка і стовпця збуджених виходів дешифраторів молодшої і старшої частин адреси. Така структура часто використовується з однорозрядною організацією N*1 біт. Для цієї пам'яті ємністю 1 К слів треба мати два дешифратори з кількістю виходів у кожному $N=2^5=32$. Для пам'яті зі

структурою 2D із тією самою ємністю дешифратор має 1024 виходи.

Недоліком структури 3D є використання складних ЕП, які допускають двокоординатну вибірку та складнішу структуру матриці М. У пам'яті з модифікованою структурою 2DM поєднуються переваги структур 2D і 3D.

У ЗП зі структурою 2DM адресний код довжиною k розбивається на дві частини: $A_x=A_kA_{k-1}…A_{r+1}$, що надходить на дешифратор рядків, та $A_y=A_rA_{r-1}…A_1$, що подається на входи мультиплексорів з організацією „2k → 1". Дешифратор обслуговує 2^{k-r} рядків, кожний з яких зберігає 2^k m-розрядних слів. У кожній групі зберігаються значення однойменних розрядів, і обслуговується вона своїм мультиплексором з організацією „2k → 1". Таким чином, активований вихід дешифратора вибирає рядок, а молодші розряди адреси за допомогою мультиплексорів забезпечують формування вихідного слова (по одному розряду з кожної групи). Організація пам'яті зі структурою 2DM найбільш розповсюджена, особливо для схем великої ємності.

Виконання роботи

Складіть схему для дослідження функціонування чотирирозрядної пам'яті з адресним способом доступу до даних, як зображено на рис. 14.2. Виходи чотирирозрядного двійкового лічильника приєднайте до адресних входів схем пам'яті та через буферні схеми – до входів даних. Передбачте в схемі лінію керування зчитуванням / записом інформації в пам'ять. У підсхемі SUB17 використайте готову модель мікросхеми пам'яті MEMORY_12_8, приєднавши до входу WR мікросхеми через двовходовий елемент логічного множення (AND_2), до входів якого приєднайте лінії WR та CLK. У підсхемі SUB18 зберіть схему пам'яті зі структурою 2D, як це показано на рис. 14.3. Вміст чотирирозрядних комірок пам'яті SUB1 – SUB16 побудуйте на основі синхронних D-тригерів зі статичним керуванням записом, як це показано на рис. 14.4. З рисунка видно, що під час запису інформації у комірки пам'яті (WR = 1) цифрові ключі комутують між собою лінії даних та входи тригерів, а під час зчитування інформації з комірок пам'яті (RD = 1) цифрові ключі комутують між собою лінії даних та виходи тригерів. Окрім того, необхідною

умовою можливості зчитування / запису інформації в деяку комірку пам’яті є наявність сигналу логічної одиниці на відповідній адресній лінії з виходу дешифратора адреси.

Результатом роботи складеної схеми повинен бути процес запису у схеми пам’яті двійкових кодів цифр 0, 1, 2, ..., E, F упродовж перших 16 тактів генератора синхроімпульсів CLK та зчитування записаної інформації впродовж наступних 16 тактів, що видно з часових діаграм цифрового аналізатора на рис. 3.2. Керування процесом запису / зчитування відбувається за допомогою імпульсного сигналу WR/!RD, який з’єднаний також із буферними схемами для переведення у високоомний стан лінії від лічильника під час зчитування інформації зі схем пам’яті.

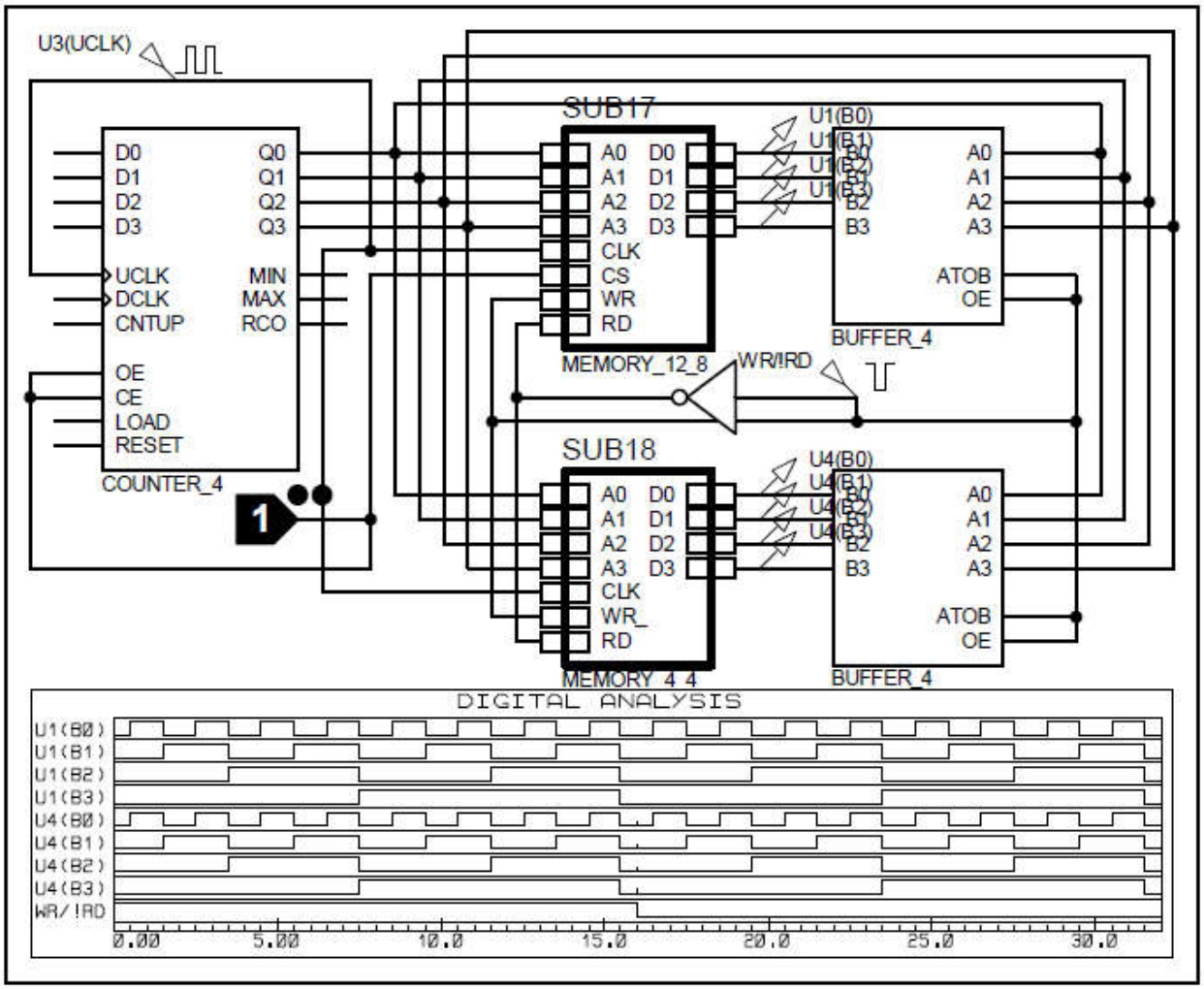


Рис. 14.2 - Основна схема

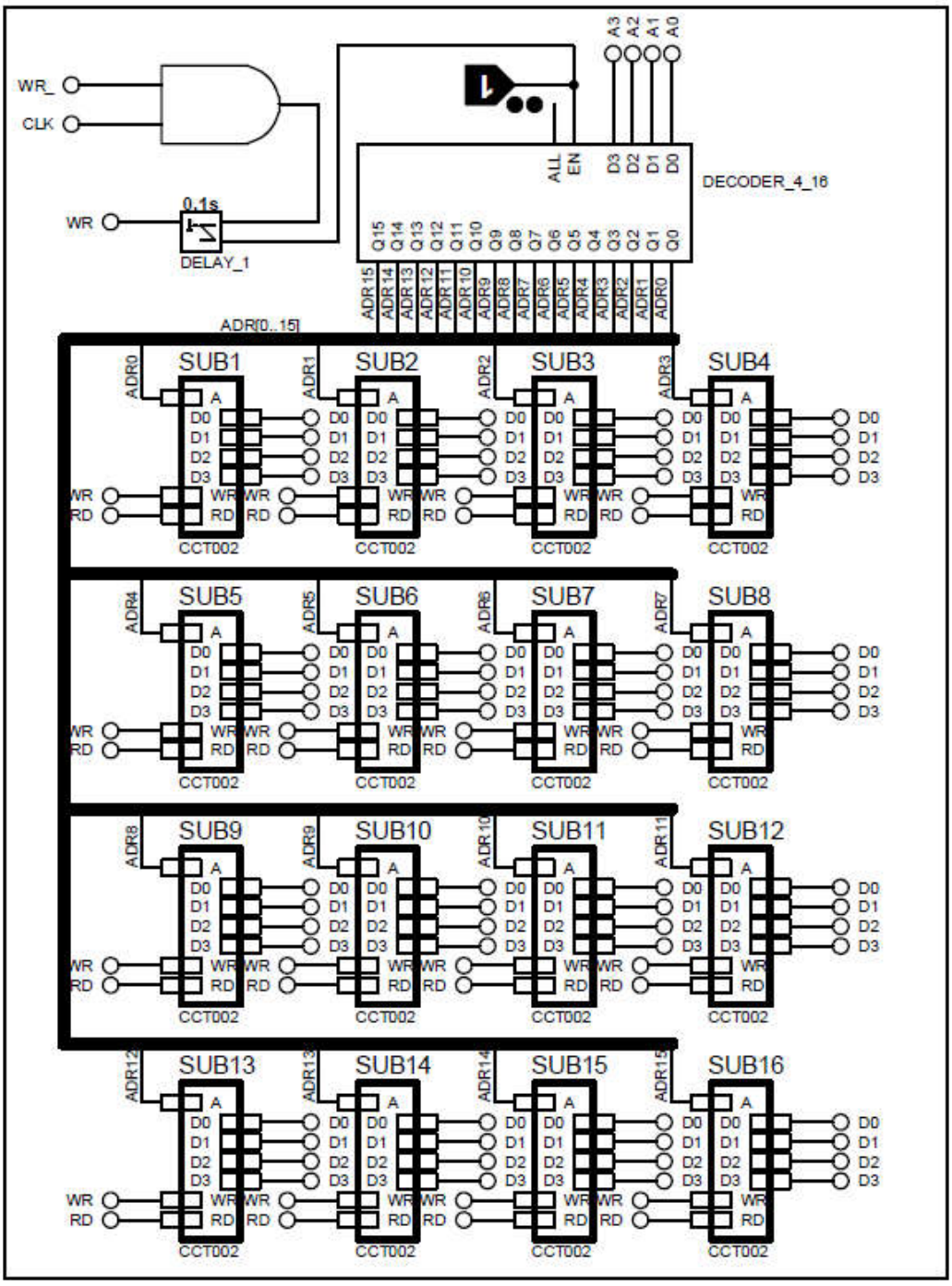


Рис. 14.3 - Вміст SUB18 на рис. 14.2

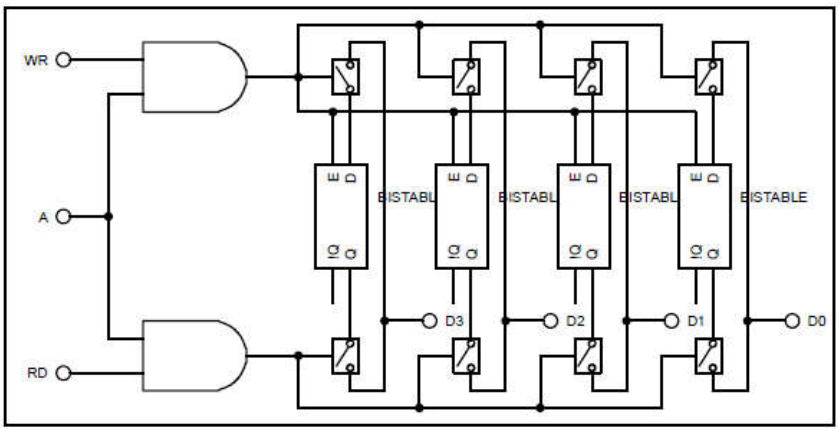


Рис. 14.4 - Вміст SUB1 – SUB16 на рис. 14.3

Практична робота № 15

Організація переривань в мікроконтролерах AVR

Вивчення принципів програмування задач управління з використанням механізму переривань мікроконтролерів.

Основні теоретичні положення

Важливим елементом мікроконтролера є система переривань. Ця система наявна в будь-якому сучасному мікроконтролері. Вона також є в усіх мікроконтролерах AVR. Система переривань мікроконтролера обслуговує декілька джерел переривань. Кількість джерел переривань для різних мікроконтролерів є різною.

Основне призначення системи переривань – організація взаємодії всіх засобів мікроконтролера при вирішенні прикладних задач. У всіх моделях мікроконтролерів AVR містяться засоби апаратної, а, отже, паралельної реалізації різноманітних стандартних задач, які мають вирішуватися при управлінні технічними об’єктами. Всі апаратні засоби мікроконтролера (процесор, таймери, інтерфейси, АЦП і т. п.) можуть працювати і працюють паралельно та незалежно один від одного. Очевидно, що можливість паралельної реалізації декількох функцій вельми істотно розширює можливості мікроконтролера і збільшує швидкодію. В той же час кожен з пристроїв забезпечує виконання будь-яких обмежених функцій, які є тільки елементами загальної прикладної задачі. Ця спільна задача на певних етапах вимагає обміну даними між пристроями, запуску або зупинки окремих процедур, зміни параметрів або режимів роботи і т. ін. Система переривань є необхідним і найважливішим елементом управління роботою апаратних засобів.

Система векторів переривань дозволяє виділити моменти часу, коли завершуються чергові операції й потрібна певна реакція інших елементів мікроконтролера для продовження роботи.

Саме процедури обробки переривань в мікроконтролерах AVR дозволяють виконувати досить гнучкий, програмно-керований контроль

реалізованих функцій. Засоби керування роботою апаратних засобів в обробці переривань перераховані нижче:

• встановлені пріоритети векторів переривань – при одночасному надходженні декількох запитів переривань спочатку викликається вектор переривання з найменшою адресою;

• програмне управління прапорцем загального дозволу переривання (прапорець I регістру SREG) – в будь-яких програмах на довільних етапах за допомогою цього прапорця можна програмно забороняти або дозволяти обробку всіх переривань;

• апаратне керування прапорцем загального дозволу переривання – при виклику будь-якого вектора переривання прапорець глобального дозволу I очищається апаратно і відновлюється (командою reti) при завершенні обробки переривання (крім того, підпрограма обробки будь-якого вектора переривання також може програмно змінювати стан прапорця I, дозволяючи обробку інших переривань);

• вектори переривань містять у відповідних регістрах введення/виведення індивідуальні прапорці дозволу (маскування) – це дозволяє програмно дозволяти або забороняти виклик кожного з векторів переривань на будь-яких етапах роботи незалежно від інших векторів;

• прапорці виклику векторів переривань в регістрах введення/виведення очищаються апаратно при зверненні до підпрограми обробки переривання;

• прапорці виклику векторів переривань можуть очищатися програмно записом одиниці в ці біти регістрів введення/виведення – це дозволяє, за необхідності, скасовувати виклик підпрограм обробки переривань.

Стандартний розподіл паралельно реалізованих в мікроконтролері функцій для керування технічним об’єктом може бути таким, як описано нижче (кожен пункт в даному прикладі – окремий паралельно реалізований процес).

1. АЦП виконує перетворення аналогового сигналу, що надходить від датчика (час перетворення близько 100 мкс, за цей період процесор може

виконати декілька сот команд робочої програми).

2. Процесор, виконуючи команди основної програми, робить обробку кодів, що надійшли раніше від АЦП і інших інтерфейсів мікроконтролера, та формує нові значення сигналів керування і даних для індикації.

3. Паралельні порти введення/виведення забезпечують виведення сформованих раніше сигналів керування та індикації.

4. Таймери формують сигнали з заданими раніше в програмі часовими характеристиками і/або в режимі модулятора ШІМ видають сигнали управління виконавчими пристроями.

5. Інтерфейс SPI передає отримані від процесора дані для управління символьним або матричним індикатором.

6. Інтерфейс UART реалізує обмін даними з COM-портом персонального комп'ютера для координації роботи мікроконтролера з іншими пристроями.

7. Зовнішні переривання забезпечують прийом і обробку сигналів цифрових датчиків і сигналів управління, що вимагають швидкої реакції мікроконтролера (наприклад, сигнал аварійної зупинки системи управління).

Очевидно, що перераховані процеси асинхронні один відносно одного, кожен з пристроїв реалізує задані функції зі своїми режимами і часовими межами. Основну координувальну роль може виконувати тільки процесор, забезпечуючи необхідний обмін даними між пристроями і керуючи їх режимами та параметрами через регістри введення/виведення. Основна вимога, яка висувається до організації взаємодії, – безконфліктний доступ до ресурсів, оскільки різні засоби мікроконтролера можуть одночасно виставляти запити на доступ до одних і тих же ресурсів. Коректне управління має забезпечуватися навіть за умови зниження ефективності роботи окремих апаратних засобів. Необхідні координувальні функції мають бути передбачені в алгоритмі основної робочої програми і можуть бути виконані завдяки ефективній системі обробки переривань. Для реалізації координувальних функцій необхідні певні, в першу чергу програмні, ресурси мікроконтролера,

ці функції зазвичай досить складні і є найважливішою частиною загального алгоритму роботи.

Механізм реалізації переривань в мікроконтролерах. Переривання – це запуск спеціальної підпрограми (званої «обробником переривання» чи «програмою обслуговування переривання»), і викликається воно сигналом апаратури. На час виконання цієї підпрограми реалізація поточної програми зупиняється (рис. 15.1). Термін «запит на переривання» (interrupt request) використовується тому, що іноді програма відмовляється підтвердити переривання і запустити обробник переривання негайно.

Рис. 15.1 – Послідовність дій під час реалізації переривання

МК може не реагувати на переривання, поки не завершиться виконання поточної задачі – це реалізується шляхом заборони (маскування) обслуговування запиту переривання. Після розв’язання задачі можливий один з двох варіантів: скидання маски і дозвіл обслуговування переривання, що приведе до виклику обробника переривання, чи аналіз значення бітів, що вказують на надходження запитів переривання і безпосереднє виконання програми обслуговування без виклику обробника переривання. Такий метод обробки переривання використовується, коли потрібно забезпечити заданий час виконання основної програми, оскільки будь-яке переривання може порушити реалізацію необхідного інтерфейсу.

Обробник завжди забезпечує нижче вказану послідовність дій.

1. Зберегти вміст регістрів контексту.

2. Скинути контролер переривань і устаткування, що викликало запит.

3. Обробити дані.

4. Відновити вміст регістрів контексту.

5. Повернутися до перерваної програми.

Регістри контексту – це регістри, що визначають поточний стан виконання основної програми. Зазвичай до них відносять лічильник команд ЛК, регістри стану і акумулятор. Інші регістри МК, наприклад, індексні регістри, можуть бути використані в процесі обробки переривання, тому їхній вміст також необхідно зберегти. Всі інші регістри є специфічними для конкретного типу МК і його застосування.

Після скидання у вихідний стан контролер переривань готовий сприймати наступний запит, а устаткування, що викликає переривання, готове надсилати запит, коли виникають відповідні причини. Якщо надійде новий запит переривання, то регістр маскування переривань процесора запобіжить обробці переривання, але регістр стану переривань зафіксує цей запит, який буде очікувати свого обслуговування. Після завершення обслуговування поточного переривання маска переривання буде скинута, і запит, що надійшов, знову надійде на обробку. Вкладені переривання складно виконати деяким типам МК, які не мають стека. Ці переривання так само можуть викликати проблеми, пов’язані з переповненням стека.

Іноді МК може швидко відреагувати на запит переривання, прийнявши необхідні дані, які будуть потім використані після вирішення поточної задачі. Це реалізується шляхом збереження надійшовших даних в масиві пам’яті та наступної їх обробки, коли виконання вихідної програми буде завершено. Такий спосіб обслуговування є гарним компромісом між негайною повною обробкою переривання, що може потребувати багато часу, та ігноруванням переривання, що може призвести до втрати інформації про подію, яка викликала переривання.

При обробці переривання вміст регістра стану звичайно (але не завжди) автоматично зберігається разом із вмістом лічильника команд (ЛК) перед обробкою переривання. Це позбавляє від необхідності зберігати його в пам’яті програмними засобами за допомогою команди пересилання, а потім

відновлювати при поверненні до вихідної програми. Однак таке автоматичне збереження реалізується не в усіх типах МК.

Якщо вміст регістра стану зберігається перед початком обробки переривання, то за командою повернення виробляється його автоматичне оновлення. Якщо вміст інших регістрів змінюється при виконанні обслуговування переривання, то воно також має бути збережене в пам'яті до зміни і відновлено перед поверненням в основну програму.

«Вектор переривання» – це адреса, яка завантажується в лічильник команд при переході до обробника переривання. Існує кілька типів векторів. Адреса, що завантажується в ЛК при запуску МК (RESET), називається «вектором скиду». Для різних переривань можуть бути задані різні вектори. Але іноді різним перериваннях призначається один вектор. Це не має викликати проблем при роботі з МК, оскільки найчастіше він виконує одну єдину програму. У МК, де апаратна частина добре відома, не має виникнути жодних проблем при спільному використанні векторів переривань.

Управління перериваннями в мікроконтролерах AVR. У мікроконтролерах AVR за керування перериваннями відповідають, головним чином, чотири регістри:

- GIMSK (General Interrupt Mask Register – GICR для Atmega): дозволяє або забороняє зовнішні переривання за входом INT0/INT1;

- GIFR (General Interrupt Flag Register) – регістр прапорців зовнішніх переривань;

- TIMSK (Timer/Counter Interrupt Mask Register) – регістр маскування переривань від таймера/лічильника T/C0 і T/C1;

- TIFR (Timer/Counter Interrupt Flag Register) – регістр прапорців переривань від таймерів/лічильників.

Про стан переривання сигналізує відповідний прапорець, який встановлюється або скидається в регістрі прапорців. Навіть якщо в регістрі маски переривань встановлений відповідний окремий розряд дозволу переривання, то переривання можуть активізуватися тільки тоді, коли в

регістрі стану SREG установлений розряд загального дозволу переривань I (розряд 7). Якщо це має місце і настає переривання, то виконання програми відгалужується за відповідною адресою (див. табл. 15.1) і розряд загального дозволу переривань I в регістрі SREG скидається в стан логічного 0, блокуючи тим самим подальші переривання. Якщо потрібно перервати підпрограму іншим перериванням, то після входу в підпрограму обробки переривання програма користувача має встановити прапорець I в логічну 1.

Разом з входом в підпрограму обробки переривання апаратно скидається також і відповідний прапорець, що викликав переривання. Деякі прапорці переривань можуть бути скинуті самим користувачем за допомогою встановлення відповідного прапорця в логічну 1.

Регістр GIMSK (GICR). Регістр GIMSK (GICR для Atmega 8515) (рис. 15.2) розташований в області введення/виведення за адресою 003B (адреса в SRAM – 0x005B), використовується для дозволу зовнішніх переривань.

7	6	5	4	3	2	1	0
INT1	INT0	–	–	–	–	–	–

Рис.15.2 – Структура регістра GIMSK (GICR)

Якщо розряд INT1/INT0 установлений у логічну 1, то зовнішнє переривання за входом INT1/INT0 буде дозволено до тих пір, поки встановлений у стан 1 розряд I в регістрі стану SREG.

Регістр GIFR. Стан зовнішнього переривання визначається за регістром GIFR (рис. 15.3), який розташований в області введення/виведення за адресою 0x003A (адреса SRAM – 0x005A).

7	6	5	4	3	2	1	0
INTF1	INTF0	–	–	–	–	–	–

Рис. 15.3 – Структура регістра GIFR

Прапорець INTF1/INTF0 встановлюється в логічну 1, якщо виникає зовнішнє переривання за сигналом на виводі INT1/INT0. При вході в підпрограму обробки переривання цей розряд переводиться апаратно в початковий стан логічного 0.

Регістри TIMSK і TIFR. Регістр TIMSK (рис. 15.4) розташований в

області введення/виведення за адресою 0x0039 (адреса в SRAM - 0x0059), використовується для дозволу переривань від таймерів/лічильників.

7	6	5	4	3	2	1	0
TOIE1	OCIE1A	OCIE1B	–	TICIE1	–	TOIE0	–

Рис. 15.4 – Структура регістра TIMSK

Стан переривань, що стосується таймерів/лічильників мікроконтролерів AVR, визначається за регістром TIFR (рис. 3.5), який розташований в області введення/виведення за адресою 0x0038 (адреса SRAM – 0x0058).

7	6	5	4	3	2	1	0
TOV1	OCF1A	OCF1B	–	ICF1	–	TOV0	–

Рис. 15.5 – Структура регістра TIFR

Коли розряд TOIE1 і розряд I в регістрі стану SREG встановлені в логічну 1, то дозволено переривання при переповненні T/C1. У разі переповнення в регістрі TIFR встановлюється прапорець TOV1.

Якщо розряд OCIE1A і розряд I в регістрі стану SREG встановлені в логічну 1, то дозволено переривання при збігу вмісту регістра порівняння A з поточним станом T/C1. У разі збігу в регістрі TIFR встановлюється прапорець OCF1A.

Якщо розряд OCIE1B і розряд I в регістрі стану SREG встановлені в логічну 1, то дозволяється переривання при збігу вмісту регістра порівняння B з поточним станом T/C1. У разі збігу в регістрі TIFR встановлюється прапорець OCF1B.

Якщо розряд TICIE1 і розряд I в регістрі стану SREG встановлені в логічну 1, то дозволяється переривання при виконанні умови захоплення. Коли виникає спрацьовування за захопленням, в регістрі TIFR встановлюється прапорець ICF1.

Якщо розряд TOIE0 і розряд I в регістрі стану SREG встановлені в логічну 1, то дозволяється переривання при переповненні таймера/лічильника T/C0. У такому випадку в регістрі TIFR встановлюється прапорець TOV0.

Встановлення в логічну 1 одного з прапорців в регістрі TIFR веде до переходу за відповідним вектором переривання. При вході в підпрограму

обробки переривання прапорець в регістрі TIFR апаратно скидається в логічний 0.

Обробка переривань в мікроконтролері ATmega 8515. Як входи зовнішніх переривань використовуються входи портів з альтернативною функцією: PD2, PD3 – для переривань INT0, INT1 і PE0 – для переривання INT2 в мікроконтролері ATmega8515. Запити зовнішніх переривань INT0, INT1 можуть бути подані низьким рівнем сигналу переривання (L), переходом від високого рівня сигналу до низького (HL – за негативним фронтом), переходом від низького рівня сигналу до високого (LH – за позитивним фронтом), запит INT2 тільки переходами (LH) і (HL). Залежно від типу запиту в регістрі управління мікроконтролера MCUCR необхідно встановити біти ISCx0 і ISCxl згідно з табл. 15.1 для кожного з переривань INTx (x = 0,1) і визначити біт ISC2 в регістрі EMCUCR для переривання INT2. При ISC2 = 0 переривання здійснюється за негативним фронтом, при ISC2 = 1 – за позитивним.

Таблиця 15.1 – Таблиця вибору типу запиту

ISCx1	ISCx0	Тип запиту
0	0	L
0	1	-
1	0	HL
1	1	LH

Далі підготуємо програму перемикання світлодіодів з використанням зовнішнього переривання від кнопки STOP. Згідно з поставленими умовами, в блок ініціалізації мікроконтролера внесемо низку змін:

- додамо вектор переривань;

- покажчик стека встановимо на останній осередок ОЗП;

- дозволимо зовнішнє переривання INTO (за сигналом 0 на лінії 2 порту PD) і переривання взагалі.

Оскільки зовнішнє переривання INTO подано сигналом на вході порту PD2, як STOP використовуємо кнопку SW2 і програмуємо PD2 на введення. Натискання кнопки STOP викликає переривання.

Приклад програми наведено нижче. Затримка подана підпрограмою DELAY.

Програма 15.1

```
;*****************************************
;Програма для почергового перемикання світлодіодів при натисканні
;на кнопку START (SW0). Після натискання на кнопку STOP
; перемикання припиняється і поновлюється (SW2)
;з місця зупинки при повторному натисканні на кнопку START
;*************************************
.include "m8515def.inc" ;файл визначень для ATmega8515
.def temp = r16 ;тимчасовий регістр
.def reg_led = r20 ;стан регістра світлодіодів
.equ START = 0 ;0-й розряд порту PD
;***Вектори переривань***
.org $000
rjmp INIT ;обробка скидання
.org $001
rjmp STOP_PRESSED ;обробка зовнішнього переривання
; INT0 (STOP)
;*** Ініціалізація МК ***
INIT: ldi reg_led,0xFE
ldi temp,$5F ; встановлення
ldi SPL, temp ; покажчика стека
ldi temp, $02 ; на останню
out SPH, temp ; комірку ОЗП
sec ;C=1
set ;T=1
ser temp ;ініціалізація виводів
out DDRB, temp ;порту PB на вивід
out PORTB, temp ;погасити світлодіоди
clr temp ;ініціалізація
out DDRD, temp ;порту PD на введення
ldi temp, 0x05 ; ввімкнення підтягувальних
out PORTD, temp ;резисторів порту PD
ldi temp, 0x7F ;дозвіл переривання INT0
out GICR, temp ; (6-й біт GICR)
ldi temp, 0x00 ;обробка переривання INT0
out MCUCR, temp ;за низьким рівнем
sei ; дозвіл переривань
WAITSTART: sbic PIND, START ;очікування натискання
rjmp WAITSTART ; кнопки START
LOOP: out PORTB, reg_led ; ввімкнення світлодіодів
rcall DELAY ;затримка
ser temp ;вимикання
out PORTB, temp ;світлодіодів
brts LEFT ;перехід, якщо прапорець T встановлений
sbrs reg_led, 0 ;пропуск наступної команди, якщо
; 0-й розряд reg_led встановлений
set ;T=1
```

```
ror reg_led ;зсув reg_led управо
rjmp LOOP
LEFT: sbrs reg_led,7 ;пропуск наступної команди, якщо
;1-й розряд reg_led встановлений
clt ;T=0
rol reg_led ;зсув reg_led вліво
rjmp LOOP
;***Обробка переривання від кнопки STOP***
STOP_PRESSED:
WAITSTART_2: ;очікування
sbic PIND, START ;натискання
rjmp WAITSTART_2 ; кнопки START
reti
;*** Затримка ***
DELAY: ldi r17, 128
d1: ldi r18, 100
d2: dec r18
brne d2
dec r17
brne d1
ret
```

Виконання роботи

Скласти програму мовою Асемблер для мікроконтролера Atmega 8515, що реалізує такі дії, як програма 15.1, але затримку організувати на таймері/лічильникові T/C1. Вважати, що частота тактового генератора вибрана 1024 кГц (1,024 мГц).

Час затримки для кожного варіанта вказаний в таблиці 15.2.

Завдання

1. Ознайомитися з призначенням системи переривань.

2. Ознайомитися з принципами програмування функцій з обробкою переривань.

3. Записати і налагодити в середовищі AVR Studio задану програму, завантажити її в стенд STK500 і виконати.

4. Скласти звіт про виконану роботу.

ПЕРЕЛІК ПИТАНЬ ДЛЯ ПІДСУМКОВОГО КОНТРОЛЮ ЗНАНЬ

1. Причини появи перших ЕОМ.
2. Приклади перших спеціалізованих комп'ютерів.
3. ЕОМ Z3, ЕОМ Colossus, Mark I та відзначити те нове, що принесли ці ЕОМ.
4. Призначення та спектр завдань ЕОМ першого покоління.
5. Наведіть приклади ЕОМ другого покоління.
6. Технології, що лягли основою ЕОМ третього покоління
7. Перші комп'ютери четвертого покоління.
8. Сформулюйте закон Мура.
9. Архітектура ЕОМ.
10. Основні засади архітектури фон Неймана.
11. Програмне керування роботою ЕОМ.
12. Наведіть приклад обладнання, яке не керується програмно.
13. Принцип збереженої програми.
14. Принцип адресації пам'яті
15. Машинне слово
16. Переваги використання двійкової системи обчислення.
17. Принцип ієрархічності запам’ятовуючих пристроїв
18. Швидкодія пам'яті
19. У чому відмінність довгострокової та оперативної пам'яті?
20. Технології реалізації пам'яті різного виду.
21. Тригер.
22. Чому налаштування комп'ютера не можна зберігати на вінчестері, а потрібен спеціальний вид довгострокової пам'яті?
23. Умовний перехід
24. Опишіть, як перші комп'ютери обходилися без умовного переходу.
25. Системна шина
26. Структура системної шини.
27. Схема типової ЕОМ, створеної на основі архітектури фон Неймана.

28. Назвіть переваги та недоліки гарвардської архітектури.

29. Наведіть випадки використання гарвардської архітектури на практиці.

30. Чому двійкова система обчислення є базовою для представлення даних в ЕОМ?

31. Складіть алгоритм переведення десяткового запису числа в двійковий.

32. Біт, байт, основні одиниці виміру.

33. Машинне слово. Наведіть приклади довжини машинного слова різних процесорів.

34. Пряме машинне подання цілого числа.

35. Знаковий розряд

36. Алгоритм побудови додаткового коду цілого числа.

37. Яким множникам при позначенні кількості даних відповідають приставки кіло-, мега-, гіга- та тера-?

38. У чому полягала перевага модульності з появою та розвитком комп'ютерів IBM PC?

39. Системна плата

40. Якими є типові обсяги оперативної пам'яті сучасних офісних ПК?

41. Навіщо використовується технологія DMA?

42. Визначення та опис поняття чіпсет

43. У чому відмінність вбудованого та окремого графічних адаптерів?

44. Назвіть переваги та недоліки механічних жорстких дисків.

45. Назвіть переваги та недоліки твердотільних накопичувачів?

46. Флеш-пам'ять і де вона використовується у сучасних комп'ютерах

47. Які дані з і якою метою зберігаються в пам'яті для налаштувань комп'ютера?

48. Історія терміну «вінчестер».

49. Назвіть різні типи зовнішніх портів.

50. Для яких цілей використовуються швидка та повільна системні шини?

51. Принцип зчитування та записування оптичних дисків.

52. Навіщо потрібна система охолодження комп'ютера? Які проблеми вона

вирішує?

53. Напівпровідник

54. Домішки, що легують

55. Фоторезист і навіщо він використовується

56. Монокристал

57. P-n-перехід і як він використовується в електротехніці

58. Як виготовляються кремнієві підкладки для інтегральних схем?

59. Фотолітографія.

60. Опишіть покроково процес виготовлення інтегральної схеми.

61. Технологія виробництва друкованих плат

62. Поясніть причину перегріву друкованих плат та спосіб подолання цієї проблеми.

63. Булеві функції.

64. Таблиця істинності

65. Система базових булевих функцій

66. Наведіть приклади вентилів.

67. Логічна схема

68. Чому для процесорів важливими є логіко-арифметичні операції?

69. Арифметико-логічний пристрій

70. Чому велика його роль пристрої процесора?

71. Багаторозрядний суматор.

72. Спекулятивні обчислення

73. АЛП і які операції воно може виконувати

74. Побітові логічні операції

75. Реалізація побітових логічних процедур

76. Побітове зрушення

77. Загальні характеристику процесора.

78. Тактовий імпульс

79. Внутрішня та зовнішня тактові частоти процесора

80. Які типові значення тактової частоти притаманні ЕОМ різних поколінь?

81. Які чинники не дозволяють необмежено збільшувати тактову частоту процесорів?

82. Розрядність процесора

83. Сумісність процесорів

84. Машинна мова процесора

85. У чому перевага багатоядерних процесорів порівняно з одноядерними?

86. CISC-архітектури.

87. Перерахуйте властивості машинних мов CISC-процесорів.

88. Назвіть переваги та недоліки CISC-архітектури.

89. RISC-архітектури.

90. Перерахуйте властивості машинних мов RISC-процесорів.

91. Конвеєр.

92. Перерахуйте види конфліктів конвеєра.

93. Позачергове виконання машинних команд

94. Технологія Hyper-Threading.

95. Кеш-пам'яті процесора.

96. Назвіть основні характеристики кеш-пам'яті.

97. Механізм відображення кеш-пам'яті

98. Навіщо потрібна ієрархічна кеш-пам'ять?

99. Фізична адреса

100. Пряма адресація

101. Адресний простір

102. Навіщо служить сегментна організація пам'яті процесу?

103. Віртуальна адреса

104. Хто виділяє оперативну пам'ять процесу?

105. Статична (динамічна) пам'ять програми

106. Особливості сегментної організації оперативної пам'яті процесу у разі багатопотокової програми.

107. Призначення операційної системи, її поняття, функції.

108. Драйвер пристрою, і навіщо він потрібний

109. Перерахуйте особливості ОС сімейства мейнфреймів IBM.

110. Перерахуйте особливості ОС сімейства Unix.

111. Назвіть причини появи ОС Linux.

112. Перерахуйте особливості операційної системи сімейства DOS.

113. Перерахуйте нововведення, запропоновані Windows.

114. Псевдопаралелізм

115. Процес і як процес співвідноситься з програмою?

116. Перерахуйте відомі вам файлові системи.

117. Наведіть приклади сучасних обчислювально-складних завдань.

118. Назвіть види паралельних обчислювальних систем із загальною пам'яттю.

119. Які можливості відкриває багатопотокове програмування?

120. Розподілені обчислення

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Азаров О. Д., Гарнага В. А., Клятченко Я. М., Тарасенко В. П. Комп'ютерна схемотехніка : підручник. Вінниця : ВНТУ, 2018. 230 с.
2. Вічужанін В. В. Цифрова схемотехніка : навч. посібник. Одеса : ОНПУ, 2018. 62 с.
3. Злобін Г. Г., Рикалюк Р. Є. Архітектура та апаратне забезпечення комп'ютерів : навч. посіб. Київ : Каравела. 2016. 224 с.
4. Комп'ютерна схемотехніка та архітектура комп'ютерів : навч.-метод. посібник / О. В. Задерейко та ін. ; Нац. ун-т «Одеська юридична академія». Одеса, 2022. 288 с. URL: https://dspace.onua.edu.ua/bitstreams/d5aaa5eb-1be9-4eed-b56c-9e5c3ca42baf/download
5. Комп'ютерна схемотехніка та архітектура комп'ютерів : навч. посіб. / О. В. Задерейко, Н. І. Логінова, О. Г. Трофименко та ін. Одеса : Фенікс, 2021. 163 с.
6. Комп'ютерна схемотехніка та архітектура комп'ютерів : навч. посіб. для підгот. здобув. вищої освіти галузі знань 12 «Інформаційні технології» / О. В. Задерейко та ін. ; Нац. ун-т «Одес. юрид. академія». Одеса : Фенікс, 2024. 251 с. https://doi.org/10.32837/11300.27830
7. Матвієнко М. П., Розен В. П, Закладний О. М. Архітектура комп'ютерів. Київ : Ліра-К, 2013. 264 с.
8. Минайленко Р. М., Конопліцька-Слободенюк О. К., Гермак В. С. Комп'ютерна схемотехніка : навч. посіб. Кропивницький : Видавець Лисенко В. Ф., 2022. 153 с.
9. Сенько В. І. Панасенко М. В., Сенько Є. В. Електроніка і мікропроцесорна техніка. Київ : Каравела, 2015. 676 с.

10. Сенько В. І., Панасенко М. В., Сенько Є. В. Електроніка і мікросхемотехніка. Том 3. Цифрові пристрої : підручник. Київ : Каравела, 2017. 400 с.
11. Цирульник С. М., Азаров О. Д., Крупельницький Л. В., Трояновська Т. І. Мікропроцесорна техніка : навчальний посібник. Вінниця : ВНТУ, 2017. 123 с.
12. Multisim Education. URL: https://www.ni.com/en/support/downl++oads/software-products/download.multisim.html
13. SIV - System Information Viewer. URL: http://rh-software.com
14. TOP500 Becomes a Petaflop Club for Supercomputers. URL: https://www.top500.org

Навчальне видання

КОМП’ЮТЕРНА СХЕМОТЕХНІКА ТА АРХІТЕКТУРА КОМП’ЮТЕРІВ

Методичні рекомендації

Укладач:

Жебко Олександр Олегович

Формат 60х84 1/16. Ум. друк. арк. 6.81.

Наклад 50 прим. Зам. № _____

Надруковано у видавничому відділі

Миколаївського національного аграрного університету

54020, м. Миколаїв, вул. Георгія Гонгадзе, 9

Свідоцтво суб’єкта видавничої справи ДК № 4490 від 20.02.2013