МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання лабораторних робіт здобувачами першого (бакалаврського) рівня вищої освіти всіх форм навчання за спеціальністю 141 «Електроенергетика, електротехніка та електромеханіка»
(освітні програми «Електричні та електронні апарати» та «Електромеханічне обладнання енергоємних виробництв»)
при вивченні навчальної дисципліни
«Основи мікропроцесорної техніки»

2024

Методичні рекомендації до виконання лабораторних робіт здобувачами першого (бакалаврського) рівня вищої освіти всіх форм навчання за спеціальністю 141 «Електроенергетика, електротехніка та електромеханіка» (освітні програми «Електричні та електронні апарати» та «Електромеханічне обладнання енергоємних виробництв») при вивченні навчальної дисципліни «Основи мікропроцесорної техніки» / Укл.: Л. Б. Жорняк. – Запоріжжя: НУ «Запорізька політехніка», 2024. – 90 с.

Укладачі: Л. Б. Жорняк, доцент, к.т.н.

Рецензент: В. В. Василевський, доцент, к.т.н.

Відповідальний за випуск: Р. Е. Мохнач, завідувач лабораторією кафедри ЕЕА

Затверджено
на засіданні кафедри
«Електричні та
електронні апарати»
Протокол № _7_
від «_26_» грудня 2023 р.

Затверджено УМПК ЕТФ
Протокол № _5_
від «_25_» січня 2024 р.

ЗМІСТ

ВСТУП

У системі роботи над сприйняттям і засвоєнням учнями нового матеріалу широко використовується метод лабораторних робіт.

Лабораторні роботи – це такий метод навчання, при якому учні під керівництвом викладача та за заздалегідь складеним планом проводять досліди (тестування) або виконують певні практичні завдання. І під час їх виконання учні можуть сприйняти і зрозуміти новий навчальний матеріал.

Для виконання лабораторних робіт необхідно розуміти нову навчальну програму та матеріал, що містить такі методичні прийоми:

– визначення теми і мети занять і завдань лабораторної роботи;

– короткі теоретичні відомості;

– визначення порядку виконання лабораторної роботи та її окремих етапів;

– безпосереднє виконання студентами лабораторних робіт і контроль викладача за ходом занять та контроль за технікою безпеки;

– підбиття підсумків лабораторної роботи та формулювання основних висновків.

Метою даних лабораторних робіт є формування у студентів навичок практичної роботи з мікропроцесором, закріплення та конкретизація набутих теоретичних знань, повна реалізація зв’язку теорії з практикою у навчанні.

Методичні рекомендації до організації та проведення лабораторних робіт розроблені відповідно до робочої програми освітньої складової ВК 08 «Основи мікропроцесорної техніки» та призначені для виконання студентами лабораторних робіт.

Лабораторні роботи з навчальної дисципліни спрямовані на засвоєння знань, оволодіння вміннями та навичками та формування елементів загальних компетентностей, передбачених робочою програмою з навчальної дисципліни. У результаті засвоєння навчальної дисципліни студент повинен

вміти:

– працювати з мікропроцесорними системами;

– програмувати мікропроцесорні системи;

знати:

– призначення, функції, характеристики та склад мікропроцесорних систем;

– системи команд, особливості організації системи переривань мікропроцесорних систем;

– організація пам'яті МП та доступ до неї.

Робоча програма передбачає виконання студентами практичних занять, у тому числі практичних занять як обов'язкової складової завдань з використанням персонального комп'ютера.

Основною функцією, яку виконують лабораторні роботи, є формування у студентів практичних навичок роботи з комп'ютерною технікою.

Аналізуючи зміст лабораторних робіт з основ мікропроцесорної техніки, легко помітити, що методи виступають як узагальнені навички. Для виконання будь-якої лабораторної роботи з основ мікропроцесорної техніки необхідно вивчити особливості функціонування мікропроцесора, провести розрахунки та проаналізувати режими роботи.

Планування лабораторної роботи здійснюється з використанням методичних рекомендацій до проведення лабораторних робіт з основ мікропроцесорної техніки.

Методичні рекомендації включають:

– тема лабораторної роботи з програми з дисципліни «Основи мікропроцесорної техніки»;

– мета лабораторної роботи (необхідно враховувати, що формулювання цілей часто нечіткі та не орієнтують студентів на конкретну діяльність);

– стислі теоретичні положення (у цій частині посібник до лабораторних робіт дублює зміст підручника);

– перелік обладнання та приладів для проведення лабораторних досліджень;

– принципова (установочна) схема лабораторного дослідження;

– порядок виконання, стислий опис способів діяльності учнів, форми представлення результатів вимірювань (таблиці, діаграми, графічні зображення);

– висновки з роботи;

– запитання для самоперевірки.

Така структура методичних рекомендацій сьогодні використовується в усіх типах навчальних закладів (училищах, технікумах, вищих навчальних закладах). При цьому, слід визнати, вона організовує діяльність учнів, але не розкриває логічної послідовності операцій і методів лабораторних досліджень.

На даний час студенти виконують лабораторні роботи на навчальному мікропроцесорному комплекті (УМПК-80) на базі мікропроцесора КР580ВМ80А (Intel 8080). Лабораторна установка складається з навчального мікропроцесорного комплекту (УМК), набору модулів, підключених до його системної шини, та різноманітних периферійних пристроїв. УМК представляє навчальний мікрокомп'ютер, призначений для навчання програмуванню, конструюванню та конфігуруванню мікропроцесорних пристроїв і систем, виготовлений на МП КР580ВМ80. Розвиток інформаційних технологій призвів до появи концепції «Віртуальна лабораторна майстерня» (ВЛМ), в основі якої лежить комп'ютерне моделювання. Основні шляхи використання ВЛМ у навчальному процесі:

– як комп'ютерний «тренажер» для підготовки до практичної роботи в реальній лабораторії (при цьому програми комп'ютерних і фізичних експериментів, як правило, збігаються);

– як додаток до реальної майстерні, що передбачає проведення таких комп'ютерних експериментів, які з різних причин (технічних, фінансових, організаційних тощо) не можуть бути реалізовані на фізичному обладнанні.

Використання ВЛМ як комп'ютерного «тренажера» дозволяє краще підготуватися студенту до проведення фізичного

експерименту, краще зрозуміти досліджувані ефекти та набути навичок роботи з вимірювальними приладами (якщо віртуальний практикум містить комп’ютерні моделі вимірювань). інструменти, подібні за своїми властивостями до властивостей реальних інструментів). Як правило, такий підхід можна рекомендувати студентам дистанційної форми навчання, оскільки він не тільки сприяє кращому засвоєнню матеріалу, що вивчається, але й дозволяє скоротити тривалість занять у реальних лабораторіях під час перебування в стінах навчального закладу. .

Розвиток комп'ютерних мережевих технологій призвів до появи лабораторного практикуму, реалізованого в режимі віддаленого доступу до будь-якого обладнання. Враховуючи, що забезпечення віддаленого доступу до будь-якого обладнаного приміщення пов’язане з багатьма проблемами (необхідність підключення прототипу лабораторії до ПК, для надійного захисту від наслідків аварійних ситуацій, низька ефективність використання обладнання через неможливість у ряді випадків реалізації колективного доступу), інтелектуальні технології мають достатню кількість таємниць. Однак він теж має право на існування і в деяких випадках має очевидні переваги перед ВЛМ.

Важливим є питання про те, чи є ВЛМ альтернативою реальній лабораторній практиці. З одного боку, сучасні технології комп’ютерного моделювання дозволяють створювати віртуальні інтерфейси реального лабораторного обладнання, які відтворюють як зовнішній вигляд, так і його параметри з дуже малими відхиленнями. З іншого боку, утримання та своєчасне оновлення лабораторного обладнання, в тому числі засобів вимірювальної техніки, потребує значних фінансових ресурсів. Проте будь-яка, навіть найякісніша, ВЛВ в більшості випадків не замінює вчителя, який має роботу з реальним обладнанням.

Одним із таких ВЛМ є програмна модель емулятора стенду УМПК-80, для якої розроблено теми лабораторних робіт.

Критерії оцінювання лабораторних робіт: при оцінюванні набутих умінь і навичок при виконанні практичних робіт використовується шкала оцінок «склав/не склав».

Оцінювання практичних/лабораторних робіт здійснюється за такими правилами:

– положення про постійний контроль успішності та проміжну атестацію студентів;

– положення про планування, організацію та проведення лабораторних досліджень і практичних занять.

1 ЛАБОРАТОРНА РОБОТА №1 НАВЧАЛЬНИЙ МІКРОПРОЦЕСОРНИЙ КОМПЛЕКТ УМПК-80

Мета роботи – Вивчення будови навчального мікропроцесорного комплекту УМПК-1 і органів його керування, уведення, індикації.

1.1 Призначення та характеристики комплекту УМПК–80

УМПК являє собою закінчену мікроЕОМ і призначений для підготовки фахівців в галузі мікропроцесорної техніки шляхом вивчення структур й основ програмування мікропроцесора КР580ВМ80А, УМПК може бути використаний у якості керуючої ЕОМ при створенні і дослідженні роботи систем керування електричними апаратами і процесами. Він є легко освоюваним і зручним засобом для налагодження невеликих (до 2 Кбайт) програм користувача. Засоби індикації на чільній панелі дозволяють спостерігати процеси перетворення і передачі інформації під час роботи УМПК. Технічні характеристики УМПК-1 представлені в таблиці 1.1

Таблиця 1.1–Технічні характеристики УМПК-1

Характеристика	Параметр
Тип застосовуваного мікропроцесора	КР580ВМ80А
Об’єм оперативного запам'ятовуючого пристрою	2Кбайта
Об’єм постійного запам'ятовуючого пристрою	2Кбайта
Можливість переривання	1 вектор
Програмне забезпечення	Системна програма "Монітор"
Напруга живлення	220 В ± 22 В частотою 50Гц ± 1 Гц
Рівні вхідних і вихідних сигналів сумісні з рівнями ТТЛ ІС	

1.2 Комплектація УМПК–80

УМПК складається з таких складових частин (рисунок.1.1):
– мікроЕОМ;
– пульт оператора;
– блок живлення.

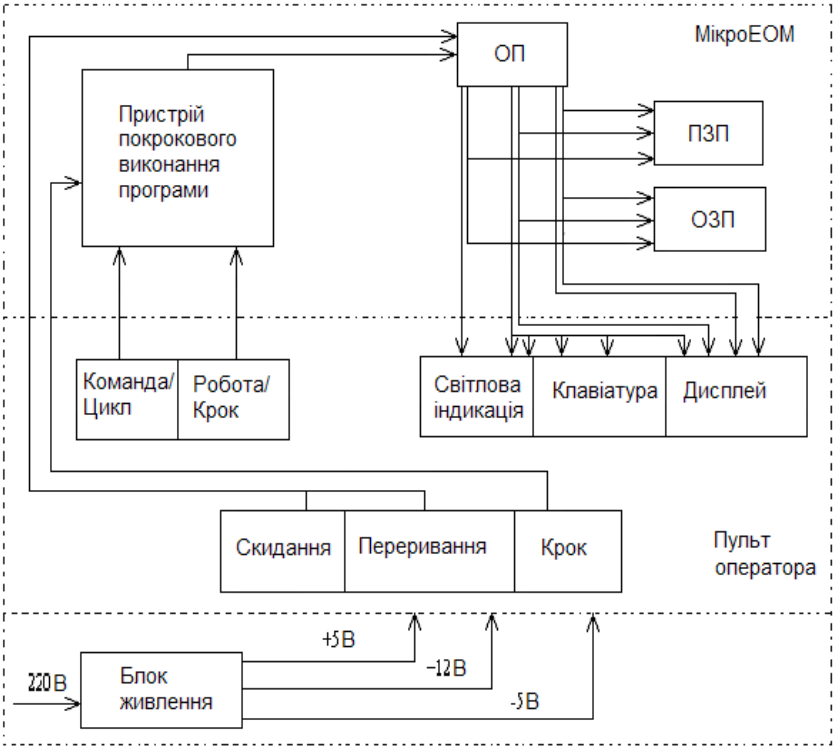


Рисунок 1.1 – Структура УМПК

МікроЕОМ є основною частиною і керує роботою усього УМПК, здійснює звертання до пам'яті, уведення, виведення і індикацію інформації. Основою мікроЕОМ є операційний пристрій ОП, що складається з мікропроцесора (МП) і генератора тактової частоти (ГТЧ).

Мікропроцесор КР580ВМ80А призначений для виконання визначеного набору команд і реалізований на одній ВІС, що включає акумулятор і кілька регістрів загального призначення (РЗП).

Регістром будемо називати мінімальний адресуючий запам'ятовуючий елемент, що складається з 8 розрядів (бітів). Інформація, яка зберігається в такому регістрі, називається словом. Регістри оперативного запам'ятовуючого пристрою (ОЗП) звичайно називають комірками, кожному з який приписаний деякий номер-адреса. Ці регістри використовуються для збереження даних і команд користувача і виконані у виді окремих ВІС. Мікропроцесор може звертатися до них за допомогою спеціальних команд. Акумулятор (А) – найважливіший регістр МП. Більшість команд використовує акумулятор для збереження вихідних даних і результатів операції. Робота УМПК полягає в обміні даними між регістрами й описується системою команд, кожна з яких визначає конкретний вид обміну і перетворення даних. Тому для розробки програми мовою машинних команд необхідно знати склад і призначення регістрів УМПК і систему команд МП КР580.

1.2.1. Архітектура пам'яті УМПК

Розглянемо структуру доступних користувачу регістрів УМПК на базі МП КР580ВМ80А (рисунок 1.2). Адресація всіх регістрів усередині УМПК здійснюється в двійковій системі числення, у той час як користувачу зручно для скорочення запису користатися шістнадцятковою системою. У подальшому для позначення адрес і вмісту регістрів будемо використовувати шістнадцяткову систему. Розмір слова МП КР580 складає 8 біт (1 байт), тому розрядність усіх наведених регістрів, крім покажчика стеку SP (Stack Pointer) і лічильника команд PC (Program Counter) також складає 1 байт. МП КР580 має один акумулятор А і шість РЗП: B,C,D,E,H,L. РЗП можуть поєднуватися в двобайтові регістри BC,DE,HL. У цьому випадку в регістрах B,D,H знаходиться старший, а в регістрах C,E,L – молодший байт шістнадцятирозрядного слова. МП містить також регістр прапорів умов F, у якому використовуються 5 розрядів з 8.

Виконання програм вимагає, як правило, звертання до основної пам'яті і буферних регістрів пристрою вводу-виводу (ПВВ). Призначенням буферних регістрів є узгодження характеристик МП

і зовнішніх пристроїв користувача (принтер, дисплей, датчики, вимірювальні прилади та інші).

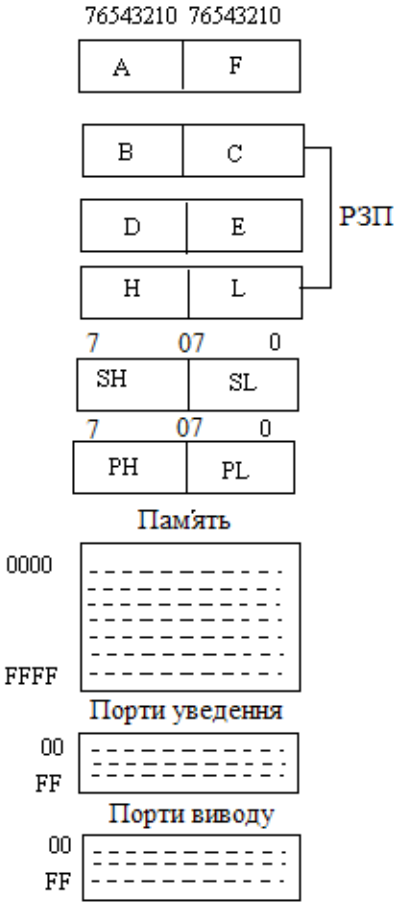


Рисунок 1.2 – Структура регістрів мікропроцесора

Звернення до усіх цих регістрів викликано тим, що велика кількість даних не вміщується в РЗП і програма повинна мати методи зв'язку з «зовнішнім світом» для одержання та видачі інформації. Тому необхідно уявляти собі доступний об’єм пам'яті та регістрів ПВВ.

Адресний простір (поле) пам'яті - це сукупність комірок пам'яті, до яких може адресуватися МП. Буферний регістр адреси МП КР580 – шістнадцятирозрядний, тому поле пам'яті містить 2^{16} = 65536 комірок (64 Кбайт, тому що 1 Кбайт = 1024 байт) ємністю 1 байт кожний з відповідною адресою від 0000 до FFFF. Реальна мікроЕОМ може використовувати не усе адресне поле, а тільки його частину. Сукупність комірок пам'яті, що фактично існують в даній мікроЕОМ, утворюють робочий (фізичний) простір (поле) пам'яті. В УМПК робочий простір пам'яті складає 4 Кбайти, з яких 2 Кбайти займає ПЗП і 2 Кбайти – ОЗП. З ПЗП інформація може тільки зчитуватися, а ОЗП дозволяє як зчитувати зміст його комірок, так і записувати в них нові дані.

Адресний простір ПВВ дозволяє мікроЕОМ мати до 2^8 = 256 пристроїв (портів) уведення і виводу, адресованих роздільно. Адреси портів можуть знаходитися в межах 00 - FF.

В УМПК 1 Кбайт ПЗП (адреси 0000 – 03FF) займає програма “Монітор” і 1 Кбайт ПЗП (адреси 0400 – 07FF) наданий у розпорядження користувача. ОЗП використовується для збереження програм, що змінюються, і даних користувача.

Останні 54 елементи ОЗП (адреси FFCA – FFFF) зайняті програмою «Монітор» для збереження оперативних даних і не повинні використовуватися оператором. Адреса 0800 є початковою, з якої починаються всі програми користувача.

1.2.2. Операційний блок

Операційний пристрій проводить всі операції по переробці інформації. Його вихідним станом є читання інформації з нульової адреси ПЗП кожен раз після натискання керуючої кнопки <СБРОС> на пульті. При цьому викликається програма «Монітор», що забезпечує уведення інформації з клавіатури і виведення її на індикатор. Інформація про стан ОП фіксується в регістрі стану на початку кожного машинного циклу. У таблиці 1.2 наведені можливі стани ОП.

Стану 0 у таблиці відповідає низький рівень потенціалу, а стану 1 – високий. У залежності від стану цього регістра формуються сигнали, що керують роботою усієї мікроЕОМ.

У таблиці 1.2 надане визначення кожного біта регістра стану. Риска над керуючим сигналом WO у таблицях 1.2 і 1.3 вказує, що активним станом сигналу є логічний 0.

Пристрій покрокового виконання програми переводить ОП в стан «очікування» після виконання чергового кроку.

Пульт оператора призначений для взаємодії оператора з мікроЕОМ. Органи керування, уведення й індикації УМПК розташовані на чільній панелі й описані нижче. Блок живлення забезпечує постійними стабілізованими напругами мікроЕОМ і пульт оператора.

Таблиця 1.2 – Можливі стани ОП

Стан ОП	Розряди регістра стану ОП							
	D7 MEMR	D6 INP	D5 M1	D4 OUT	D3 HLTA	D2 STACK	D1 WO	D0 INTA
Вибір команди	1	0	1	0	0	0	1	0
Читання пам'яті	1	0	0	0	0	0	1	0
Запис до пам'яті	0	0	0	0	0	0	0	0
Читання стеку	1	0	0	0	0	1	1	0
Запис до стеку	0	0	0	0	0	1	0	0
Ввод	0	1	0	0	0	0	1	0
Вивод	0	0	0	1	0	0	0	0
Переривання	0	0	1	0	0	0	1	1
Зупин	1	0	0	0	1	0	1	0
Переривання при зупину	0	0	1	0	1	0	1	1

Таблиця 1.3 – Біти регістру стану ОП

Значення сигналу	Розряд регістра стану	Пояснення
INTA	D0	Сигнал підтвердження запиту переривання. Використовується для уведення на шину даних команди RST.
WO	D1	Вказує на то, що в даному циклі виконується запис до пам'яті або операція виводу.
STACK	D2	Означає наявність на шині адреси вмісту покажчика стеку.
HLTA	D3	Сигнал підтвердження команди HLT
OUT	D4	Вказує на то, що в даному циклі виконується операція виводу.
M1	D5	Вказує на то, що даний цикл використовується для вибірки першого байту команди.
INP	D6	Вказує на то, що в даному циклі виконується команда уведення.
MEMR	D7	Вказує на те, що в даному циклі буде проведене читання пам'яті.

1.3 Пристрої керування, введення та індикації набору УМПК

На передній панелі УМПК розташовано:

– кнопка вмикання/вимикання;
– кнопки скидання та переривання;
– кнопки керування покроковим режимом роботи;
– функціональна клавіатура;
– клавіатура вводу даних;
– шестисегментний дисплей;
– світлодіоди індикації стану шини даних, шини адреси, керуючих сигналів МП;
– роз'єм для приєднання макетного ТЕЗ.

Кнопка вмикання/вимикання УМПК (символ "~") розташована в лівій нижній частині чільної панелі. Натиснута кнопка відповідає увімкненому стану УМПК, віджата - вимкненому. Над кнопкою розміщені три світлодіоди: + 5 В; - 5 В; + 12 В. При перевантаженнях спрацьовує захист блоку живлення і загоряється від-

повідний світлодіод. У цьому випадку необхідно вимкнути УМПК. Кнопка скидання (<СБ>) служить для ініціалізації системної програми "Монітор". Після натискання кнопки <СБ> здійснюється запуск цієї програми й у лівій позиції дисплея з'являється символ " - " це означає, що УМПК готовий до прийому команд.

Кнопка переривання (<ПР>) розташована під кнопкою <СБ>. При натисканні цієї кнопки виробляється сигнал **ЗАПИТ НА ПЕРЕРИВАННЯ** сьомого рівня і, якщо переривання дозволені (виконана команда EI – Enabled Interrupt),буде закінчене виконання поточної команди, після чого керування передається на адресу 38H (відповідає сьомому рівню переривань). Починаючи з цієї адреси розміщується програма обробки переривань. Якщо кнопка <ПР> натиснута під час роботи програми «Монітор», то на дисплеї виводиться символ « ? ». В іншому випадку на дисплей виводиться адреса точки переривання й керування передається програмі "Монітор".

Керування покроковим режимом роботи УМПК виробляється за допомогою кнопок РАБОТА / ШАГ (<РБ/ШГ>), КОМАНДА / ЦИКЛ (<КМ / ЦК>), ШАГ (<ШГ>). Цими кнопками встановлюється один із двох режимів покрокової роботи. Перший режим - командний. Для встановлення цього режиму слід натиснути кнопку <РБ / ШГ>. Кожне натискання кнопки <ШГ> викликає виконання поточної команди. При цьому на світлодіодах індикації стану шин даних, адреси й керуючих сигналів будуть висвітлюватися в двійковому коді, відповідно, адреса і код виконуваної команди, а також керуючі сигнали МП. Другий режим покрокової роботи - робота за командними циклами. Для встановлення цього режиму слід нажати кнопки <РБ/ШГ> і <КМ/ЦК>. У цьому випадку можна простежити хід виконання команди. При кожнім натисканні кнопки <ШГ> буде виконаний наступний машинний цикл. На світлодіодах індикації буде відбиватися інформація, що відповідає кожному машинному циклу. Для роботи в автоматичному режимі обидві клавіші <РБ/ШГ> і <КМ/ЦК> повинні бути віджаті.

Клавіатура УМПК розділена на дві частини. У лівій частині розташовані функціональні клавіші. За кожною клавішею закріплена визначена функція програми "Монітор".

Призначення функціональних клавіш:

– <П> – перегляд і модифікація вмісту комірок пам'яті;

– <РГ> – перегляд і модифікація вмісту регістрів МП;

– <СТ> – старт програми;
– <КС> – підрахунок контрольної суми;
– <ЗК> – заповнення масиву пам'яті константою;
– <ПМ> – переміщування масиву пам'яті;
– <–> – роздільник;
– <ВП> – виконати.

Права частина клавіш призначена для уведення параметрів у шістнадцятковій системі. На ці клавіші нанесені символи: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. На клавіші 4,5,6,7,8,9 нанесені позначення ідентифікаторів регістрів МП:

– <РН> – старший байт лічильника команд;
– <PL> – молодший байт лічильника команд;
– <SH> – старший байт покажчика стеку;
– <SL> – молодший байт покажчика стеку.

Для ідентифікації інших регістрів МП використовуються клавіші:

– <A> – регістр A;
– <B> – регістр B;
– <C> – регістр C;
– <D> – регістр D;
– <E> – регістр E;
– <F> – регістр F.

Шестисегментний дисплей призначений для відображення даних у шістнадцятковій системі. Ліві чотири сегменти використовуються для відображення адреси й ідентифікаторів регістрів МП, а праві два для відображення даних. Індикація стану шин адреси, даних і керуючих сигналів МП здійснюється за допомогою світлодіодів. Шина адреси ідентифікується шістнадцятьма світлодіодами, шина даних – за допомогою восьми світлодіодів (відображення в двійковій системі). Ці світлодіоди позначені, відповідно, <АДРЕСА> і <ДАНІ>.

Під кожним світлодіодом зазначені номери розрядів, яким вони відповідають. Світлодіди керуючих сигналів МП позначені написом <СОСТОЯНИЕ> і кожен розряд має своє призначення.

1.4 Підготовка УМПК до роботи

Установіть кнопку вмикання УМПК у віджатий стан.

Підключить УМПК до мережі змінного струму напругою 220 В, частотою 50 Гц.

– кнопки <РБ / ШГ> і <КМ / ЦК> установіть у натиснутий стан;

– включить УМПК кнопкою вмикання;

– при цьому не повинні загорятися світлодіоди захисту + 5В, - 5 В, + 12 В;

– натисніть кнопку <СБ> і в крайньому лівому розряді дисплея повинний з'явитися знак " – ".

Після цього УМПК готовий до прийому команд.

Повторне вмикання УМПК слід робити не раніше, ніж через 20 с після вимикання. У протилежному випадку спрацює захист блоку живлення і загоряться відповідно світлодіоди захисту. У цьому випадку необхідно вимкнути УМПК і дочекатися, коли світлодіоди захисту згаснуть, після чого повторно вмикнути УМПК.

1.5 Завдання

1.5.1 Вивчити будову, основні технічні характеристики та органи управління УМПК.

1.5.2 Виконувати основні операції з клавіатурою, встановлення різних режимів та використання індикації.

1.6 Методичні рекомендації

1.6.1 Визначити місце основних органів керування на УМПК, їхнє призначення.

1.6.2 З'ясувати місце і призначення індикаторів, а також засобів приєднання зовнішніх пристроїв.

1.6.3 Вивчити призначення основних вузлів УМПК, та їх характеристики.

1.6.4 Освоїти практично порядок вмикання та вимикання УМПК з обов'язковою витримкою часу до 20 секунд (момент згасання всіх трьох світлодіодів зліва на чільній панелі УМПК) перед наступним вмиканням.

1.6.5 Освоїти порядок виклику вмісту комірок пам'яті, регістрів, перегляду вмісту пам'яті, а також регістрів.

1.6.6 Освоїти правильні прийоми роботи з клавіатурою. Натискання окремих клавіш робити вертикально, з мінімальним зусиллям кисті (не руки), як це прийнято на електричних і електронних друкарських машинках.

1.6.7 З'ясувати можливості встановлення покрокових режимів при виконанні програм, роль лінійних індикаторів.

1.7 Зміст звіту

Звіт повинен містити:

- назву та мету роботи;
- стислі відповіді на контрольні запитання;
- висновки.

1.8 Питання для самоперевірки

1.8.1 Основні характеристики УМПК, його функціональні можливості.

1.8.2 Структура УМПК, основні вузли, їхні характеристики.

1.8.3 Характеристика й організація пам'яті.

1.8.4 Регістрова структура мікропроцесора.

1.8.5 Характеристика можливих станів операційного пристрою.

1.8.6 Характеристика органів керування та індикації УМПК.

1.8.7 Основні вимоги до роботи з УМПК.

2 ЛАБОРАТОРНА РОБОТА № 2 ВИВЧЕННЯ ІНТЕРФЕЙСУ ТА РЕЖИМУ РОБОТИ ПРОГРАМНОГО СТЕНДУ УМПК

Метою роботи є вивчення інтерфейсу програмної моделі стенду УМПК-80, а також можливостей програмного емулятора системи «Монітор» та набуття навичок роботи з основними командами МП на базі К580.

2.1 Загальні відомості про роботу з емулятором К580

Для виконання запропонованих програм можна використовувати емулятор К580, що знаходиться в ауд. 226 а та від адміністратора системи на сервері або в мережі Інтернет для самостійної роботи студентів при вивченні даної навчальної дисципліни знаходяться в репозитарії НУ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА» за посиланням http://eir.zp.edu.ua/handle/123456789/488 а також в системах дистанційного навчання за посиланням https://moodle.zp.edu.ua/course/view.php?id=2089. Це програмна модель (емулятор) лабораторного стенда навчального мікропроцесорного комплекту НМПК-80 (мікроЕОМ із базовим мікропроцесором типу К580, U880, Intel 8080, 8085, Z80 і вбудованим дисплеєм. У комплекті УМПК програма вбудована в постійний запам'ятовуючий пристрій (ПЗП), котру можна використовувати в навчальному процесі для вивчення різноманітних мікро-ЕОМ разом з УМПК, що дозволяє студентам підготувати лабораторні роботи вдома за допомогою обчислювальних пристроїв.

Ця модель була розроблена для можливості скорочення студентами часу на рутинні операції (ассемблювання, дізассемблювання і т.і.). Вона дає більш широкі та зручні можливості для набору та відпрацювання програм (наприклад, може бути можливий одночасна перевірка всіх регістрів, комірок пам'яті, ввід команд в мнемонічних позначках (операторах) та в шістнадцятирічних кодах, ассемблювання та дізасемблювання команд і т.і.).

Програмна модель стенду УМПК-80 є засобом, який надає значно ширші та зручніші можливості для набору та налагодження програм (наприклад, можна одночасно переглядати всі регістри, пам'ять, вводити команди в мнемосхемі та в шістнадцятковому коді, монтаж і демонтаж інструкції тощо).

Отримання компетенцій у програмі варто починати з вивчення робочого вікна. Для цього відкрийте на комп'ютері файл із програмною моделлю стінок та УМПК-80. На екрані монітора з'явиться діалогове вікно програми (рисунок 2.1).

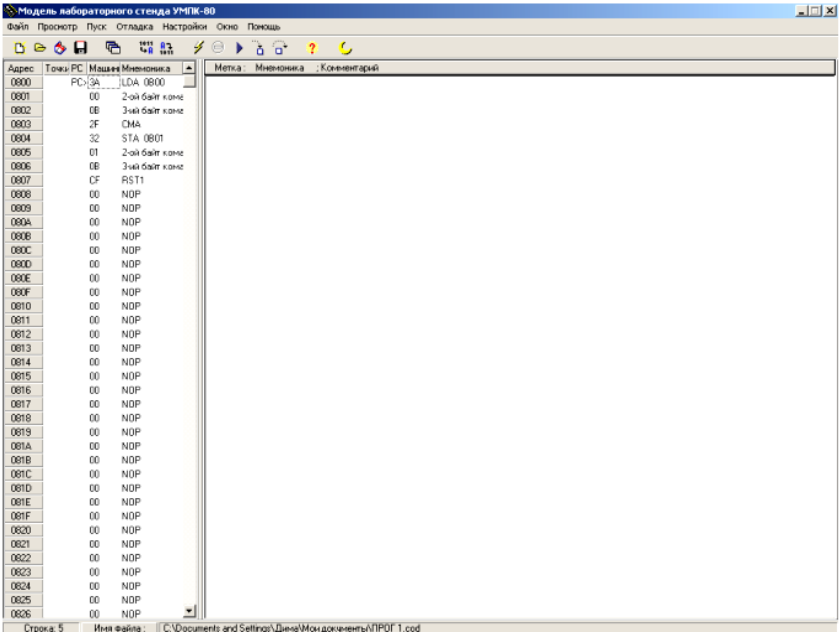


Рисунок 2.1 – Вигляд екрана комп'ютера при роботі з програмною моделлю стенду УМПК-80. Головне вікно моделі емулятора набору К580

Меню моделі розташоване у верхній частині головного вікна моделі. Він містить усі команди, необхідні для роботи з моделлю (рисунок 2.2).

Рисунок 2.2 – Вигляд меню моделі

Він включає:

– підменю «Файл» містить команди для роботи з файлами. Підменю «Файл» знаходиться в головному меню моделі і викликається комбінацією клавіш «Alt+F». Він містить команди для роботи з файлами (таблиця 2.1);

– Підменю «Перегляд», містить команди для керування вихідною інформацією. Підменю «Перегляд» знаходиться в головному меню моделі і називається комбінацією клавіш «Alt+П». Містить

команди для роботи з візуальними об’єктами моделі (вікна, панелі тощо) (табл. 2.2);

– підменю «Пуск» містить команди для виконання та обробки досліджуваної програми. Підменю «Пуск» знаходиться в головному меню моделі і викликається комбінацією клавіш «Alt+У». Містить команди, необхідні для запуску API в автоматичному режимі, а також його трансляцію в різні формати коду (шістнадцяткові або мнемонічні коди);

– підменю «Отладка» містить команди для налагодження досліджуваної програми. Підменю «Отладка» знаходиться в головному меню моделі і називається комбінацією клавіш «Alt+О». Він містить команди, необхідні для налагодження API в покрокових режимах;

– пункт «Настройки (Настройки)» відкриває вікно налаштувань моделі. Команда «Настройки (Настройки)» знаходиться в головному меню моделі і називається комбінацією клавіш «Alt+Н». Ця команда викликає вікно налаштувань моделі.

Панель інструментів (рисунок 2.3) призначена для швидкого доступу до найбільш часто використовуваних команд меню і містить:

– створення нового файлу;
– відкриття файлу;
– закриття файлу;
– збереження файлу на диск;
– відкриття всіх вікон;
– розбирання досліджуваної програми;
– складання досліджуваної програми;
– скидання процесора;
– зупинити виконання досліджуваної програми;
– виконання досліджуваної програми в автоматичному режимі
– виконання досліджуваної програми за циклами команд;
– виконання досліджуваної програми в машинних циклах;
– викликати довідкову інформацію;
– вихід з моделі.

Таблиця 2.1 – Таблиця команд підменю «Файл»

Команда	Опис дій при виборі команди
Новий Новый	Поточна програма закрита. Якщо в не було збережених даних, пропонується їх зберегти. Створена порожня програма.
Відкрити Открыть	Поточна програма закрита. Якщо там не було збережених даних, пропонується зберегти їх. На диску відкривається файл із попередньо збереженою програмою
Знову відкрити Открыть вновь	Модель може запам'ятати до 10 назв останніх відкритих файлів. У цьому випадку за командою «Відкрити знову» відкривається їх список, з якого можна вибрати потрібний. Після цього відбувається наступне: поточна програма закривається, і якщо в ній не було збережених даних, то пропонується їх записати; вибраний файл відкривається, якщо його знайдено зберегти
Зберегти Сохранить	Поточна програма зберігається у файл на диску. Якщо програма раніше не була записана, вам буде запропоновано ввести назву файлу
Зберегти як *.asm Сохранить как *.asm	Мнемонічні коди поточної програми зберігаються у файлі asm
Зберегти як *.cod Сохранить как *.cod	Зберігати машинні коди у файлі cod
Закрити Закрыть	Поточна програма закрита. Якщо там не було збережених даних, пропонується записати їх
Вихід Выход	Поточна програма закрита. Якщо в системі не було збережених даних, то пропонується їх зберегти. Модель завершує свою роботу

Панель інструментів можна відключити (підключити), вибравши в головному меню програми «Вид / Панель инструментов (Просмотр / Панель инструментов)». Якщо праворуч від команди встановлено позначку, панель інструментів стає видимою:

Рисунок 2.3 – Панель інструментів

Таблиця 2.2 – Таблиця команд підменю «Перегляд»

Команда	Опис дій при виборі команди
Реєстри та прапори Регистры и флаги	Регистры и флаги Відкриється вікно та активує «Реєстри и флаги (Регистры и флаги)»
Стенд RAM ОЗУ стенда	Відкривається вікно і активується «Стенд RAM (ОЗУ стенда)»
Стек	Відкриється вікно та активує «Стек»
Порти s Порты	ідкриється вікно та активує «Порти (Порты)»
Підставка Інструменти Средства стенда	Відкриється вікно та активує «Інструменти стенда (Средства стенда)»
Все	Відкриваються та стають активними всі вікна: «Реєстри та прапорці (Регистры и флаги)», «Стенд RAM ОЗУ стенда)», «Стек («Стек)», «Порти (Порты)», «Стенд засоби «Средства стенда)»
Дисплеї панелі інструментів Панель инструментов	Панель інструментів Відображає (приховує) панель інструментів. З галочкою праворуч від панелі інструментів команд видно
Декодування кодів Дешифрация кодов	Якщо справа від команди стоїть галочка, то в розділі машинних кодів шістнадцяткові коди розбираються, починаючи з адреси в лічильнику команд
Перегляд повідомлень Просмотр сообщений	Якщо праворуч від команди є галочка, розділ повідомлень видно

Розділ машинного коду головного вікна моделі призначений для введення та редагування поточних програм у машинних кодах і являє собою таблицю, кожен рядок якої відповідає одній комірці ОЗУ. Стовпці таблиці мають таке призначення (таблиця 2.3)

Щоб ввести значення в клітинку, використовуйте клавіші керування або наведіть на неї курсор (пунктирну рамку) мишкою та введіть нове значення, після чого підтвердіть введення клавішею «enter». Слід зазначити, що завжди потрібно вводити обидві цифри номера, тобто для введення значення, наприклад, «7» або «F», потрібно набрати відповідно «07», «0F».

Таблиця 2.3 – Таблиця розділу машинного коду

Заголовок стовпця	Призначення стовпця
Адреса	Містить адреси комірок пам'яті
Точки зупину	Відображає встановлені точки зупину
PC	Відображає лічильник програм у вікні "PC"
Машинний код	Містить значення комірки пам'яті, адресу, вказану в стовпці «адреса»
Мнемоніка	Містить мнемонічний опис значень комірок

Розділ машинного коду має локальне меню – меню розділу машинного коду, яке викликається клацанням правої кнопки миші.

Додаткове вікно «Регістри і прапорці (Регистры и флаги)» викликається командою «перегляд / регістри і прапорці (просмотр / регистры и флаги)» в головному меню моделі або комбінацією клавіш «Ctrl+R». Він містить регістри мікропроцесора К580ВМ80. Вікно розділене на три частини – секції: «реєстри символів («регистр признаков)», «індикатори (указатели)», «Регістри RGP (регистры РОН)» (рисунок 2.4).

Переміщення по розділах та їх елементах здійснюється за допомогою клавіш «Tab» (вперед) і «Shift+Tab» (назад) або за допомогою миші. Якщо ви наведете вказівник миші на поле під час введення та утримаєте його протягом кількох секунд, спливаюча підказка, що з’явиться, міститиме десяткове та двійкове представлення значення цього поля.

Вікно «ОЗУ стенда» викликається командою «Просмотр / ОЗУ стенда» головного меню моделі або комбінацією клавіш «Ctrl+M». У ньому можна переглянути будь-який фрагмент пам'яті. Вміст пам'яті відображається у вигляді таблиці, кожен рядок якої показує шістнадцять байт, тому адреса комірки, що лежить на перетині рядка «0820» і стовпця «D», дорівнює «082D» (рисунок 2.5). Основною функцією вікна є перегляд пам’яті, проте, коли Ви можете редагувати значення будь-якої клітинки, якщо хочете. Для цього за допомогою клавіш управління курсором або миші перемістіть курсор (пунктирну рамку) на комірку пам’яті, значення якої потрібно виправити, і натисніть клавішу «введення» або двічі кла-

цніть по ній лівою кнопкою миші. При цьому активується редактор RAM, в якому потрібно ввести нове значення комірки, після чого редактор необхідно закрити.

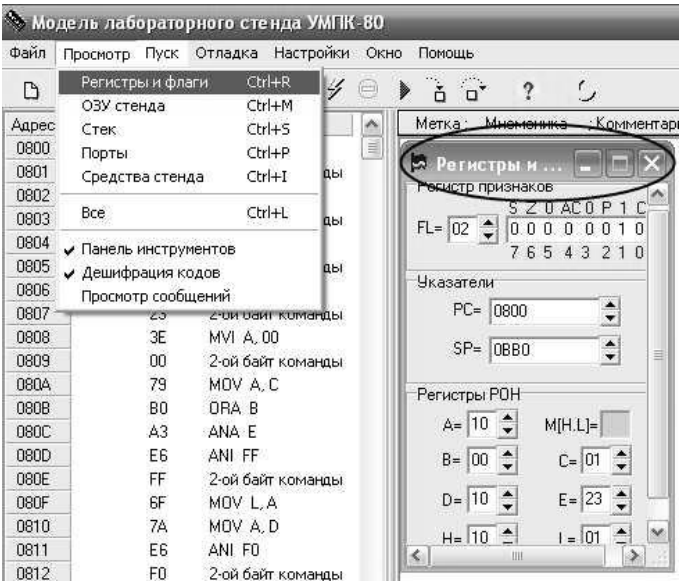


Рисунок 2.4 – Перевірка вмісту регістрів

Вікно «Редактор ОЗП» викликається з вікон «Редактор ОЗУ» і «Стек», оскільки їх основна функція – перегляд пам’яті. Редактор призначений для редагування пам'яті. Редагування виконується по одному байту. Адреса поточної комірки відображається зліва від напису «Адреса ОЗУ». Редактор має чотири кнопки (таблиця 2.4).

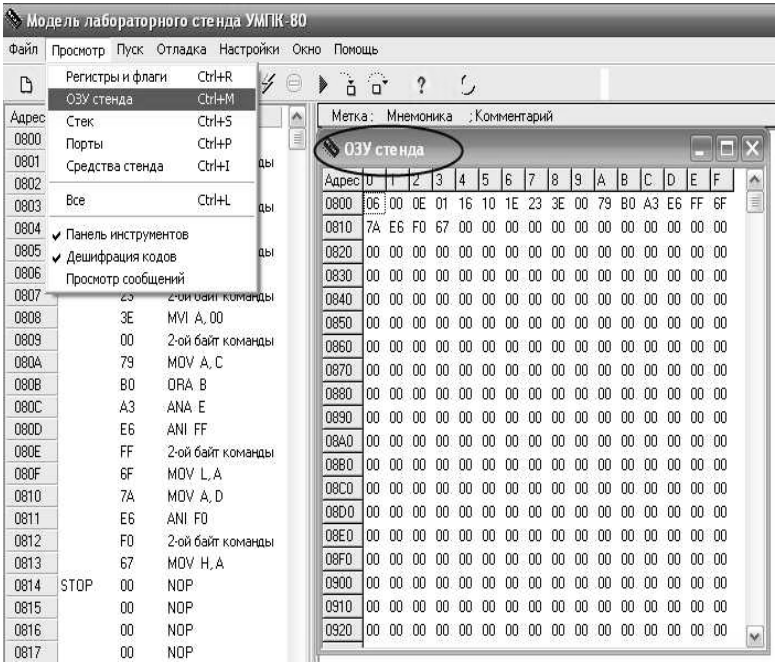


Рисунок 2.5 – Перевірка вмісту комірок пам’яті

Таблиця 2.4 – Таблиця команд вікна редактора ОЗП

Кнопка	Опис дій при натисканні кнопки
Запис+масштаб. Зап.+Увел.	Редаговане значення клітинки записується в поточну адресу, після чого поточна адреса збільшується на одиницю
Вперед	Поточна адреса збільшується на одиницю без запису нового значення клітинки
Назад	Поточна адреса зменшується на одиницю без запису нового значення клітинки
Закрити Закрыть	Редактор ОЗП закривається

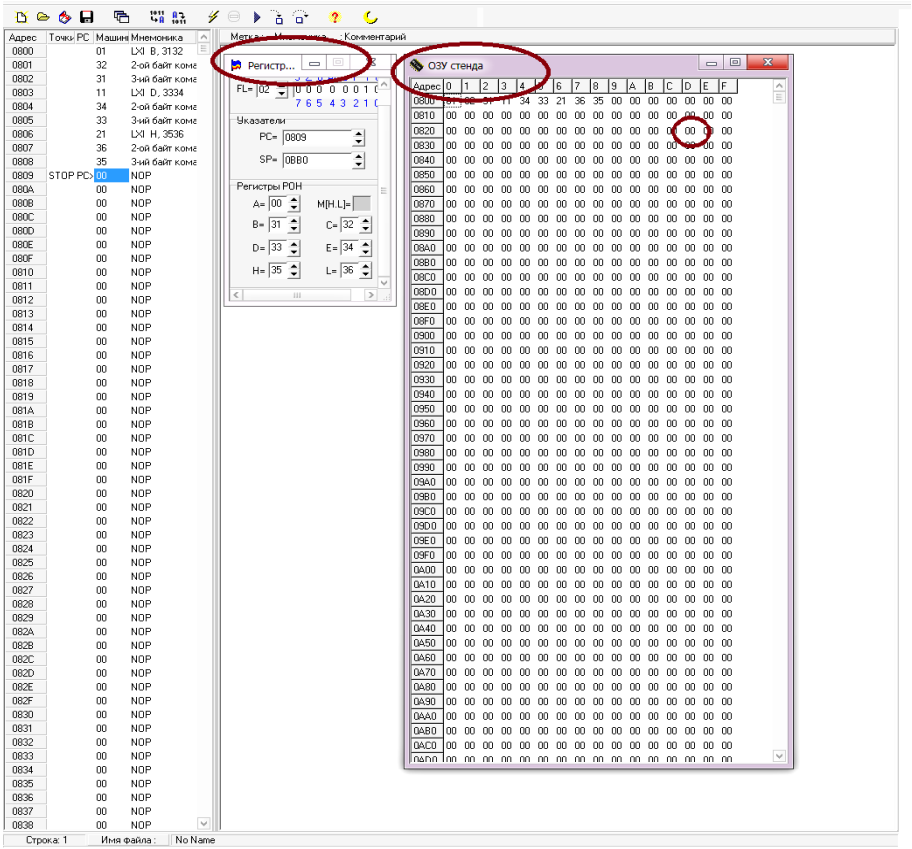


Рисунок 2.6 – Вікно представлення моделі з оперативною пам’яттю, регістрами та прапорцями

Для редагування комірки пам’яті необхідно ввести нове значення для комірки пам’яті в полі введення, а потім підтвердити введення, натиснувши кнопку «Зап.+Увел.».

Щоб отримати навички роботи з емулятором, необхідно спробувати його роботу на прикладі. Тому дотримуйтеся цієї послідовності, щоб програмувати в емуляторі.

Наприклад, виконайте програму згідно з таблицею 2.5, яка реалізує рівняння: HL = (B OR C) AND DE AND F0FF H.

Таблиця 2.5

Адреса	Команда	Код	Коментар
800	MVI B, 00	06 00	Завантажити в B← 00 (вміст даних)
802	MVI C, 01	0E 01	Завантажити в C ← 01 (вміст даних)
804	MVI D, 10	16 10	Завантажити в D ← 10 (вміст даних)
806	MVI E, 23	1E 23	Завантажити в E ← 23 (вміст даних)
808	MOV A, C	79	Переслати вміст даних регістру C в регістр A (A ← C)
809	ORA B	B0	Порівняння A = A OR B
80A	ANA E	A3	Порівняння A = A AND E
80B	ANI FF	E6 FF	Порівняння A = A AND FF H
80D	MOV L, A	6F	Переслати вміст даних регістру A в регістр L (L ← A)
80E	MOV A, D	7A	Переслати вміст даних регістру D в регістр A (A ← D)
80F	ANI F0	E6 F0	Порівняння A = A AND F0 H
811	MOV H, A	67	Переслати вміст даних регістру A в регістр H (H ← A)

Для запуску емулятора необхідно активувати безпосередньо програму K580. На екрані ви побачите головне вікно разом із контекстним меню (рисунок 2.1).

Наступним кроком є запис машинних кодів у пам'ять. Для цього потрібно перемістити курсор на «Код машини», як показано на рисунку 2.2. Емулятор автоматично запише мнемоніку (або команди збірки), яку ви завантажили.

Якщо необхідно встановити початок програми, тобто початкову адресу пам’яті, то потрібно двічі клацнути лівою клавішею

біля відповідної адреси до появи знаку «PC», як показано на рисунку 2.7.

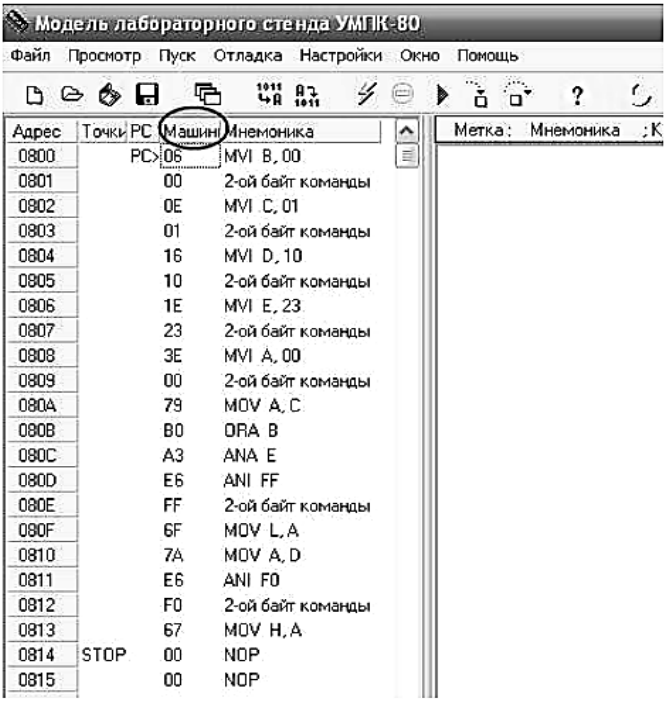


Рисунок 2.7 – Запис кодів команд до пам’яті емулятора

Якщо вам потрібно записати код в останню комірку пам’яті, вам потрібно клацнути лівою кнопкою миші біля адреси, щоб з’явилося слово «STOP», як показано на рисунку 2.8.

Для безпосереднього виконання самої програми необхідно під час роботи емулятора натиснути курсор на початок програми (тобто на першу адресу), зайти в контекстне меню «Пуск» і натиснути кнопку «Виконати», як показано на рисунку 2.9 має зупинятися на останній адресі виконуваної програми.

Для перевірки записаної програми або отриманого результату, що знаходиться в комірці пам’яті, необхідно зайти в «Огляд» і «ОЗУ стенда» та для перевірки отриманого результату необхідно зайти в «Просмотр» і у вікні вибрати «Регистры и флаги» (див. рисунок 2.6).

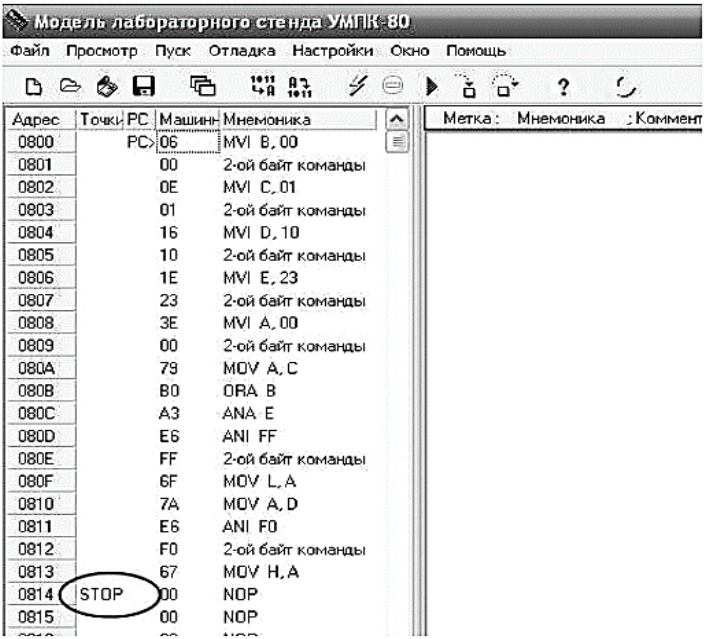


Рисунок 2.8 –Запис кодів команд до пам’яті емулятора

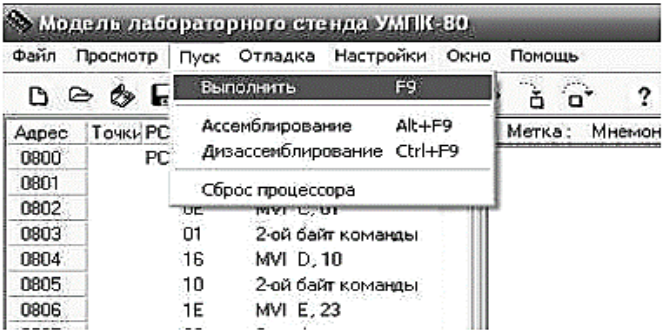


Рисунок 2.9 – Запуск виконання програми

2.2 Зміст звіту

Звіт повинний містити:
– назву та мету роботи;
– алгоритми виконання кожної команди відповідно до завдань в підрозділі 2.3;
– висновки.

2.3 Питання для самоперевірки

2.3.1 Призначення й можливості емулятора програми «Монітор». Характеристика її системних ознак.

2.3.2 Характеристика основних команд і операцій, їхній зв'язок з функціональною клавіатурою й індикацією.

2.3.3 Основні операції з масивами пам'яті.

2.3.4 Поняття буфера, особливості його використання.

2.3.5 Порядок і особливості роботи з областями пам'яті, що перекриваються.

3 ЛАБОРАТОРНА РОБОТА № 3 АРХІТЕКТУРА МІКРОПРОЦЕСОРА КР580ВМ80А. РЕГІСТРИ МІКРОПРОЦЕСОРА. КОМАНДИ ЗАВАНТАЖЕННЯ РЕГІСТРОВ. КОМАНДИ ПЕРЕДАВАННЯ ДАНИХ

Мета роботи – вивчення структури МП КР580ВМ80А і команд роботи з програмно доступними регістрами цього мікропроцесора.

3.1 Архітектура мікропроцесора КР 580ВМ80А

3.2.1 Організація мікропроцесора КР580ВМ80А.

Структурна схема досліджуваного МП представлена на рисунку 3.1.

До складу МП КР580ВМ80А входять такі вузли:

– шість восьмирозрядних регістрів загального призначення (РЗП) – B, C, D, E, H, L;

– восьмирозрядний акумулятор (А);

– чотири восьмирозрядних регістри для тимчасового зберігання інформації БР1, БР2, W і Z;

– восьмирозрядний регістр ознак (прапорів) РО (F);

– восьмирозрядний арифметико-логічний пристрій (АЛП) паралельної дії;

– схема десяткової корекції (СДК);

– восьмирозрядний регістр команд (РК);

– шістнадцятирозрядний лічильник команд (ЛК або РС);

– шістнадцятирозрядний покажчик стека (ПС або SP);

– шістнадцятирозрядний регістр адреси (РА);

– дешифратор команд і схема керування машинним циклом (ДШК і СКМЦ);

– мультиплексор (М);

– схема вибору регістра (СВР);

– пристрій керування (ПК);

– буфер даних (БД);

– буфер адреси (БА).

Для програміста в МП КР580ВМ80А доступними є дев’ять регістрів: шість РЗП, акумулятор, лічильник команд, покажчик стеку, регістр ознак. РЗП використовуються для збереження чисел, що беруть участь в операції, для показу адреси комірки пам'яті чи шістнадцятирозрядних даних. В останньому випадку восьмирозрядні регістри B, C, D, E, H, L поєднуються в регістрові пари BC, DE, HL. У перших регістрах пар - B, D, H зберігаються старші байти, а в других - C, E, L -молодші. Звертання до регістрової пари здійснюється за ім`ям першого регістра. Отже, до перерахованих пар можна звертатися по буквах B, D, H відповідно. Крім імені кожен РЗП, а також регістр А мають тризначний двійковий код:

АЛП призначений для виконання арифметичних і логічних операцій над восьмирозрядними даними й операндами.

Акумулятор А – головна частина арифметичного пристрою МП. Всі арифметичні і логічні операції виконуються в АЛП з використанням акумулятора. Для кожної з таких операцій передбачається розміщення одного операнда в акумуляторі, другого – в пам'яті або в РЗП. Результат операції розміщується в акумуляторі. Операції запису в стек і зчитування зі стека показані на рисунку.3.2.

М – це комірка пам'яті, адреса якої міститься в регістровій парі HL.

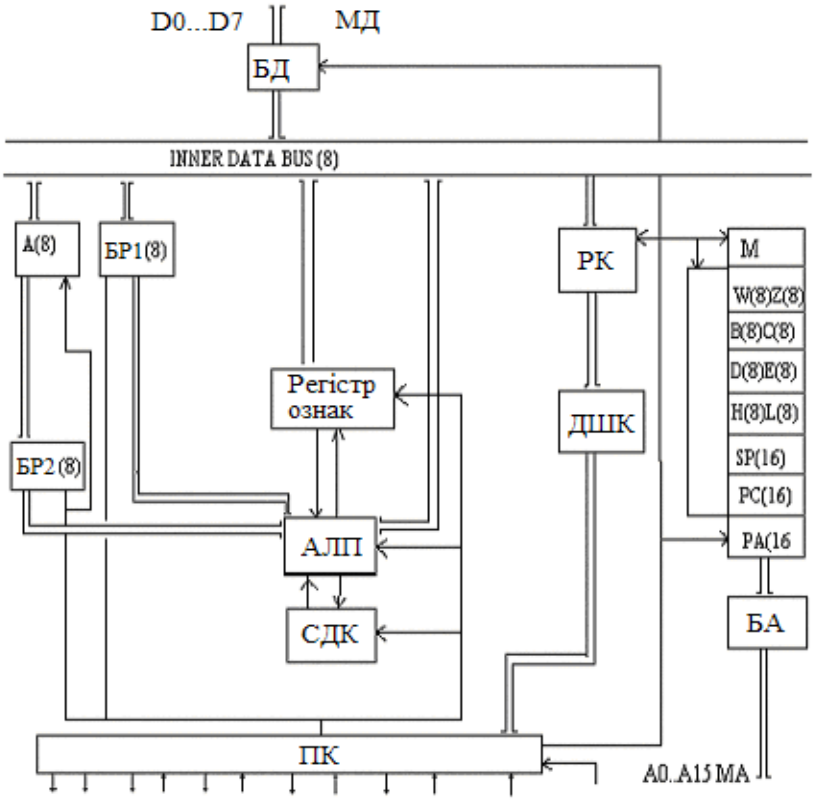


Рисунок 3.1 – Структурна схема МП КР580 ВМ80

Ім'я регістру.....	B	C	D	E	H	L	M	A
Двійковий код ...	000	001	010	011	100	101	110	111

Лічильник команд ЛК використовується для організації звертання до комірок пам'яті, у яких зберігається програма. Він усякий раз зберігає адресу команди, що повинна виконуватися. Після виконання чергової команди його вміст збільшується на 1, на 2, або 3 в залежності від довжини команди в байтах.

Покажчик стека SP (ПС). Стек – це область пам'яті, до якої дані записуються, і з якої зчитуються в строго визначеному порядку відповідно до алгоритму LIFO (Last-In, First-Out) –останній увійшов - перший вийшов. За допомогою шістнадцятирозрядного ПС програміст визначає яка область пам'яті виділяється під стек.

ПС містить адресу комірки, що доступна для зчитування інформації. Після виконання чергового зчитування вміст ПС автоматично збільшується на 1. Перед записом до стеку вміст ПС зменшується на 1, після чого виконується запис.

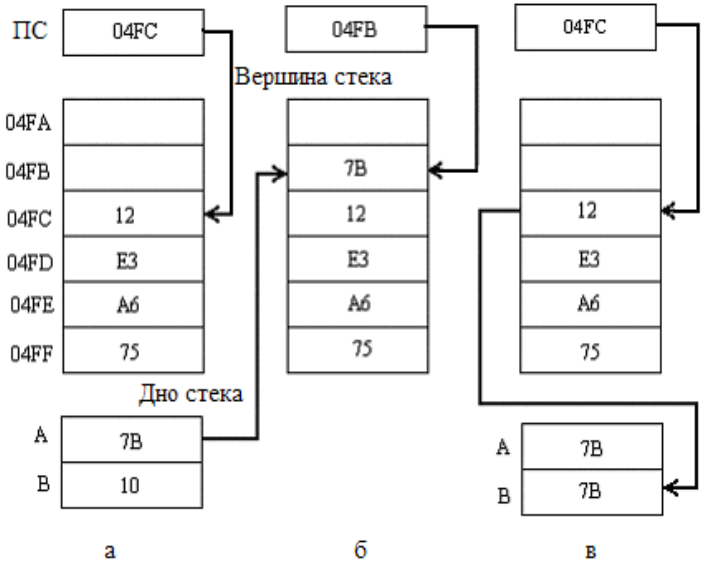


Рисунок 3.2 – Операції зі стеком

а) – початковий стан стека; б) – стек після виконання команди "записати в стек код з акумулятора А"; в) – стек після виконання команди "читати зі стека код у регістр В"

Регістр ознак F призначений для зберігання п’яти ознак (прапорів), які з`являються при виконанні деяких операцій. Ці ознаки (біти умов) зберігаються у відповідних розрядах РО (рисунок 3.3).

РЕГІСТР ОЗНАК F

7	6	5	4	3	2	1	0
S	Z	O	AC	0	P	1	C

Рисунок 3.3 – Регістр ознак

S (Sign) – ознака знака результату, що зберігається в акумуляторі. При позитивному знаку S=0, при від’ємному S=1.

Z (Zero) – якщо вміст акумулятора =0, то ознака Z=1, інакше Z=0.

C (Carry) – ознака переносу. C=1, якщо при виконанні команд з'являється одиниця переносу зі старшого розряду (переповнення).

AC (Auxiliary Carry) – додаткова ознака переносу. Встановлюється в одиницю, якщо при виконанні команд виникає одиниця з третього розряду числа (з молодшої тетради в старшу).

P (Parity) – ознака парності. P=1, якщо кількість одиниць у розрядах акумулятора буде парною. Нульовий результат також відноситься до парного. У протилежному випадку P=0.

Розряди 1, 3 і 5 у регістрі ознак не використовуються як ознаки. Повний опис кожної команди МП повинен містити інформацію про те, на які ознаки в регістрі F ця команда впливає. Вміст пари регістрів A і F називають словом стану програми і позначають PSW.

3.2 Структура команд

Двійкове число, обране з пам'яті і, що вказує на виконання означеної операції, називається командою. Двійкове число, яке підлягає обробці, називається операндом. Система команд, реалізована МП, містить 78 базових команд (загальне число команд – 244). Команда може містити від одного до трьох форматних полів. Звичайно команда МП КР580ВМ80А складається з двох форматних полів: поля коду операції (КОП) і поля адреси операнду (адресна частина). У МП КР580 використовуються кілька способів завдання адреси операндів - способів адресації: пряма регістрова адресація, непряма регістрова адресація, пряма адресація і безпосередня адресація. Для різних способів адресації потрібне різне число розрядів у форматі команди. Тому команди МП мають різну довжину – один, два чи три байти.

При прямій регістровій адресації операнд (один чи два) знаходяться в РЗП, адреса якого вказується в адресній частині команди (у двійковій системі адреси мають код 000 – 111, а в шістнадцятирічній – 0-7). Такі команди мають довжину один байт. На рисунку 3.4,*а* приведене двійкове представлення команди пересилання вмісту акумулятора в регістр B. Праворуч зазначений шістнадцятирічний код команди.

При прямій адресації адреса комірки пам'яті вказується в полі адреси команди. Такі команди мають довжину в три байти: перший байт - код операції, другий байт - молодші розряди адреси, третій байт - старші розряди адреси. На рисунку 3.4,*б* показана команда завантаження акумулятора вмістом комірки 7AC3. При безпосередній адресації операнд є частиною самої команди і у залежності від його розрядності займає другий чи другий і третій байти команди.

На рисунку 3.4, *в* представлена команда пересилання коду 4C в акумулятор, а на рисунку 3.4, *г* – завантаження в регістрову пару B коду 1F5A. Будь-яка команда має мнемонічний (символічний) опис, що полегшує написання програми для програміста. Наприклад, ADD (скласти), MOV (переслати), XCHG (обмін вмістом регістрових пар D і H) та ін. Поле мнемоніки повинне відокремлюватися від операндів хоча б одним пробілом. Операнди – це числа в шістнадцятирічній системі.

Якщо число починається з букви, то перед буквою ставиться незначущий нуль, інакше це число буде сприйняте як ім'я. Наприклад, 0FAH, 0B7H, 0DEH і т.і.

Операнди відокремлюються комою, якщо їх декілька. Пробіли між операндами неприпустимі. Слід пам'ятати, що:

а) максимальне восьмирозрядне число –- 0FFH, а шістнадцятирозрядне – 0FFFFH;

б) необхідно по можливості використовувати більш короткі (менше число байтів) команди;

в) якщо команда впливає на означені біти регістра ознак, то решта бітів цього регістра не змінює свого стану.

Усі команди можна розділити на сім груп:

1) команди пересилання інформації;

2) команди передачі чи керування переходу;

3) команди арифметичних і логічних операцій;

4) команди організації підпрограм;

5) команди уведення й виведення;

6) команди роботи зі стеком;

7) команди керування станом МП.

У подальшому будемо користатися такими умовними позначеннями:

– R, R1, R2, ... – один з восьмирозрядних РЗП чи акумулятор;

– RP – регістрова пара B,D чи H, а також покажчик стека;

– data – восьмирозрядний операнд чи його ім'я;
– data16 – шістнадцятирозрядний операнд чи його ім'я;
– addr – адреса комірки пам'яті;
– port – восьмирозрядна адреса ПУВ чи його ім'я.

КОП	Регістр приймач	Регістр джерело	B1	0	1	0	0	0	1	1	1	47

а

КОП	B1	0	0	1	1	1	0	1	0	3A
Молодший байт адреси	B2	1	1	0	0	0	0	1	1	C3
Старший байт адреси	B3	0	1	1	1	1	0	1	0	7A

б

КОП	B1	0	0	1	1	1	1	1	0	3E
Операнд	B2	0	1	0	0	1	1	0	0	4C

в

КОП		0	0	0	0	0	0	0	1	01
Операнд 1		0	1	0	1	1	0	1	0	5A
Операнд 2		0	0	0	1	1	1	1	1	1F

г

а – пряма регістрова адресація; *б* – пряма адресація; *в, г* – безпосередня адресація.

Рисунок 3.4 – Способи адресації

3.3 Команди завантаження РЗП

Команда **MVI (Move Immediate)** служить для завантаження в РЗП восьмирозрядного двійкового операнду. Довжина її два байти, виконується за сім тактів. Формат команди MVI R, data.

Наприклад: MVI A, 0FFH – завантажити акумулятор числом FFH; MVI C, FFH – завантажити регістр с вмістом імені FFH.

Команда завантаження кожного регістра має свій шістнадцятирічний код. У першому байті вказується КОП і регістр приймач,

у другому – операнд. Якщо як регістр R у команді зазначений умовний регістр М, то другий байт команди (операнд) завантажується в комірку пам'яті, адреса якої попередньо записана у регістрову пару HL.

Таблиця 3.1 – Коди першого байту команди MVI

Регістр - приймач	A	B	C	D	E	H	L	M
Код першого байта команди MVI	3E	06	OE	16	1E	26	2E	36

Як приклад, рекомендується виконати таке завдання: записати в пам’ять, починаючи з адреси 800H, послідовність команд.

Таблиця 3.2

Адреса	Команда	Машинний код	Коментар
800	MVI A, 00	3E 00	Завантажити регістр A кодом 00H
802	MVI B, 01	06 01	Завантажити регістр B кодом 01H
804	MVI C, 02	0E 02	Завантажити регістр C кодом 02H
806	MVI D, 03	16 03	Завантажити регістр D кодом 03H
808	MVI E, 04	1E 04	Завантажити регістр E кодом 04H
80A	MVI H, 05	26 05	Завантажити регістр H кодом 05H
80C	MVI L, 06	2E 06	Завантажити регістр L кодом 06H

Варто пам'ятати, що команда MVI – двобайтна, тому для кожної команди приділяється дві комірки пам'яті. Машинний код команди вводиться побайтно. Так, в комірку 800 записується код 3E,

а в комірку 801 – 00 і т.д. Виконайте цю послідовність за допомогою команди <СТ>800 <–> 80Е<ВП>. На дисплеї з'явиться адреса зупину 80Е.

Перевірити вміст регістрів, використовуючи команду **ПЕРЕГЛЯД І МОДИФІКАЦІЯ ВМІСТУ РЕГІСТРІВ**.

Команда **LXI (Load register pair Immediate)** служить для завантаження зазначеної в команді регістрової пари шістнадцятирозрядними даними. Довжина команди 3 байти, виконується за 10 тактів. У першому байті вказується код операції і регістр-приймач, у другому і третьому – шістнадцятирозрядний операнд. Причому третій байт команди завантажується в перший регістр пари, а другий байт – у другий регістр пари.

Формат команди LXI RP, data 16

У якості RP використовуються регістрові пари B, D, H, SP.

Наприклад: LXI D, 0FF00H – у регістр D записується старший байт – FF, у регістр E молодший – 00.

Коди команди LXI наведені в таблиці 3.3.

Таблиця 3.3 – Коди команди LXI

Адресуєми регістрові пари	B	D	H	SP
Код першого байта команди LXI	01	11	21	31

Як приклад, рекомендується виконати таке завдання:

– запишіть у пам'ять, починаючи з адреси 800H, послідовність команд

– виконайте цю послідовність команд за допомогою команди **СТАРТ ПРОГРАМИ**;

– перевірте вміст усіх регістрів, що повинний збігатися з заданими значеннями.

Таблиця 3.4

Адреса	Команда	Машинний код	Коментар
800	LXI B, 3132H	01 32 31	Завантажити RP BC кодом 3132H
803	LXI D, 3334H	11 34 33	Завантажити RP DE кодом 3334H
806	LXI H, 3536	21 36 35	Завантажити RP HL кодом 3536 H

Команди завантаження регістра покажчика стека

Команда безпосереднього завантаження регістра SP має вигляд:

LXI SP, data 16

Команда непрямого завантаження регістра SP має вигляд:

SPHL.

Команда однобайтна. Код цієї команди F9.По команді SPHL у покажчик стека завантажується вміст пари HL. Тому регістрова пара HL попередньо завантажується потрібним числом.

Як приклад, рекомендується виконати таке завдання:

– запишіть у пам'ять за адресою 800H коди команди.

Таблиця 3.5

Адреса	Команда	Машинний код	Коментар
800	LXI SP, 0B10H	31 10 0B	Завантажити покажчик стека SP кодом 0B10H

– виконайте цю команду;

– перевірте вміст регістра SP;

– запишіть у пам'ять за адресою 800H такі команди

– виконайте ці команди: <СТ> 800 <–> 804 <ВП>;

– перевірте вміст регістра SP побайтно.

Усі команди завантаження РЗП ознаки не формують, тобто стан регістра F не змінюють.

Таблиця 3.6

Адреса	Команда	Машинний код	Коментар
800	LXI H,0B30H	21 30 0B	Завантажити RP HL кодом 0B30H
803	SPHL	F9	Завантажити регістр SP кодом з RP HL

3.4 Команди пересилання даних

Дана команда призначена для передачі інформації з регістра в регістр, з регістра в пам'ять і з пам'яті в регістр. Загальний вид команди:

MOV Rl, R2

R1- регістр – приймач;

R2- регістр - джерело.

Команда займає 1 байт і виконується за 5 тактів. Ознак не формує. Наприклад, MOV A, B – переслати вміст регістра B в акумулятор; MOV M, A – переслати вміст акумулятора в комірку пам'яті, адреса якої зберігається в парі HL. При виконанні всіх цих команд вміст регістра – джерела R2 зберігається.

Як приклад, рекомендується виконати таке завдання: запишіть у пам'ять, починаючи з адреси 800H, послідовність команд.

– виконайте цю послідовність команд;

– перевірте вміст регістрів.

Порівняйте об'єм пам'яті, займаний цією програмою, і тривалість її виконання з об’ємом і тривалістю аналогічної програми обнуління тих же регістрів, виконуваної за допомогою команд MVI для кожного регістра (див. завдання до 3.2.3).

Таблиця 3.7 – Коди команди MOV

Адреса	Команда	Машинний код	Коментар
800	MVI A, 01H	3E 01	Завантажити регістр A 01
802	MOV B, A	47	Переслати вміст A в B
803	MOV C, B	48	Переслати вміст B в C
804	MOV D, C	51	Переслати вміст C у D
805	MOV E, D	5A	Переслати вміст D у E
806	MOV H, E	63	Переслати вміст E в H
807	MOV L, H	6C	Переслати вміст H в L

3.5 Команда завантаження лічильника команд

По цій команді в лічильник команд записується вміст регістрової пари HL.

Команда має вид: PCHL. Код цієї команди E9.

Наприклад, щоб завантажити в лічильник команд адреса 900 потрібно попередньо завантажити його в RP HL, а потім виконати команду PCHL. Ця команда не має операндів і виконує безумовний перехід з непрямою адресацією. Довжина команди 1байт, тривалість 4 такти. Команда PCHL ознак не формує.

Як приклад, рекомендується виконати таке завдання:

– запишіть у пам'ять за адресою 800H послідовність команд;

– виконайте ці команди: <СТ> 800 <–> 804<ВП>.

На дисплеї з'явиться адреса 0900.Це буде означати, що в лічильник команд завантажена адреса 0900H і здійснений перехід на цю адресу.

3.6 Зміст звіту

– назву та мету роботи;

– написані програми відповідно до завдань в підрозділах 3.3, 3.4 та 3.5 в мнемокодах та машинних кодах з коментарями;

– висновки.

Таблиця 3.8

Адреса	Команда	Машинний код	Коментар
800	LXI H, 0900H	21 00 09	Завантажити RP HL кодом 0900H
803	PCHL	E9	Завантажити PC кодом RP HL. Перейти на адресу 0900H

3.7 Питання для самоперевірки

3.7.1 Розрядність основних шин.
3.7.2 Програмно доступні вузли мікропроцесора, їхня коротка характеристика.
3.7.3 Система команд, група команд, опис команди.
3.7.4 Особливості адресації.
3.7.5 Команди завантаження регістрів і пари регістрів.
3.7.6 Команди пересилання.

4 ЛАБОРАТОРНА РОБОТА № 4 АДРЕСАЦІЯ ПАМ'ЯТІ. МЕТОДИ І КОМАНДИ РОБОТИ З ПАМ'ЯТТЮ

Мета роботи – вивчення методів адресації і команд роботи з пам'яттю.

4.1 Короткі теоретичні відомості

Пам'яттю називають пристрій, що забезпечує збереження команд і даних. Розрізняють пам'ять постійну (довгострокову) і оперативну. Відповідно і пристрої, що реалізують кожний вид пам'яті, скорочено називають ПЗП і ОЗП. Доступним користувачу практично у всіх основних режимах роботи УМПК є ОЗП, його роботу надалі і слід розуміти при використанні терміна пам'ять. Пристрій пам'яті складається з блоків однакового розміру - комірок пам'яті, розміром один байт кожна, призначених для збереження

одного слова інформації в двійкові коді. У свою чергу кожна комірка складається з елементів пам'яті, стан кожного з яких відповідає одній двійковій цифрі - біту інформації. Комірки пам'яті нумеруються числами від 0 до 65535, що називають адресами. Часто для зручності використовують шістнадцятирічні значення адрес, тоді діапазон адресації УМПК складає 0000H-FFFFH.Щоб записати в пам'ять слово, належить подати на шину адреси пам'яті сигнали, що відповідають адресі комірки, у яку необхідно помістити записуване слово, і подати необхідні сигнали, що відповідають значенню слова, на шину запису.

Пам'ять організована таким чином, що задане слово буде передано в комірку із зазначеною адресою і може зберігатися там як завгодно довго. В ОЗП слово зберігається доти, поки подана живляча напруга. У будь-який момент, звернувшись до пам'яті, можна одержати вміст збереженого слова (копію). Для цього вказується необхідна адреса. При зчитуванні вміст комірки пам'яті не змінюється. Час доступу не залежить від адреси комірки пам'яті. В УМПК доступними користувачу є комірки з адресами з 0800 до FFC9. Адреси портів уведення-виведення можуть знаходиться в межах ООН-FFH, програма «Монітор» займає простір FFCAH - FFFFH. Для проміжного оперативного збереження слів застосовуються регістри і регістрові пари, що складаються з набору запам'ятовуючих елементів, пронумерованих за розрядами.

Розрізняють такі засоби адресації:

1) пряму, 2) безпосередню, 3) непряму, 4) пряму регистрову, 5) непряму регистрову, 6) індексну, 7) відносну,) базову або індексну-базову, 9) неявну і 10) стекову.

1) Пряма адресація забезпечує доступ до обмеженої частини адресного простору, оскільки в коді команди записується лише невелике число біт адреси (коротка пряма адреса), що доповнюється нулями в інших розрядах. Іноді для розміщення виконавчої адреси використовується збільшення команди до двох або трьох машинних слів. Молодший байт адреси міститься в другому байті команди, а старший – у третьому.

2) Безпосередня адресація (рисунок 4.1, *б*) забезпечує швидке виконання команди, оскільки в полі адреси записана не адреса операнда, а власне операнд. Після вибірки такі команди виконуються без обертання до пам'яті, реалізуючи, наприклад, занесення коефіцієнта в РЗП або інші дії з константами.

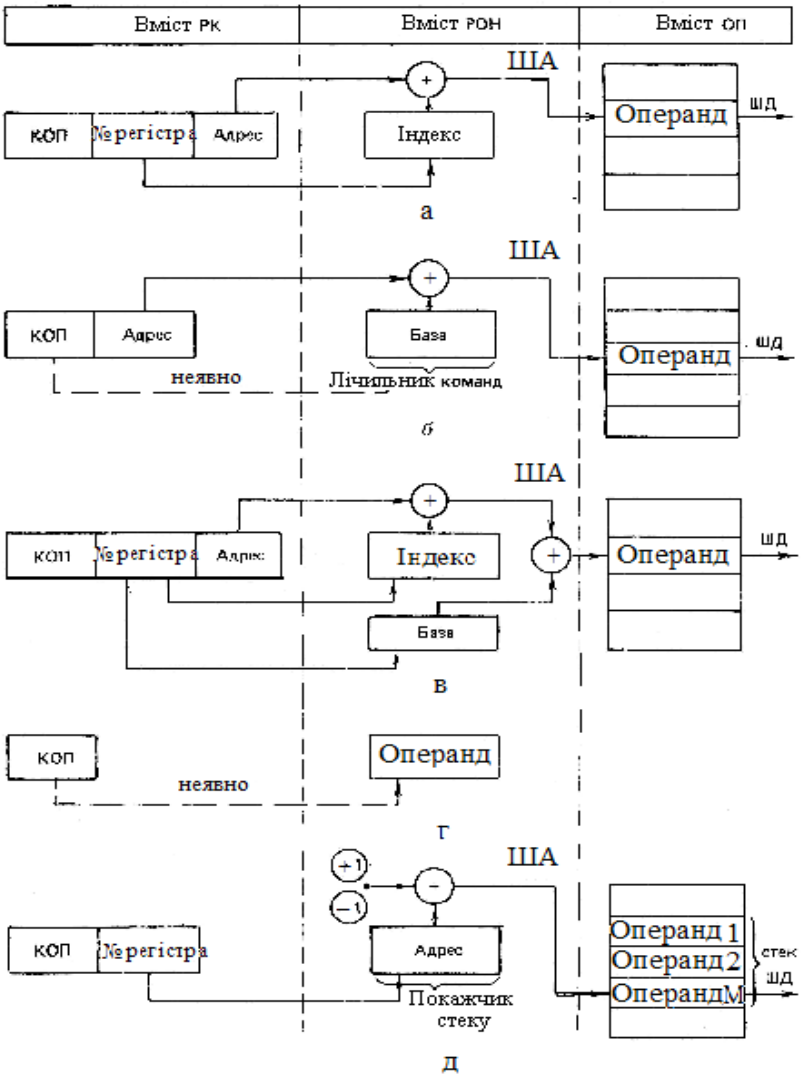
3) Непряма (косвенная) адресація (рисунок 4.1, *в*) передбачає вибірку адреси операнда з комірки пам'яті, адреса якого вказується в команді. Цей засіб можна назвати «адресацією адреси». Він дозволяє виконувати різноманітні дії з адресами, що знаходяться в пам'яті, не змінюючи команд програми. Це спрощує адресацію елементів масивів, реалізацію розгалуження (ветвления) і т.д.

Регистрова адресація. Вибірка операндів пришвидшується, якщо вони розміщені в регистрах РЗП процесора. Для їхньої адресації (указівки номера) потрібно усього 3 - 4 біта в коді команди. При такій 4) прямій регистровій адресації (рисунок 4.1, *г*) скорочується час виконання команд, але використовані дані попередньо повинні бути сформовані в процесорі або занесені ззовні. Для роботи з масивами, а також у багатьох інших випадках зручна 5) непряма регистрова адресація (рисунок 4.1, *д*), коли адреса операнду знаходиться в однім із регістрів процесора і для обертання до пам'яті його вміст пересилається, як виконавча адреса. При цьому з'являється проста можливість зміни адресів даних, наприклад, додавання (або віднімання) по одиниці до адреси в регістрі, що організує лічильник адреси. Цей засіб одержав назву автоінкрементної (+1) і автодекрементної (-1) адресації (рисунок 4.1, *е*).

6) Індексна адресація. Одним із засобів зміни виконавчих адрес без модифікації команд є використання спеціального зсуву (індексу), який записується в окремому індексному регістрі процесора і додається до адреси команди, забезпечуючи, наприклад, вибірку елементів масивів по їхніх індексах у циклах. Спеціальні засоби адресації ОП показані на рисунок 4.1. Індексна адресація (рисунок 4.1, *а*) призводить до деяких затримок витягу операнда, тому що необхідно обчислити його адресу, що звичайно виконується АЛП або спеціальним суматором у процесорі. Варіант цього засобу адресації можна розглядати як непряму регистрову адресацію на основі індексного регістра, до вмісту якого додається зсув із поля адреси команди.

7) Відносна адресація. Одним із засобів формування складової адреси є відносна адресація або базова (рисунок 4.1, *б*), що дозволяє робити настроювання програми, і зокрема, команд організації циклів, переходів та інших на адреси, що змінюються. Адреса операнду, що виконується, обчислюється шляхом додавання базової адреси (бази) і позитивного або негативного зсуву від поля адреси команди. У якості базового регістра використовується лічиль-

ник команд, до номерів яких задаються як адреси даних, так і адреси переходів. Відносна адресація забезпечує можливість переміщення програм і даних у пам'яті ЕОМ без зміни команд.



а – індексна, б – відносна, в – індексно-базова, г – неявна, д – стекова

Рисунок 4.1 – Спеціальні засоби адресації

8) Базова або індексна-базова адресація. Для обертання до пам'яті великого об’єму використовуються складові адреси, коли в коді команди записується лише зсув, а виконувана адреса утворюється додатком до зсуву розрядів базової перемінної (покажчика), що зберігається в спеціальному базовому регістрі, а в ряді випадків і індексу, номер регістра якого визначається в коді команди. Механізм індексної-базової адресації із зсувом показаний на рисунок 4.1, *в*. Він дозволяє істотно розширити область оперативної пам'яті (наприклад, до 1 Мбайта) при широких можливостях зміни (переміщення) виконавчих адрес без модифікації програм.

9) Неявна адресація. Ряд команд в ЕОМ виконується без спеціальної вказівки місця положення операнду, оскільки по характеру самої операції це місце однозначно припускається, наприклад, команди зсувів або пересилок у РЗП. Це неявний засіб адресації (рисунок 4.1, *г*), який забезпечує швидке виконання команди.

10) Стекова адресація. Роботу з виділеним в ОП стеком забезпечує спеціальний регістр - покажчик стеку (ПС), у котрому постійно зберігається адреса «верхівки стеку». Для занесення в стек або вибірки зі стеку використовується варіант непрямої регистрової адресації з автозменьшенням (-1) і автопідвищенням (+1), що базується на ПС (рисунок 4.1, *д*). Команди зі стековою адресацією дозволяють, ефективно реалізувати процедури переривання програм і переходів на підпрограми, чітко зберігаючи послідовність їхнього виконання.

У кожної ЕОМ використовуються ті або інші засоби адресації в залежності від її організації і призначення.

У мікропроцесорній системі застосовуються два методи адресації до пам'яті: безпосередня і непряма. Розрізняють такі види адресації: регістрову, непряму - регістрову, безпосередню і пряму. При безпосередній адресації адреса комірки пам'яті указується в другому і третьому байтах команди. Формат команди в загальному виді: КОП AHAL,

де КОП - код операції,

AHAL - адреса комірки пам'яті (АН – старший байт адреси, AL – молодший байт адреси).

Через те, що зазначена команда трибайтна, вона буде розміщена в трьох комірках пам'яті. Після байта коду операції розташовуються молодший байт адреси, а потім – старший:

n КОП;

n + 1 AL;
n + 2 AH.

4.2 Методичні рекомендації

Існує дві команди безпосереднього запису в пам'ять.

STA AHAL - запис у пам'ять по безпосередній адресі AHAL вмісту регістра A.

SHLD AHAL - запис у пам'ять вмісту регістрової пари HL.

По даній команді за адресою AHAL записується вміст регістра L, а за адресою AHAL + 1 – вміст регістра H.

Для прикладу виконайте таке завдання:

– запишіть у пам’ять команди з таблиці 4.1;

Таблиця 4.1

Адреса	Команда	Код	Коментар
800	STA 880H	32 80 08	Запис у пам'ять з регістра A за адресою 880H
803	SHLD 890H	22 90 08	Запис у пам'ять: з регістра L за адресою 890H; з регістра H за адресою 891H.

– уведіть значення регістрів: A = 37H, H = 55H, L = AAH;

– виконайте команду: <СТ> 800 <–> 806 <ВП>;

– перевірте результат за значеннями вмісту комірок 880H, 890H, 891H.

Непряма адресація припускає, що адреса комірки пам'яті розташована у регістрових парах HL, BC, DE. Для кожної конкретної команди роботи з пам'яттю закріплена своя регістрова пара. Тому перш, ніж задати команду необхідно вказати адресу у відповідній регістровій парі.

Наприклад:

1) LXI H, 800 H – запис адреси в регістр HL;
   MOV M, A – запис у пам'ять з регістра A за адресою HL.
2) LXI D 900H – запис адреси в регістр DE;
   STAX D – запис у пам'ять з регістра A за адресою DE.

Команди читання пам'яті так само розрізняють за типами адресації на безпосередню і непряму.

Команди безпосереднього читання пам'яті:

LDA AHAL - читання пам'яті по безпосередній адресі AHAL у регістр A;

LHLD AHAL - читання пам'яті по безпосередній адресі AHAL у регістрову пару HL.

При цьому в регістр L записується вміст комірки AHAL, а в регістр H - вміст комірки з адресою AHAL + 1.

Як приклад, запишіть у пам’ять команди з таблиці 4.2:

– виконайте зазначені команди: <СТ> 800 <–> 806 <ВП>;

– перегляньте вміст регістрів: A = 37 H, H = 55 H, L = AA H.

Таблиця 4.2

Адреса	Команда	Код	Коментар
800	LDA 880H	3A 80 08	Читання в регістр A з комірки 880H
803	LHLD 890H	2À 90 08	Читання в регістр L з комірки 890H; у регістр H вмісту комірки 891H

Команди читання - запису пам'яті при адресації через регістрову пару HL мають вид:

MOV M, R – запис у пам'ять вмісту регістра;

MOV R, M – завантаження регістра з пам'яті,

де R – регістр загального призначення A, B, C, D, E, H, L.

Як приклад, запишіть в пам'ять програму з табл. 5.3:

– виконайте команду: <СТ> 800 <–> 826 <ВП>;

– перевірте вміст комірок пам'яті: 900 H = AAH, 901 = CCH, 902 = BBH, 903 = EEH, 904 H = DDH, 905 H = 09H, 906 H = 06H.

Таблиця 4.3

Адреса	Команда	Код	Коментар
800	MVI A, AAH	3E AA	Завантаження регістра A
802	MVI B, BBH	06 BB	Завантаження регістра B
804	MVI C, CCH	0E CC	Завантаження регістра C
806	MVI D, DDH	16 DD	Завантаження регістра D
808	MVI E, EEH	1E EE	Завантаження регістра E
80A	LXI H, 900H	21 00 09	Завантаження HL = 900H, адреса M
80D	MOV M, A	77	Запис M=A, за адресою HL
80E	LXI H, 901H	21 01 09	---------- ‘ ’ ----------
811	MOV M, C	71	---------- ‘ ’ ----------
812	LXI H, 902H	21 02 09	---------- ‘ ’ ----------
815	MOV M, B	70	---------- ‘ ’ ----------
816	LXI H, 903H	21 03 09	---------- ‘ ’ ----------
819	MOV M, E	73	---------- ‘ ’ ----------
81A	LXI H, 904H	21 04 09	---------- ‘ ’ ----------
81D	MOV M, D	72	---------- ‘ ’ ----------
81E	LXI H, 905 H	21 05 09	---------- ‘ ’ ----------
821	MOV M, H	74	---------- ‘ ’ ----------
822	LXI H, 906 H	21 06 09	---------- ‘ ’ ----------
825	MOV M, L	75	---------- ‘ ’ ----------

Як приклад, запишіть в пам’ять програму з таблиці 4.4:

– виконайте команду: <СТ> 800 <–> 81C <ВП>;

– перевірте вміст регістрів:

A = DDH, B = EEH, C = BBH, D = CCH,

E = AAH, H = 09H, L = 06H.

Таблиця 4.4

Адреса	Команда	Код	Коментар
800	LXI H, 900 H	21 00 09	Завантаження HL = 900 адресою M
803	MOV E, M	5E	Читання E=M, за адресою HL
804	LXI H, 901H	21 01 09	і т.д.
807	MOV D, M	56	---------- ‘ ’ ----------
808	LXI H, 902H	21 02 09	---------- ‘ ’ ----------
80B	MOV C, M	4E	---------- ‘ ’ ----------
80C	LXI H, 903H	21 03 09	---------- ‘ ’ ----------
80F	MOV B, M	46	---------- ‘ ’ ----------
810	LXI H, 904H	21 04 09	---------- ‘ ’ ----------
813	MOV A, M	7E	---------- ‘ ’ ----------
814	LXI H, 905H	21 05 09	---------- ‘ ’ ----------
817	MOV H, M	66	---------- ‘ ’ ----------
818	LXI H, 906 H	21 06 09	---------- ‘ ’ ----------
81B	MOV L, M	6E	---------- ‘ ’ ----------

Команди читання – запису при адресації з використанням регістрових пар BC і DE мають вид:

STAX B – запис вмісту регістра A в пам'ять за адресою зазначеною в регістровій парі BC;

STAX D – запис з регістра A в пам’ять за адресою з регістрової пари DE;

LDAX D – читання вмісту пам’яті в регістр A за адресою зазначеному в регістровій парі DE;

LDAX B – читання з пам’яті в регістр A за адресою з регістрової пари BC.

Як приклад, виконайте таке завдання:

– записати в пам’ять програму з таблиці 4.5;

Таблиця 4.5

Адреса	Команда	Код	Коментар
800	LXI B, 900H	01 00 09	Завантаження BC = 900H
803	MVI A, 0FH	3E 0F	Завантаження A = 0FH
805	STAX B	02	Запис M = A за адресою BC
806	LXI D, 910H	11 10 09	Завантаження DE = 910H
809	MVI A, F0H	3E F0	Завантаження A = F0H
80B	STAX D	12	Запис M = A за адресою DE

– виконайте зазначену послідовність команд: <СТ> 800 <–> 80C <ВП>;

– перевірте вміст комірок 900H і 910H; 900H = 0FH, 910H = F0H.

Як приклад, запишіть в пам’ять програму з таблиці 4.6.

Таблиця 4.6

Адреса	Команда	Код	Коментар
800	LXI D, 900H	11 00 09	Завантаження DE = 900H
803	LDAX D	1A	Читання A = M за адресою DE
804	MOV L, A	6F	Пересилання L ←A
805	LXI B, 910H	01 10 09	Завантаження BC = 910H
808	LDAX B	0A	Читання A = H за адресою BC
809	MOV H, A	67	Пересилання H ← A

– виконайте зазначену послідовність команд:

<СТ> 800 <– > 80A <ВП>; - перевірте вміст регістрів H, L: H = F0H, L = 0FH.

4.3 Зміст звіту

Звіт повинен містити:
- назву та мету роботи;
- програми, написані на мові Асемблера;
- результати виконання програм, висновки по роботі.

4.4 Питання для самоперевірки

4.4.1 Фізичні основи пам'яті, її технічна реалізація. Види пам'яті.

4.4..2 Засоби і види адресації.

4.4.3 Особливості і види регістрової адресації.

4.4..4 Особливості звертання до комірок пам'яті.

4.4.5 Відмінність між прямою і безпосередньої адресаціями.

4.4.6 Відмінність регістрової адресації від непрямої регістрової.

5 ЛАБОРАТОРНА РОБОТА № 5 ДВІЙКОВА АРИФМЕТИКА МІКРОПРОЦЕСОРА. ОПЕРАЦІЇ І КОМАНДИ.

Мета роботи – **в**ивчення арифметичних операцій і команд мікропроцесора КР580ВМ80А.

5.1 Короткі теоретичні відомості

Незалежно від форми подання чисел у вихідних програмах МП оперує тільки з двійковими числами, що має свою специфіку. Поняттям двійкова арифметика відображується специфіка арифметичних операцій із двійковими числами. Базовими арифметичними операціями МП є додавання і віднімання двійкових чисел, що випливає з принципів організації обчислювальних процесів.

Додавання двійкових чисел, як і десяткових, виконується за розрядами від молодшого до старшого, але значно простіше в реалізації, тому що числа мають тільки два значення. Найбільшу відповідність природному представленню додавання дає таблична форма, у якій додатки утворюють рядки і стовпці, а результат додавання цифр однойменних розрядів знаходиться на перетинанні

відповідних рядка і стовпця. Стосовно до додавання двійкових цифр зазначена таблиця має вид таблиці 5.1.

Таблиця 5.1 – Додавання двійкових чисел

Додаток	Додаток		
	+	0	1
	0	0	1*
	1	1	0

* має місце перенос одиниці в старший розряд.

Додавання двійкових чисел виконується за наступними правилами:

– додавання двох одиниць дає нуль у молодшому розряді результату і перенос одиниці в старший розряд (1+1=0*);

– додавання одиниці до нуля молодшого розряду дасть у результаті одиницю без переносу в старший розряд (1+0=1);

– нулі, що складаються, результатом дають нуль (0+0=0).

При всій простоті двійкового додавання як процедури, громіздкість запису великих чисел у двійковій формі викликає необхідність безлічі переносів з одного розряду в іншій, що створює певні незручності.

Двійкове віднімання також багато в чому подібне десятковому. Таблична форма запису в даному випадку має вид таблиці 6.2.

Таблиця 5.2 – Віднімання двійкових чисел

Зменьшу-ване	Від’ємне		
		0	1
	0	0	1*
	1	1	0

* означає позику одиниці із сусіднього старшого розряду.

Основні правила віднімання:

– віднімання з одиниці саме одиниці і з нуля саме нуль дає нуль (1 – 1 = 0, 0 - 0 = 0);

– віднімання з одиниці нуля дає одиницю (1 – 0 = 1);

– віднімання з нуля одиниці вимагає позики одиниці зі старшого розряду і дає в результаті одиницю (0 – 1 = 1).

До числа елементарних арифметичних операцій відносять зсув двійкового числа на один розряд уліво чи управо, порівняння двох двійкових чисел, а також збільшення чи зменшення числа на 1.

До числа складних арифметичних операцій відносять такі, які можна представити у виді комбінації декількох простих. Так множення можна представити у виді комбінації додавання і зсуву, а ділення у виді послідовності віднімання і зсуву.

У мікропроцесорній системі КР580, варіантом якої є УМПК, передбачена реалізація таких команд двійкової арифметики:

а) додавання восьмирозрядних чисел;

б) додавання шістнадцятирозрядних чисел;

в) віднімання восьмирозрядних чисел;

г) інкремент;

д) декремент.

Всі арифметичні операції з восьмирозрядними операндами в МП КР580 основані на уявлені про те, що один з операндів розміщується в акумуляторі (регістр А), а інший - у регістрі (РЗП) чи в пам'яті (М). Адреса комірки пам'яті (М) вказується в регістровій парі HL. Другий операнд може бути також числом, заданим безпосередньо в самій команді. Результат арифметичної операції записується в акумулятор. Операція віднімання має ту особливість, що робиться завжди із вмісту акумулятора, тобто зменшуване повинне бути операндом, розміщеним в акумуляторі. По результаті арифметичних операцій додавання і віднімання встановлюються такі типи ознак:

С – переносу, Z – нуля, S – знаку, P – парності, AC – допоміжного переносу.

Додавання шістнадцятирозрядних двійкових чисел у зазначеній системі називають подвійним додаванням. Воно засновано на тому, що один з операндів знаходиться в регістровій парі HL, а другий - або в DE, або у BC. Результат записується в HL. По результату операції встановлюється чи скидається біт переносу – С.

Операція інкремента полягає в збільшенні вмісту регістрів, регістрових пар, комірки пам'яті за адресою в HL на одиницю. Інкремент регістра і пам'яті змінює біти ознак: Z, S, P, AC. Інкремент регістрової пари не торкається бітів ознак.

Операція декремента полягає в зменшенні вмісту регістрів, регістрових пар, комірки пам'яті за адресою в HL на одиницю. Змінювані біти ознак аналогічні команді інкремента.

5.2 Завдання

Вивчіть основні команди арифметичних дій з числами в шістнадцятирічній системі.

Команди додавання восьмирозрядних чисел:

– ADD R – додавання регістра: A, B, C, D, E, H, L;

– ADD M – додавання комірки пам'яті (адреса в HL);

– ADI B – додавання безпосереднього числа B;

– ADC R – додавання регістра:A, B, C, D, E, H, L плюс біт переносу C;

– ADC M – додавання комірки пам'яті (адреса в HL) плюс біт переносу C;

– ACI B – додавання безпосереднього числа B плюс біт переносу C.

Як приклад, виконайте таке завдання:

– уведіть програму, приведену в таблиці 5.3., що реалізує операцію A = A + B + M + 1;

Таблиця 5.3

Адреса	Команда	Код	Коментар
800	ADD B	80	A=A+B
801	LXI H, 900H	21 00 09	Завантаження адреси HL=900H
804	ADD M	86	A=A+M
805	ADI 01	C6 01	A=A+1

– виконайте програму, попередньо задавши вихідні значення відповідно до таблиці 5.4 (M=900H) за допомогою команди: <СТ> 800 <–> 807 <ВП>;

– перевірте отримані результати (результат операції в регістрі A та біти умов у регістрі F).

Таблиця 5.4

Регістр	Початкові дані	A	Ваш зріст
		B	Ваш зріст
		M (900H)	Ваш зріст
	Результат	A	
		F регістр ознак	

Як приклад, виконайте таке завдання:

– уведіть в пам'ять програми додавання шістнадцятирозрядних чисел на основі команд восьмирозрядного додавання представлену в таблиці 5.5 для виразу: HL=DE+BC;

– виконайте програму, попередньо задавши вихідні значення відповідно до Вашого зросту, за допомогою такої дії: <СТ > 800 <–> 806 <ВП>;

– перевірте отримані результати в регістрах H, L та F .

Таблиця 5.5

Адреса	Команда	Код	Коментар
800	MOV A, C	79	A ← C
801	ADD E	83	A + E
802	MOV L, A	6F	L ← A
803	MOV A, B	78	A ← B
804	ADC D	8A	D + B, з урахуванням переносу
805	MOV H, A	67	H ← A

Команди віднімання восьмирозрядних чисел мають вид:

– SUB R - віднімання регістра: A, B, C, D, E, H, L;

– SUB M- віднімання комірки пам'яті (адреса в HL);

– SUI B - віднімання безпосереднього числа B;

– SBB R - віднімання регістра: A,B,C,D,E,H,L мінус біт переносу C;

– SBB M - віднімання комірки пам'яті(адреса в HL), мінус біт переносу С;

– SBI B - віднімання безпосереднього числа мінус біт переносу.

Як приклад, виконайте таке завдання:

– уведіть в пам'ять програму, представлену в таблиці 5.6, що реалізує вираз: A = A – B – M – 1;

Таблиця 5.6

Адреса	Команда	Код	Коментар
800	SUB B	90	A = A – B
801	LXI H, 900H	21 00 09	Завантаження HL =900H, адреса M
804	SUB M	96	A=A – M
805	SBI 01	DE 01	A=A – 1

– виконайте програму, попередньо задавши вихідні дані відповідно до Вашого зросту командою: <СТ> 800 <–> 807 <ВП>;

– перевірте отримані результати.

Як приклад, виконайте таке завдання:

– уведіть в пам'ять програму, наведену в таблиці 5.7, віднімання шістнадцятирозрядних чисел: HL=DE – BC;

– виконайте програму, попередньо задаючи вихідні значення відповідно до Вашого зросту за допомогою команди: <СТ> 800 <–> 806 <ВП>;

– перевірте отримані результати.

Таблиця 5.7

Адреса	Команда	Код	Коментар
800	MOV A, E	7B	E ← A
801	SUB C	91	E – C
802	MOV L, A	6F	L ← A
803	MOV A, D	7A	A ← D
804	SBB B	98	D – B з урахуванням переносу
805	MOV H, A	67	H ← A

Команди подвійного додавання мають вид:

– DAD H – додавання HL=HL+HL;

– DAD B – додавання HL=HL+BC;

– DAD D – додавання HL=HL+DE.

Як приклад, виконайте таке завдання:

– уведіть згідно з таблицею 6.8 програму, що реалізує операцію: HL=BC+DE;

– виконайте програму, задавши вихідні значення відповідно до Вашого зросту за допомогою команди: <СТ> 800 <–> 803 <ВП>;

– перевірте результати.

Таблиця 5.8

Адреса	Команда	Код	Коментар
800	MOV H, B	60	H ← B
801	MOV L, C	69	L ← C
802	DAD D	19	HL = HL + DE

Команди інкремента мають вид:

– INR R - збільшення на одиницю вмісту регістрів: A, B, C, D, E, H, L;

– INR M - збільшення на одиницю вмісту комірки пам'яті, адреса M в HL;

– INX RP - збільшення на одиницю вмісту пари регістрів: BC, DE, SP. У даній команді вказується ідентифікатор старшого регістра (INX B).

Як приклад, виконайте таке завдання:

– запишіть в пам'ять команду INR E (E=E+1) за адресою 800H: 800 1C;

– виконайте зазначену команду для заданих значень вмісту регістра E згідно з таблицею 5.9 за допомогою команди: <СТ> 800 <–> 801 <ВП>;

– перевірте отриманий результат.

Таблиця 5.9

Регістр	E початкове значення	Ваш зріст
	E результат	
	F регістр ознак	

Завдання:

– запишіть в пам'ять коди команд:

800 21 00 09 LXI H, 900H Завантаження. HL = 900H, адреса M

803 34 INR M M = M + 1

– виконайте зазначену послідовність для заданих в таблиці 5.10 значень вмісту комірки пам'яті M за допомогою команди: <СТ> 800 <–> 804 <ВП>;

– перевірте отримані результати.

Таблиця 5.10

Регістр	M початкове значення	Ваш зріст
	M результат	
	F регістр ознак	

Завдання:
– запишіть в пам'ять код команди:

800 13 INX D DE = DE + 1;

– виконайте приведену команду для заданих у таблиці 5.11 значень вмісту регістрової пари DE за допомогою команди: : <СТ> 800 <–> 801 <ВП>;
– перевірте отримані результати.

Таблиця 5.11

DE початкове значення	Ваш зріст (два байти)
DE результат	
F регістр ознак	

Примітка: Значення регістра ознак F залишається рівним останньому значенню попереднього завдання через те, що інкремент пари регістрів не змінює ознак.

Команди декремента мають вид:
– DCR R - зменшення на одиницю вмісту регістра: A, B, C, D, E, H, L;
– DCR M - зменшення на одиницю вмісту комірки пам'яті - адреса M у HL;
– DCX RP зменшення на одиницю вмісту пари регістрів: BC, DE, HL, SP.
У команді вказують ідентифікатор старшого регістра, наприклад, DCX B.

Як приклад, виконайте таке завдання:

– запишіть в пам'ять таку команду:

800 0D DCR C C=C - 1.

– виконайте зазначену команду для заданих у таблиці 5.12 значень регістра C: <СТ> 800 <–> 801 <ВП>;

– перевірте отримані результати.

Таблиця 5.12

C початкове значення	Ваш зріст
C результат	
F регістр ознак	

Завдання:

– запишіть в пам'ять коди команд:

800 21 00 09 LXI H, 900H Завантаження HL = 900H, адреса M;

803 35 DCR M: M = M – 1.

– виконайте зазначені команди для початкових значень вмісту комірки пам'яті M, заданих у таблиці 5.13;

– виконайте команду за допомогою команди: <СТ> 800 <–> 804 <ВП>;

– перевірте отримані результати.

Таблиця 5.13

M початкове значення	Ваш зріст
M результат	
F регістр ознак	

Завдання:

– запишіть в пам'ять код команди:

800 2B DCX H HL = HL – 1

– виконайте команду для значень вмісту регістрової пари H, заданих у таблиці 5.14.

Таблиця 5.14

H поч.	0000	1000	FFFF	0001
H рез.	FFFF	0FFF	FFFE	0000
F	13	13	13	13

– команда запуску: <СТ> 800 <–> 801 <ВП>;

– перевірте отримані результати.

Примітка: Команда декремент регістрової пари не стосується вмісту бітів ознак.

5.3 Зміст звіту

Звіт повинен містити:

– назву та мету роботи;

– написані програми відповідно до завдань;

– результати виконання програм;

– висновки.

5.4 Питання для самоперевірки

5.4.1 Характеристика елементарних операцій двійкової арифметики.

5.4.2 Особливості і форми представлення операцій додавання і віднімання двійкових чисел.

5.4.3 Сутність і призначення операцій подвійного додавання, віднімання.

5.4.4 Операції зсуву, їхнє практичне значення.

5.4.5 Принципи заповнення і переносу масивів.

6 ЛАБОРАТОРНА РОБОТА № 6
ЛОГІЧНІ ОПЕРАЦІЇ. ОРГАНІЗАЦІЯ ТА КОМАНДИ

Мета роботи – **в**ивчення принципів, організації і команд логічних операцій.

6.1 Короткі теоретичні відомості

Багато застосувань мікропроцесорних пристроїв припускають логічну обробку інформації . Тому вони здатні замінити собою безліч логічних схем , реалізувати складні логічні функції на основі використання елементарних логічних операцій.

У системі команд МП КР580 для реалізації логічних операцій існують такі команди :

– логічне додавання;

– логічне множення;

–АБО, що вилучає;

– інверсія.

Команди логічного додавання реалізують логічну операцію АБО. Результат дорівнює 1 , якщо хоча б один з відповідних байтів дорівнює 1, і дорівнює нулю, якщо обоє дорівнюють нулю. Наприклад:

10101001

OR

00110010

10111011

де OR – позначення операції АБО .

Команди логічного множення реалізують логічну операцію І. Результат дорівнює 1, якщо обоє відповідних біта рівні 1, і дорівнює нулю , якщо хоча б один з них дорівнює нулю.

Наприклад :

10101000

AND

00110010

00100000,

де AND – позначення операції логічне І.

Команди АБО, що вилучає реалізують логічну операцію ДОДАВАННЯ ПО МОДУЛЮ ДВА . Результат дорівнює 1 , якщо

відповідні біти протилежні (1 і 0) , дорівнює 0 , якщо вони однакові (1 і 1, 0 і 0) .

Наприклад :

10101001

XOR

00110010

10011011

де XOR – позначення операції ДОДАВАННЯ ПО МОДУЛЮ ДВА .

Команда інверсії реалізує операцію ЗАПЕРЕЧЕННЯ вмісту акумулятора .

Наприклад :

10101001

NOT_____

01010110

де NOT - позначення операції ЗАПЕРЕЧЕННЯ .

Усі логічні команди виконуються побітно з восьми розрядними операндами . При цьому один з операндів розміщується в регістрі накопичувачі, акумуляторі , а другий - або в одному з регістрів загального призначення , або в комірці пам'яті , або задається в другому байті команди . Результат виконання команди записується в акумулятор. При цьому біт переносу встановлюється в нуль, а інші біти установлюються відповідно до результату виконання команди .

6.1.1 Команди логічного додавання

Команди логічного додавання є наступними:

– ORA R - з регістром: A, B, C, D, E, H, L;

– ORA M - з коміркою пам'яті , адреса якої в HL;

– ORI B - з безпосереднім операндом B.

Як приклад, рекомендується виконати такі завдання:

а) записати в пам'ять коди програми , відповідно до таблиці 6.1, що реалізує вираз : A = (A OR C OR M OR 80H).

Таблиця 6.1

Адреса	Команда	Код	Коментар
800	ORA C	B1	A = A OR C
801	LXI H, 900H	21 00 09	Завантажити HL = 900 H, адреса M
804	ORA M	B6	A = A OR M
805	ORI 80H	F6 80	A = A OR 80H

Задаючи вихідні значення відповідно до таблиці 6.2, виконати програму командою <СТ> 800 <–> 807 <ВП>.

Таблиця 6.2

A вихідні дані	Ваш зріст
C	Ваш зріст
M (900H)	Ваш зріст
A результат	
F регістр ознак	

Перевірити отримані результати.

б) записати в пам'ять коди програми відповідно до таблиці 6.3, що реалізує вираз HL = (BC OR DE).

Таблиця 6.3.

Адреса	Команда	Код	Коментар
800	MOV A, C	79	Пересилання A ← C
801	ORA E	B3	A = A OR E
802	MOV L, A	6F	Пересилання L ← A молодшого байту результату
803	MOV A, B	78	Пересилання A ← B
804	ORA D	B2	A = A OR D
805	MOV H, A	67	Пересилання H ← A, старшого байту результату

Попередньо задаючи вихідні значення відповідно до таблиці 6.4, виконати програму командою: <СТ> 800 <–> 806 <ВП>.

Таблиця 6.4

BC вихідні дані	Ваш зріст
DE вихідні дані	Ваш зріст
HL результат	
F регістр ознак	

Перевірити отримані результати.

6.1.2 Команди логічного множення

Команди логічного множення є наступними:
– ANA R – з регістром A, B, C, D, E, H, L;
– ANA M – з коміркою пам'яті, адреса якої в HL;
– ANA B – з безпосереднім операндом B.

Як приклад, рекомендується виконати такі завдання:

а) записати в пам'ять з таблиці 6.5 коди програми, що реалізує вираз: A = (A AND D AND M AND 7FH).

Таблиця 6.5

Адреса	Команда	Код	Коментар
800	ANA D	A2	A = A AND D
801	LXI H, 900H	21 00 09	Завантажити HL = 900 H, адреса M
804	ANA M	A6	A = A ANA M
805	ANI 7FH	F6 7F	A = A AND 7FH

Виконати програму, попередньо задаючи вихідні значення з таблиці 6.6, командою <СТ> 800 <–> 807 <ВП> .

Перевірити отримані результати і записати в таблицю.

A вихідні дані	Ваш зріст

D вихідні дані	Ваш зріст
M (900H) вихідні дані	Ваш зріст
A результат	
F регістр ознак	

б) Записати в пам'ять коди програми згідно з таблиці 6.7, що реалізує вираз HL = (B ORA C) AND DE AND F0FFH.

Таблиця 6.7

Адреса	Команда	Код	Коментар
800	MOV A, C	79	Пересилання A ← C
801	ORA B	B0	A = A OR B
802	ANA E	A3	A = A AND E
803	ANI FFH	E6 FF	A = A AND FFH
805	MOV L, A	6F	Пересилання L ← A, молодший байт результату
806	MOV A, D	7A	Пересилання A ← D
807	ANI F0H	E6 F0	A = A AND F0H
809	MOV H, A	67	Пересилання H ← A, старший байт результату

Виконати програму, попередньо задаючи вихідні значення, що відповідають Вашому зросту в кожний регістр, та використовуючи команду <СТ> 800 <–> 80A <ВП>.

6.1.3 Команди додавання за модулем два

Команди додавання за модулем два є такими:

– XRAR – з регістром: A, B, C, D, E, H, L;

– XRAM – з коміркою пам'яті, адреса якої в HL;

– XRIB – з безпосереднім операндом B.

Як приклад, рекомендується виконати наступнii завдання:

а) записати в пам'ять коди програми згідно з таблиці. 6.8, що реалізує вираз A = A XOR A XOR E XOR M XOR AAH .

Таблиця 6.8

Адреса	Команда	Код	Коментар
800	XRA A	AF	A = A XOR A, A = 0
801	XRA E	AB	A = A XOR E
802	LXI H, 900H	21 00 09	Завантажити HL =900 H, адреса M
805	XRA M	AE	A = A XOR M
806	XRI AAH	EE AA	A= A XOR AAH, результат

Виконати програму, задаючи вихідні значення з таблиці 6.9, командою <СТ> 800 <–> 808 <ВП> .

Таблиця 6.9

E вихідні дані	Ваш зріст
M вихідні дані	Ваш зріст
A вихідні дані	Ваш зріст
A результат	
F регістр ознак	

Перевірити отримані результати.

б) записати в пам'ять відповідно до таблиці 6.10 коди програми, що реалізує вираз HL = (A OR B) AND DE XOR (HL XOR C).

Таблиця 6.10

Адреса	Команда	Код	Коментар
800	ORA B	B0	A OR B = A
801	ANA E	AE	A = A AND E
802	MOV E, A	5F	Пересилання E ← A, запис проміжного результату
803	MOV A, C	79	Пересилання A ← C
804	XRA L	AD	A = A XOR L
805	XRA E	AB	A = A XOR E
806	MOV L, A	6F	Пересилання L ← A, молодший байт результату
807	MOV A, H	7C	Пересилання A ← H, ст. байт результату
808	XRA D	AA	A = A XOR D
809	MOV H, A	67	Пересилання H ← A, старший байт результату

Виконати програму, задаючи вихідні значення відповідно до таблиці 6.11, командою <СТ> 800 <–> 80A <ВП>. Перевірити отримані результати.

Таблиця 6.11

A вихідні дані	Ваш зріст
B вихідні дані	Ваш зріст
DE вихідні дані	Ваш зріст (два байти)
HL вихідні дані	Ваш зріст (два байти)
C вихідні дані	Ваш зріст
HL результат	два байти
F регістр ознак	

6.1.4 Команда інверсії

Команда інверсії є такою:
– CMA - інверсія акумулятора.

Як приклад, рекомендується виконати такі завдання:
а) записати в пам'ять програму, приведену в таблиці 6.12, що реалізує вираз A = NOT ((NOT B) AND (NOT C)).

Таблиця 6. 12

Адреса	Команда	Код	Коментар
800	MOV A, B	78	Пересилання A ←B
801	CMA	2F	F = NOT A
802	MOV B, A	47	Пересилання B ← A (NOT B)
803	MOV A, C	79	Пересилання A ← C
804	CMA	2F	A = NOT A (NOT C)
805	ANA B	A0	A = A AND B
806	CMA	2F	A = NOT A, результат

Виконати програму, задаючи вихідні значення відповідно до таблиці 6.13, командою <СТ> 800 <–> 807 <ВП> .

Таблиця 6.13

B вихідні дані	Ваш зріст
C вихідні дані	Ваш зріст
A результат	
F регістр ознак	

Перевірити отримані результати.
б) записати в пам'ять програму відповідно до таблиці 6.14, що реалізує вираз М3 = NOT (NOT Ml) OR (NOT M2) .
Адреси комірок пам'яті: Ml =900H, M2 = 901H, M3 = 902H.

Таблиця 6.14

Адреса	Команда	Код	Коментар
800	LXI H, 900 H	21 00 09	Завантажити HL = 900 H, адреса M1
803	MOV A, M	7E	Читання A = M1
804	CMA	2F	A= NOT A (NOT M1)
805	MOV C, A	4F	Пересилання C ← A, проміжний результат
806	INX H	23	HL = HL + 1, адреса M2 (901)
807	MOV A, M	7E	Читання A = M2
808	CMA	2F	A = NOT A (NOT M2)
809	ORA C	B1	A = A OR C
80A	CMA	2F	A = NOT A, результат
80B	INX H	23	HL = HL + 1, адреса M3 (902)
80C	MOV M, A	77	Запис M3 = A, результат

Виконати програму, задаючи вихідні значення відповідно з Вашим зростом за допомогою команди <СТ> 800 <–> 80D <ВП> .

Перевірити отримані результати.

6.2 Зміст звіту

Звіт повинен містити:

– назву та мету роботи;

– написані програми відповідно до завдань підрозділа 6.1 в мнемокодах та машиних кодах з коментарями;

– висновки.

6.3 Питання для самоперевірки

6.3.1 Особливості логічних операцій.

6.3.2 Сутність елементарних логічних операцій.

6.3.3 Технологія і команди логічного додавання.
6.3.4 Технологія і команди логічного множення.
6.3.2 Особливості і команди додавання за модулем два.
6.3.6 Зміст і практичне значення інверсії.

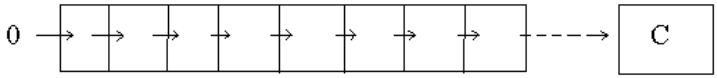
7 ЛАБОРАТОРНА РОБОТА № 7 ОПЕРАЦІЇ ТА КОМАНДИ ЗСУВУ

Мета роботи – вивчення операцій і команд простого та циклічного зсуву.

7.1 Короткі теоретичні відомості

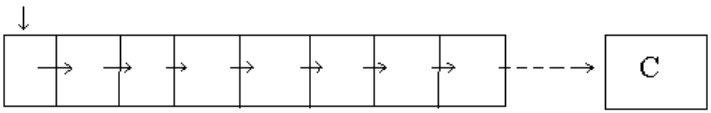
Команди зсуву дозволяють виконати на основі елементарних арифметичних операцій необхідний набір дій над числами. Крім того реалізують закінчені операції по обробці даних. Використовуються переважно при тестуванні та порівнянні бітів. Команди зсуву МП КР580 (Intel 8080) оперують тільки даними акумулятора і не вимагають інших операндів, розташованих у пам'яті чи регістрах. Спосіб адресації в даному випадку називають неявним, а часто просто не вказують. Багато які МП мають інші, ніж типовий МП типи команд зсуву. Наприклад, МП Motorola 68000 здійснює зсув вмісту не тільки акумулятора, але і комірки пам'яті, а команди операцій зсуву впливають не тільки на ознаку переносу, як це властиве базовим МП, але і на всі інші ознаки. Розглянемо діаграми можливих видів зсуву:

а) зсув управо відбувається таким чином:



Послідовність дій при цьому така:
– крайній правий біт засилається в С (регістр переносу);
– всі інші біти зсуваються управо;
– у крайній лівий розряд засилається 0.

б) арифметичний зсув управо відбувається таким чином:



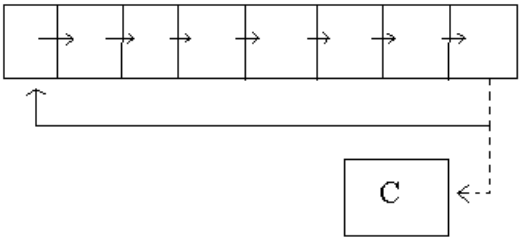
Послідовність дій при цьому така:

– крайній правий біт засилається в С;

– інші біти зсуваються управо;

– крайній лівий біт залишається без змін.

в) циклічний зсув управо відбувається таким чином:



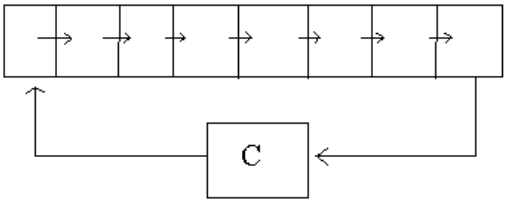
Послідовність дій при цьому така:

- крайній правий біт засилається в С;

- інші біти зсуваються управо;

- у крайній лівий розряд засилається первісний вміст крайнього правого розряду.

г) циклічний зсув управо з переносом відбувається таким чином:



Послідовність дій при цьому така:

– крайній правий біт засилається в С;

– інші біти зсуваються управо;

– у крайній лівий розряд засилається первісний вміст регістра переносу С.

Зсуви уліво за технологією ідентичні, але робляться в зворотному напрямку. Їхні діаграми й алгоритми пропонується побудувати самостійно.

У системі команд МП КР580 передбачені такі команди зсуву:

– циклічний зсув уліво;

– циклічний зсув управо;

– зсув уліво через перенос;

– зсув управо через перенос.

Команди зсуву виконуються в акумуляторі над восьмирозрядними операндами. Результат заноситься в акумулятор.

7.2 Команди циклічного зсуву

Команда циклічного зсуву уліво переміщує кожен біт у межах байту на один розряд уліво. При цьому вміст старшого розряду записується в молодший розряд і біт переносу. Команда циклічного зсуву управо переміщує кожен біт у межах байту на один розряд управо. При цьому вміст молодшого розряду записується в старший розряд і біт переносу.

– RLC – команда циклічного зсуву вліво;

– RRC – команда циклічного зсуву управо.

Як приклад, рекомендується виконати такі завдання:

а) записати в пам'ять коди програми, реалізуючої циклічний зсув байту на 4 розряди.

Таблиця 7.1

800	RLC	07	Циклічний зсув уліво на один розряд
801	RLC	07	----------//----------
802	RLC	07	----------//----------
803	RLC	07	----------//----------

Виконати програму, попередньо задаючи вихідні значення відповідно до Вашого зросту за допомогою команди <СТ> 800 <–> 804 <ВП>.

Перевірити отримані результати.

б) записати в пам'ять коди програми, реалізуючої операцію об'єднання старших тетрад двох байтів, що містяться в регістрах B і C, в один, використовуючи команду RRC.

Таблиця 7.2

800	MOV A, C	79	Пересилання A ← C
801	RRC	0F	Переміщення ст. тетради 1 байта
802	RRC	0F	на місце молодшої
803	RRC	0F	Переміщення ст. тетради 1 байта
804	RRC	0F	на місце молодшої
805	ANI FH	E6 0F	Вибір ст. тетради 1 байта
807	MOV C, A	4F	Пересилання C ← A
808	MOV A, B	78	Пересилання A ← B
809	ANI F0 H	E6 F0	Виділення ст. тетради 2 байта
80B	ORA C	B1	Об`єднання двох байтів в один

Виконати програму, задаючи попередньо вихідні значення відповідно до таблиці 7.3 командою <СТ> 800 <–> 80C <ВП>.

Таблиця 7.3

C вихідні дані	Ваш зріст
B вихідні дані	Ваш зріст
A результат	
F регістр ознак	

Перевірити отримані результати.

7.3 Команди зсуву за допомогою переносу

Команда зсуву уліво через перенос переміщує вміст кожного біта в межах байта на один розряд. При цьому вміст біта переносу записується в молодший розряд, а вміст старшого біта записується

в біт переносу. Використовуючи цю команду, можна реалізувати операцію множення на числа кратні 2. Команда зсуву уліво через перенос переміщує вміст кожного розряду в межах байта управо на один розряд. При цьому в старший розряд байта записується значення байта переносу, а в нього заноситься вміст молодшого розряду байта. Використовуючи дану команду, можна реалізувати команду ділення на числа, кратні 2.

– RAL - команда зсуву уліво через перенос;
– RAR - команда зсуву управо через перенос.

Як приклад, рекомендується виконати такі завдання:

а) записати в пам'ять коди програми, що реалізує циклічний зсув уліво на один розряд вмісту пари регістрів HL.

Виконати програму, попередньо задаючи вихідні значення відповідно до таблиці 7.4, командою <СТ> 800 <–> 80В <ВП>.

Перевірити отримані результати.

Таблиця 7.4

800	ORA A	B7	Скидання переносу в 0
801	MOV A, L	7D	A ← L
802	RAL	17	Зсув уліво на один розряд через перенос, в мол. розряд L - 0
803	MOV L, A	6F	L ← A
804	MOV A, H	7C	A ← H
805	RAL	17	Зсув уліво на один розряд через перенос
806	MOV H, A	67	H ← A
807	MOVA, L	7D	A ← L
808	ACI 0	CE 00	
80A	MOV L, A	6F	L ← A

Таблиця 7.5

HL вихідні дані	Ваш зріст
HL результат	

б) записати в пам'ять коди програми, що реалізує операцію множення на 4 вмісту регістра С: B = C · 4

Таблиця 7.6

800	MOV A, C	79	A ← C
801	ORA A	B7	Скидання біта переносу
802	RAL	17	Множення на 2
803	RAL	17	Множення на 2
804	MOV B, A	47	Результат в B

Виконати програму, попередньо задаючи вихідні значення за допомогою команди <СТ> 800 <–> 805 <ВП>. Перевірити отримані результати.

в) записати в пам'ять коди програми ділення на вісім вмісту регістра H. Цілу частину результату помістити в D, залишок в E.

Таблиця 7.7

800	MOV A, H	7C	A ← H
801	ORA A	B7	Скидання переносу в 0
802	RAR	1F	A = A/2
803	ORA A	B7	Перенос дорівнює 0
804	RAR	1F	A = A/2
805	ORA A	B7	Перенос дорівнює 0
806	RAR	1F	A = A/2
807	MOV D, A	57	D = H/8 - ціла частина
808	MOV A, H	7C	A ← H
809	ANI 07H	E6 07	Виділення залишку результату
80B	MOV E, A	5F	Залишок в E

Виконати програму, попередньо задаючи вихідні значення відповідно до Вашого зросту за допомогою команди <СТ> 800 <– > 80С <ВП>. Перевірити отримані результати.

7.4 Зміст звіту

Звіт повинен містити:
– назву та мету роботи;
– написані програми відповідно до завдань 8.3;
– результати;
– висновки.

7.5 Питання для самоперевірки

7.5.1 Характеристика зсуву управо та уліво.
7.5.2 Особливості арифметичних зсувів управо та уліво.
7.5.3 Алгоритми циклічних зсувів.
7.5.4 Циклічні зсуви з переносом, їхні особливості.
7.5.5 Операції і команди зсуву МП КР580.
7.5.6 Практичне використання операцій зсуву, їхня корисність.

8 ЛАБОРАТОРНА РОБОТА № 8 ОПЕРАЦІЇ І КОМАНДИ ПОРІВНЯННЯ

Мета роботи – **в**ивчення операцій і команд порівняння мікропроцесора КР580.

8.1 Короткі теоретичні відомості

На практиці дуже часті ситуації, коли виникає необхідність порівняння двійкових чисел. Найбільш типові вони при діагностиці правильності роботи МП (МПС) і прийнятті рішень у процесі обробки даних.

Операція порівняння складається в зіставленні двох будь-яких елементів даних.

Факт рівності двох чисел можна виявити за допомогою операцій «АБО, що виключає» і «Віднімання». Однак недоліком у даному випадку є часткова, або повна утрата вихідних даних при ви-

конанні зазначених операцій. Зазначених недоліків позбавлена операція ПОРІВНЯННЯ. Вона заснована на відніманні вмісту регістра чи комірки пам'яті з вмісту акумулятора без зміни вихідних даних, тобто вмісту порівнюваних операційних елементів (комірка пам'яті, регістр, акумулятор) після виконання зазначеної операції.

У системі команд МП КР580 передбачено три типи команд порівняння:

– порівняння вмісту акумулятора і регістра;

– порівняння вмісту акумулятора і комірки пам'яті;

– порівняння вмісту акумулятора з безпосереднім операндом.

Команди порівняння виконуються за допомогою внутрішнього віднімання з вмісту акумулятора відповідно, вмісту регістра, комірки пам'яті чи безпосереднього операнду. Вміст акумулятора при цьому не змінюється . Результатом порівняння є установка бітів ознак . Можливі варіанти установки ознак приведені в таблиці 8.1.

Таблиця 8.1

Результат порівняння	Ознаки	
	Нуль	Перенос
Дорівнює	1	0
Більше	0	0
Менше	0	1

Крім перерахованих ознак установлюються так само біти парності і знаку за результатами внутрішнього віднімання. Біт парності дорівнює 1 , у випадку коли кількість одиниць у результаті парна, і дорівнює 0 , якщо кількість одиниць непарна.

Біт знаку встановлюється рівним значенню старшого розряду результату. Це дозволяє поставити процес виконання програми в залежність від значення результату виконання попередньої операції. Для звертання до інформації про результати обчислень у складі МП існує набір тригерів , що складають програмно-доступний регістр (прапорів) F , що встановлюються в одиницю чи скидаються в нуль у залежності від результату зроблених обчислень .Ознаки формуються за результатами арифметичних чи логічних операцій.

Кожен тригер зберігає один з бітів умов. Програмно перевіряються значення чотирьох бітів умов:

а) S – ознака результату (1 – мінус, 0 – плюс);

б) Z – ознака нульового результату (1 – нульовий, 0 – не нульовий);

в) C – ознака переносу (1 – перенос є, 0 – переносу немає);

г) P – ознака паритет, парність (1 – число одиниць парне, 0 – непарне).

П'ятою ознакою є додатковий перенос AC з молодшої тетради в старшу при обробці двійково-десяткових чисел, які використовується разом з командою DAA.

Розташування ознак у розрядах PO таке:

F	S	Z	0	AC	0	P	1	C
	7	6	5	4	3	2	1	0

Перевірка ознак результату здійснюється в двійковий формі представлення числа вмісту регістра F.

8.2 Команди порівняння з вмістом регістра

Команди порівняння з вмістом регістра такі:

CMP A (BF), CMP B (B8), CMP C (B9), CMP D (BA), CMP E (BB), CMP H (BC), CMP L (BD).

Як приклад, рекомендується виконати такі завдання:

– записати в пам'ять коди програми порівняння вмісту регістрів C і B;

– виконати програму. попередньо задаючи вихідні значення відповідно до таблиці 3.3, командою <СТ> 800 <–> 802 <ВП>;

Таблиця 8.2

800	MOV A, C	79	Пересилання A ← C
801	CMP B	B8	Порівняння A з регістром B

Таблиця 8.3

Вихідні дані	C	Ваш зріст
	B	Ваш зріст
Результат	F	
	Нуль	
	Перенос	
	Знак	
	Парність	

– для перевірки зазначених у таблиці ознак вміст F переводиться з шістнадцяткової у двійкову форму представлення і фіксується вміст потрібних бітів.

8.3 Команда порівняння з вмістом комірки пам'яті

Команда порівняння з вмістом комірки пам'яті є такою:

CMP M (код BE) – порівняння вмісту регістра A і комірки пам'яті, адреса якої задана у парі HL.

Як приклад, рекомендується виконати такі завдання:

– записати в пам'ять коди програми порівняння вмісту регістра A і комірки пам'яті:

Таблиця 8.4

800	LXI H, 900H	21 00 09	Завантаження в HL. адреси 900
803	CMP M	BE	Порівняння A і M

– виконати програму, попередньо задаючи вихідні значення відповідно до таблиці 8.5, командою <СТ> 800 <–> 804 <ВП>;

– перевірити отримані результати.

Таблиця 8.5

Дані	A	21
	M	Ваш зріст
Результат	F	
	Нуль	
	Перенос	
	Знак	
	Парність	

8.4 Команда порівняння з безпосереднім операндом

Команда порівняння з безпосереднім операндом має такий вигляд:

– CPI B (код FE) – порівняння вмісту регістра A з числом, заданим у другому байті команди, де B – байт.

Як приклад, рекомендується виконати такі завдання:

– записати в пам'ять коди команди порівняння з безпосереднім операндом;

Таблиця 8.6

800	CPI 7F H	FE 7FH	порівняння 7FH з A

– виконати програму, попередньо задаючи вихідні значення відповідно до таблиці 8.7, за допомогою команди <СТ> 800 <–> 802 <ВП>;

– перевірити отримані результати.

Таблиця 8.7

Дані	A	Ваш зріст
Результат	F	
	Нуль	
	Перенос	
	Знак	
	Парність	

8.5 Зміст звіту

Звіт повинен містити:
- назву та мету роботи;
- написані програми відповідно до завдань 3.3;
- висновки.

8.6 Питання для самоперевірки

8.6.1 Особливості операції порівняння, її практична цінність.
8.6.2 Еквівалентні операції порівняння та їх недоліки.
8.6.3 Результати порівняння та їх визначення.
8.6.4 Перелічити ознаки з можливими значеннями бітів і короткою характеристикою.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. https://uk.wikipedia.org/wiki/Арифметико-логічний_пристрій
2. https://led-stars.com.ua/ua/g5876122-155-seriya-mikroshem
3. https://org2.knuba.edu.ua/mod/book/tool/print/index.php?id=32482
4. https://uk.wikipedia.org/wiki/Комп%27ютер
5. https://uk.wikipedia.org/wiki/Лічильник_імпульсів
6. https://uk.wikipedia.org/wiki/Логічні_елементи
7. https://uk.wikipedia.org/wiki/Мікроконтролер
8. https://uk.wikipedia.org/wiki/Мікропроцесор
9. https://uk.wikipedia.org/wiki/Мікросхема
10. https://uk.wikipedia.org/wiki/ПЛІС
11. https://uk.wikipedia.org/wiki/Регістр_(цифрова_техніка)
12. https://uk.wikipedia.org/wiki/Суматор
13. https://uk.wikipedia.org/wiki/Тригер
14. https://uk.wikipedia.org/wiki/Цифровий_компаратор
15. https://uk.wikipedia.org/wiki/Шифратор
16. Lecture OMPT.pdf [Електронний ресурс] – Режим доступу: https://moodle.zp.edu.ua/mod/assign/view.php?id=96396
17. Бойко, В.І. Основи схемотехніки електронних систем підручник [Текст]:/ Бойко В.І., Гуржій А.М., Жуйков В. Я. та ін. – К.: Вища шк., 2004. – 527 с.: іл.
18. Болюх, В. Ф. Основи електроніки та мікропроцесорної техніки: навч. посібник / В. Ф. Болюх, В. Г. Данько. – Харків : НТУ "ХПІ", 2011. – 257 с. [Електронний ресурс] – Режим доступу: https://repository.kpi.kharkov.ua/items/b8236fc7-4f0c-4ab2-b925-f8f0eafd9f3d
19. Бондаренко, І.М. Мікропроцесорні системи контролю та керування: Навч. посібник для студентів ЗВО / І. М. Бондаренко, О. В. Бородін, В. П. Карнаушенко. − Харків: ХНУРЕ, 2020. – 244 с.
20. Грищук, Ю. С. Мікропроцесорні пристрої: Навчальний посібник. – Харків НТУ «ХПІ», 2007. – 280 с. [Електронний ресурс] – Режим доступу: http://web.kpi.kharkov.ua/ea/wp-content/uploads/sites/25/2013/04/Mikroprotsesorni-pristroyi.pdf
21. Електронний посібник з дисципліни «Мікропроцесорні системи». [Електронний ресурс] – Режим доступу: https://ms.ptngu.com/index.html

22. Колонтаєвський, Ю. П. Конспект лекцій з дисципліни «Мікропроцесорна техніка» (для студентів, які навчаються за напрямом 6.050701 – Електротехніка та електротехнології всіх форм навчання) / Ю. П. Колонтаєвський. – Харків: ХНУМГ ім. О. М. Бекетова, 2016. – 78 с.
23. Лекції ОМРТ 2022.pdf [Електронний ресурс] – Режим доступу: https://moodle.zp.edu.ua/mod/assign/view.php?id=96396
24. Методичні вказівки до виконання лабораторних робіт студентами всіх форм навчання при вивченні дисципліни «Основи мікропроцесорної техніки» для підготовки бакалаврів за спеціальністю 141 «Електроенергетика, електротехніка та електромеханіка» з подальшим навчанням за освітніми програмами «Електричні та електронні апарати» та «Електромеханічне обладнання енергоємних виробництв». Частина 1 / уклад.: Л. Б. Жорняк, М. В Антонова. Запоріжжя: ЗНТУ, 2019. – 72 с. [Електронний ресурс] – Режим доступу: http://eir.zntu.edu.ua/handle/123456789/4107
25. Методичні вказівки до виконання лабораторних робіт студентами всіх форм навчання при вивченні дисципліни «Основи мікропроцесорної техніки» для підготовки бакалаврів за спеціальністю 141 «Електроенергетика, електротехніка та електромеханіка» з подальшим навчанням за освітніми програмами «Електричні та електронні апарати» та «Електромеханічне обладнання енергоємних виробництв». Частина 2 / Л. Б. Жорняк, М. В Антонова. Запоріжжя: ЗНТУ, 2019. – 38 с. [Електронний ресурс] – Режим доступу: http://eir.zntu.edu.ua/handle/123456789/4108
26. Методичні вказівки до виконання лабораторних робіт студентами з англійською мовою навчання при вивченні дисципліни «Основи мікропроцесорної техніки» для підготовки бакалаврів за спеціальністю 141 «Електроенергетика, електромеханіка та електромеханіка» з подальшим навчанням за освітньою програмою «Електричні машини і апарати » / Л. Б. Жорняк, Г. А. Рябенко - Запоріжжя : ЗНТУ, 2016. – 66 с. [Електронний ресурс] – Режим доступу: http://eir.zntu.edu.ua/handle/123456789/3330
27. Мікропроцесор [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Мікропроцесор

28. Мікропроцесорна техніка: Навчальний посібник з дисципліни для всіх форм навчання та студентів іноземців напряму підготовки 6.050701 «Електротехніка та електротехнології»/ В. В. Кирик.-К.: ІВЦ «Видавництво «Політехніка», 2014.- 183 с.
29. Огородник, К. В. Мікропроцесорна техніка: навчальний посібник / К. В. Огородник, Б. П. Книш. – Вінниця: ВНТУ, 2018. – 106 с.
30. Основи інформатики. Складові частини мікропроцесорних систем [Електронний ресурс] – Режим доступу: https://elprivod.nmu.org.ua/ua/interesting/components_mp_systems.php
31. Основи мікропроцесорної техніки. [Електронний ресурс] – Режим доступу: http://vozom.ho.ua/MP/page11.html
32. Основні терміни. Структура та функціонування мікропроцесорної системи [Електронний ресурс] – Режим доступу: https://studfile.net/preview/7075014/page:3/
33. Поджаренко, В. О. Основи мікропроцесорної техніки. Навчальний посібник. / В. О. Поджаренко, В. Ю. Кучерук, В. М. Севастьянов – Вінниця: ВНТУ, 2006. – 226 с. [Електронний ресурс] – Режим доступу: https://core.ac.uk/download/pdf/52159035.pdf
34. Структура і принципи роботи мікропроцесорної системи [Електронний ресурс] – Режим доступу: https://stud.com.ua/28301/tovaroznavstvo/struktura_printsipi_roboti_mikroprotsesornoyi_sistemi
35. Терлецький, А. І. Будова та програмування 8-розрядного мікропроцесора. Методичні рекомендації до виконання лабораторних робіт з дисципліни «Архітектура комп'ютерів» (2-й семестр) для студентів напряму «Комп'ютерна інженерія» / А. І. Терлецький, О. Б. Фрик. − Івано-Франківськ, 2012. − 88 с.
36. Ткачов, В.В. Мікропроцесорна техніка: навч. посібник/В.В. Ткачов, Г. Грулер, Н. Нойбергер та ін. – Д.: Національний гірничий університет, 2012. – 188 с. [Електронний ресурс] – Режим доступу: https://library.kre.dp.ua/Books/2-4%2.pdf

ДОДАТОК А

ПЕРША ТЕТРАДА	ДРУГА ТЕТРАДА																	
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	0	NOP	LXI BC	STAX BC	INX BC	INR B	DCR B	MVI B	RLC	--	DAD BC	LDAX BC	DCX BC	INR C	DCR C	MVI C	RRC	0
	1	--	LXI DE	STAX DE	INX DE	INR D	DCR D	MVI D	RAL	--	DAD DE	LDAX DE	DCX DE	INR E	DCR E	MVI E	RAR	1
	2	--	LXI HL	SHLD HL	INX HL	INR H	DCR H	MVI H	DAA	--	DAD HL	LHLD HL	DCX HL	INR L	DCR L	MVI L	CMA	2
	3	--	LXI SP	STA adr	INX SP	INR M	DCR M	MVI M	STC	--	DAD SP	LDA adr	DCX SP	INR M	DCR M	MVI A	CMC	3
	4	MOV B, B	MOV B, C	MOV B, D	MOV B, E	MOV B, H	MOV B, L	MOV B, M	MOV B, A	MOV C, B	MOV C, C	MOV C, D	MOV C, E	MOV C, H	MOV C, L	MOV C, M	MOV C, A	4
	5	MOV D, B	MOV D, C	MOV D, D	MOV D, E	MOV D, H	MOV D, L	MOV D, M	MOV D, A	MOV E, B	MOV E, C	MOV E, D	MOV E, E	MOV E, H	MOV E, L	MOV E, M	MOV E, A	5
	6	MOV H, B	MOV H, C	MOV H, D	MOV H, E	MOV H, H	MOV H, L	MOV H, M	MOV H, A	MOV L, B	MOV L, C	MOV L, D	MOV L, E	MOV L, H	MOV L, L	MOV L, M	MOV L, A	6
	7	MOV M, B	MOV M, C	MOV M, D	MOV M, E	MOV M, H	MOV M, L	HLT	MOV M, A	MOV A, B	MOV A, C	MOV A, D	MOV A, E	MOV A, H	MOV A, L	MOV A, M	MOV A, A	7
	8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC C	ADC	ADC E	ADC H	ADC L	ADC M	ADC A	8
	9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A	9
	A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A	A
	B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A	B
	C	RNZ	POP BC	JNZ adr	JMP adr	CNZ adr	PUSH BC	ADI data	RST0	RZ	RET	JZ adr	--	CZ adr	CALL adr	ACI data	RST1	C
	D	RNC	POP DE	JNC adr	OUT port n	CNC adr	PUSH DE	SUI data	RST2	RC	--	JC adr	IN port n	CC adr	--	SBI data	RST3	D
	E	RPO	POP HL	JPO adr	XTHL	CPO adr	PUSH HL	ANI data	RST4	RPE	PCHL	JPE adr	XCHG	CPE adr	--	XRI data	RST5	E
	F	RP	POP psw	JP adr	DI	CP adr	PUSH psw	ORI data	RST6	RM	SPHL	JM adr	EI	CM adr	--	CPI data	RST7	F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	