МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з курсу

“ЕОМ І МІКРОПРОЦЕСОРНІ СИСТЕМИ”

для студентів
спеціальності 8.080403
“Програмне забезпечення автоматизованих систем
усіх форм навчання

2005

Методичні вказівки до лабораторних робіт з курсу “ЕОМ і мікропроцесорні системи” для студентів спеціальності 8.080403 “Програмне забезпечення автоматизованих систем усіх форм навчання /Укл.: С.К.Корнієнко, Л.П.Скачко. – Запоріжжя: ЗНТУ, 2005. - 54 с.

Укладачі:

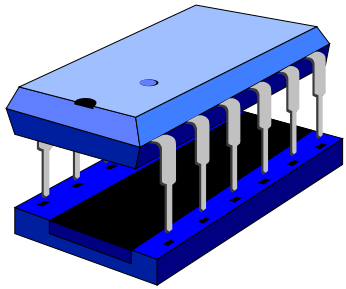
С.К.Корнієнко, доцент, к.т.н.,
Л.П.Скачко, асистент

Рецензент: Оніщенко В.Ф., доцент, к.ф.-м.н

Відповідальний за випуск: А.В.Притула

Затверджено
на засіданні кафедри ПЗ

Протокол № 6 від 10.03.2005.



ЗМІСТ

Лабораторна робота №1 “Вивчення архітектури типової МПС на базі мікропроцесора I8080” 4

Лабораторна робота №2 “Вивчення принципів реалізації лінійних програм” 6

Лабораторна робота №3 “Вивчення принципів реалізації розгалужених та циклічних програм” 9

Лабораторна робота №4 “Дослідження принципів програмної реалізації логічних функцій”....11

Лабораторна робота №5 "Вивчення принципів реалізації програм управління зовнішніми пристроями" 14

Лабораторна робота № 6 "Розробка програми управління технологічними процесами за допомогою графічного інтерпретатора портів виводу" 18

Додаток А Архітектура типової системи на базі мікропроцесора I8080...21

Додаток Б - Емулятор мікропроцесорної системи 26

Додаток В – Система команд мікропроцесора I8080 33

Додаток Д - Призначення та склад регістра ознак 39

Додаток Ж - Способи організації часових затримок....40

Додаток К - Приклад реалізації логічної функції 43

Додаток Л Графічний інтерпретатор портів виводу 46

Додаток М Варіанти завдань для самостійної роботи 54

Лабораторна робота №1 “Вивчення архітектури типової МПС на базі мікропроцесора I8080”

1 Мета роботи

Метою роботи є ознайомлення з архітектурою мікропроцесора I8080 та типової мікропроцесорної системи (МПС) на його базі, а також отримання навичок практичної роботи з програмою-емулятором, вивчення принципів складання та вводу найпростіших програм до оперативної пам’яті МПС і способу їхньої корекції.

2 Завдання до лабораторної роботи

2.1 Вивчити архітектуру мікропроцесора i8080 і мікропроцесорної системи на його основі.

2.2 Вивчити структуру емулятора мікропроцесорної системи (ЕМПС) та призначення його основних команд.

2.3 Увести програму та перевірити її виконання.

3 Методичні вказівки до виконання лабораторної роботи

3.1 Вказівки до виконання пп. 2.1.

Теоретичний матеріал за п. 2.1 завдання стисло викладений у Додатку А даних методичних вказівок.

3.1 Вказівки до виконання пп. 2.2 завдання.

Відомості про призначення, основні функції та склад команд емулятора МПС викладено у Додатку Б.

3.2 Вказівки до виконання п. 2.3 завдання.

Для виконання п. 2.3 завдання необхідно отримати у викладача текст програми, ввід якої здійснити у такій послідовності.

- запустити програму ЕМПС.
- вибрати в пункті меню Файл підменю ***Новий***, щоб створити нове вікно для вводу програми.

4 Зміст звіту

Звіт повинен мати наступні матеріали:
– назва та мета роботи;
– структурна схема МПС;
– текст програми.

5 Контрольні запитання

- ✓ Які типи МПС існують?
- ✓ Укажіть склад і функції МПС.
- ✓ Сформулюйте основні етапи створення програмної частини МПС.
- ✓ Архітектура МП i8080.
- ✓ Яка послідовність налагодження програми та її запуску?
- ✓ Основні функції програми-емулятора МПС.

Лабораторна робота №2 “Вивчення принципів реалізації лінійних програм”

1 Мета роботи

Метою роботи є вивчення групи команд організації пересилань, арифметичних операцій, вводу-виводу даних, роботи з ОЗП, організації роботи зі стеком та придбання практичних навичок у складанні та налагодженні простих лінійних програм.

2 Завдання на лабораторну роботу

2.1 Вивчити призначення та склад внутрішніх регістрів мікропроцесора та формат команд.

2.2 Вивчити позначення і зміст команд, що входять до групи команд пересилань даних, арифметичних операцій, вводу-виводу даних, звертання до ОЗП та роботи зі стеком.

2.3 Скласти та реалізувати програму складання (віднімання) двох чисел з використанням ОЗП.

2.4 Скласти та реалізувати програму обробки даних з використанням ОЗП.

2.5 Скласти та реалізувати програму обміну даними між регістрами з використанням стекової пам’яті.

Примітка:
при складанні програми використовувати різні методи адресації з метою порівнювального аналізу та вибору оптимального варіанту.

3 Методичні вказівки до виконання лабораторної роботи

Для виконання завдань лабораторної роботи необхідно передчасно ознайомитися зі змістом лабораторної роботи №1.

3.1 Вказівки до виконання п.2.1 завдання

Для виконання п.2.1 лабораторного завдання необхідно ознайомитися з Додатком А методичних указівок.

3.2 Вказівки до виконання п.2.2 завдання

Виконання п.2.2 вимагає вивчення вказаних у завданні груп команд згідно табл.В.1 Додатку В.

3.3 Вказівки до виконання п.2.3 завдання

По отриманому у викладача завданню скласти, ввести та налагодити програму, користуючись вказівками попередньої лабораторної роботи, а також даними додатків А-В.

3.4 Вказівки до виконання п.2.4 завдання

Виконання п.2.4 завдання здійснюється в послідовності, аналогічній п.2.3.

3.5 Вказівки до виконання п.2.5 завдання

Реалізацію обміну між регістрами з використанням стека необхідно здійснювати в такій послідовності.

– сформувати стек, для цього до парного регістру SP завантажити константу, що дорівнює адресі чарунки ОЗП, яка є вершиною стека.

– виконати пересилку змісту РЗП до стека у необхідному порядку, а потім завантажити РЗП інформацією зі стека. В обміні між РЗП та стеком приймають участь пари регістрів. При цьому переміщення стекової інформації відбувається автоматично, реалізуючи принцип стекової організації "останнім прийшов – першим вийшов".

– виконати перевірку змісту регістрів до та після обміну інформацією.

4 Зміст звіту

Звіт повинен включати наступні матеріали:

- мету роботи;
- тексти програм по пп.2.3, 2.4, 2.5;

- схеми алгоритмів програм;
- аналіз отриманих результатів та висновки.

5 Контрольні запитання

- ✓ Назвіть склад внутрішніх регістрів ЦПЕ.
- ✓ Що таке формат команди?
- ✓ Яке призначення окремих груп команд?
- ✓ Який зміст основних команд?
- ✓ Як виконується ввід (вивід) інформації у МПС?
- ✓ Як здійснюється адресація чарунок ОЗП?
- ✓ Що таке стек, яким чином він організується?

Лабораторна робота №3 “Вивчення принципів реалізації розгалужених та циклічних програм”

1 Мета роботи

Метою роботи є вивчення групи команд, які використовуються при організації переходів, формуванні лічильників, операціях зсувів та надбання практичних навичок у використанні регістра ознак при складанні й налагодженні розгалужених та циклічних програм.

2 Завдання на лабораторну роботу

2.1 Вивчити призначення та склад регістра ознак.

2.2 Вивчити позначення і зміст команд, що входять до групи зсувів, переходів та інкремент-декремент.

2.3 Скласти та реалізувати програму обчислення складної функції.

2.4 Скласти та реалізувати програму формування часових інтервалів.

3 Методичні вказівки щодо виконання лабораторної роботи

Виконання даної лабороторної роботи базується на знаннях, отриманих під час виконання попередніх робот.

3.1 Вказівки до виконання пп.2.1 завдання

Для виконання пп.2.1 завдання необхідно вивчити матеріал Додатку Д.

3.2 Вказівки до виконання пп.2.1 завдання

Для виконання пп.2.2 необхідно вивчити вказані у завданні групи команд за таблицею В.1 Додатку В. При цьому потрібно зосередити особливу увагу на графу 3 таблиці В.1, в якій наведено час виконання різноманітних команд у машинних тактах. Час одного машинного такту ***t =0,5 мкс.***

3.3 Вказівки до виконання п.2.3 завдання

За отриманим у викладача завданням скласти схему реалізації заданої функції. Текст програми необхідно оформляти у вигляді таблиці, в якій повинні бути передбачені графи для адреси ОЗП (ПЗП), шістнадцяткового коду команди, мітки, мнемонічних позначень команд та коментарів.

Результати обчислень у точках, вказаних викладачем, а також кінцевий результат необхідно фіксувати в протоколі виконання лабораторної роботи.

3.4 Вказівки до виконання п.2.4 завдання

Для виконання п.2.4 завдання необхідно отримати у викладача завдання, у відповідності з яким визначити варіант реалізації алгоритму (Додаток Ж). Розробити необхідні розрахунки, враховуючи час виконання команд (див. графу 3 табл.В.1) і варіанти реалізації. Скласти детальну схему та текст програми у відповідності зі вказівками по п.3.2.

4 Зміст звіту

Звіт повинен мати наступні матеріали.

- мету роботи.
- завдання, схему алгоритму та текст програми, результати розрахунків за п.2.3 завдання.
- завдання, необхідні розрахунки, схему, текст програми і результати експерименту за п.2.4.
- аналіз отриманих результатів та висновки.

5 Контрольні запитання

- ✓ Що таке регістр ознак?
- ✓ Як реалізуються команди умовних переходів?
- ✓ Як організується лічильник циклів?
- ✓ Яке призначення розгалужених та циклічних програм?

Лабораторна робота №4 “Дослідження принципів програмної реалізації логічних функцій”

1 Мета роботи

Метою роботи є вивчення групи команд логічних операцій і принципів програмної реалізації логічних функціональних вузлів.

2 Завдання на лабораторну роботу

2.1 Вивчити позначення та склад команд, що входять до групи логічних операцій.

2.2 Проаналізувати задану принципову схему цифрового пристрою, скласти логічну функцію та таблицю істинності.

2.3 Скласти схему алгоритму, що реалізує логічну функцію.

2.4 Скласти текст і налагодити робочу програму.

2.5 Виконати експериментальну перевірку таблиці істинності.

3 Методичні вказівки до виконання лабораторної роботи

Для виконання завдання по даній роботі необхідно згадатиі основні відомості з цифрової техніки й булевої алгебри та виконати лабораторні роботи №1,2,3.

3.1 Вказівки до виконання п.2.1 завдання

Для виконання завдання за п.2.1 необхідно вивчити відповідний розділ таблиці В.1 Додатку В. Потрібно зосередити увагу на таблицю істинності логічної операції “виключне АБО” (команди ***XRA r, XRI***).

3.2 Вказівки до виконання п.2.2 завдання

Отримав у викладача принципову схему цифрового приладу, необхідно зрозуміти, які логічні функції повинні бути реалізовані, використовуючи правила булевої алгебри. Потім, використовуючи таблиці істинності основних операцій булевої алгебри, скласти таблиці істинності отриманої логічної функції (див. додаток К).

3.3 Вказівки до виконання п.2.3 завдання

Складання схеми, що реалізує логічну функцію, необхідно виконувати, використовуючи таблиці істинності основних логічних операцій “**І**”, “**АБО**”, “**НІ**”. При цьому складання схеми алгоритму рекомендується починати зі здійснення перевірки на рівність одиниці кожної частини логічної функції (див. додаток К).

3.4 Вказівки до виконання п.2.4 завдання

3.4.1 Розробку програми маємо проводити таким чином, щоб після отримання кожної частини логічної функції прийняття рішення про рівність або нерівність одиниці відбувалося згідно зі станом того чи іншого прапорця регістра ознак.

3.4.2 Необхідно пам’ятати, що ввід інформації про значення незалежних змінних X_i здійснюється побайтно. Отже, потрібно замаскувати “зайві” розряди тобто ті, що не приймають участі у формуванні значення функції. У випадку, коли цей байт інформації приймає участь у формуванні значень других частин функції, його треба зберігати в одному з РЗП (див. Додаток К).

3.5 Вказівки до виконання п.2.5 завдання

Виконання п.2.5 завдання здійснюється за методикою, яка описана у попередніх роботах.

4 Зміст звіту

Звіт повинен мати наступні матеріали.

- мету роботи;
- завдання на лабораторну роботу;
- отриману логічну функцію, яка реалізує заданий пристрій;
- схему алгоритму;
- текст програми;
- таблиці істинності (теоретичну та експериментальну).
- аналіз отриманих результатів та висновки.

5 Контрольні запитання

- ✓ Який зміст команд, що входять до групи логічних операцій? Таблиці істинності логічних операцій.
- ✓ Яким чином складається таблиця істинності логічної функції?
- ✓ У чому полягає принцип програмної реалізації логічних функцій?
- ✓ Для чого і яким чином проводиться маскування?
- ✓ Як здійснюється інвертування потрібних розрядів у байті інформації?
- ✓ У чому перевага програмного способу реалізації логічних функцій у зрівнянні з апаратним?

Лабораторна робота №5 "Вивчення принципів реалізації програм управління зовнішніми пристроями"

1 Мета роботи

Метою роботи є вивчення прийому блочної реалізації програм із використанням команд ***CALL*** та ***RET*** на прикладі програми керування портами індикації.

2 Завдання на лабораторну роботу

2.1 Вивчити позначення та зміст команд ***CALL*** та ***RET*** за різноманітними умовами, що входять до групи команд переходів.

2.2 Скласти схему алгоритму, який дозволяє на вихідних портах отримати “бігаючі вогні”.

2.3 Реалізувати програму “бігаючі вогні” з керуванням із портів **00** і **01.**

2.4 Внести необхідні зміни у програму для сумісного використання індикаторів **00** і **01.**

3 Методичні вказівки до виконання лабораторної роботи

3.1 Вказівки до виконання п.2.1 завдання

Для виконання п.2.1 лабораторного завдання необхідно вивчити групу команд переходів за таблицею В.1 Додатку В.

Потрібно звернути увагу, що трибайтна команда ***CALL*** є командою звернення до підпрограми, яка починається з деякої адреси. За командою ***CALL*** зміст лічильника команд центрального процесора, тобто адреса наступної команди, заноситься до стекової області ОЗП, а зміст покажчика стека зменшується на 2. Після операцій зі стеком до лічильника команд заноситься початкова адреса підпрограми, вказана другим та третім байтами команди ***CALL***, після чого починається її виконання.

У кінці підпрограми обов’язково повинна стояти команда ***RET***. Команда ***RET*** відновлює той стан процесора й стека, який був

безпосередньо перед виконанням команди ***CALL***. Зміст покажчика стека збільшується на 2, а до лічильника центрального процесора заноситься адреса, що була передана для зберігання до стекової пам'яті за командою ***CALL***.

3.2 Вказівки до виконання п.2.2 завдання

Працююча програма повинна реагувати на такі керуючі впливи, що задаються шляхом установлення відповідних розрядів **Bi** порту **01** в одиницю (рисунок 5.1):

"***ПУСК-СТОП***", тобто при $\mathbf{B_7}$=1 – переміщення інформації на порту індикації (вогні "бігуть"), а при $\mathbf{B_7}$=0 – немає переміщення;

"***ВЛІВО-ВПРАВО***", тобто при $\mathbf{B_6}$=1 – переміщення інформації вліво, а при $\mathbf{B_6}$=0 – вправо;

"***ШВИДКІСТЬ***" – розряди $\mathbf{B_0}$-$\mathbf{B_5}$ повинні дозволяти в широких межах змінювати швидкість переміщення інформації (час переносу інформації на один розряд від 0,05с до 3-4с);

"***ВВІД ІНФОРМАЦІЇ***" – в положенні "***СТОП***" інформація, яка вводиться повинна задаватися з порту **00.**

3.3 Вказівки до виконання п.2.3 завдання.

Оскільки режим роботи програми визначають старші розряди порту **B** ($\mathbf{B_7}$ і $\mathbf{B_6}$), їхній стан доцільно аналізувати за прапорцем перенесення, здійснив попередньо зсув байта, що аналізується, вліво.

Константа, що визначає час затримки між зсувами ("**ШВИДКІСТЬ**"), легко отримується в акумуляторі з використанням відповідної маски (див. лабораторну роботу №4).

"Бігаючі вогні" отримуються регулярним оновленням інформації на порту індикації. Час затримки визначається розрядами $\mathbf{B_0}$-$\mathbf{B_5}$ порту **B** (див. вище). Оновлення інформації полягає в її зсуві в акумуляторі на один розряд уліво або вправо. Після кожного зсуву інформацію необхідно запам'ятовувати в одному із РЗП і для наступного зсуву знов використовувати акумулятор. Для спрощення процесу налагодження програми керування індикатором доцільно виділити дві підпрограми:

перша – підпрограма мінімальної затримки t_{min}=10мкс;

друга – підпрограма змінної затримки з кроком t_{min};

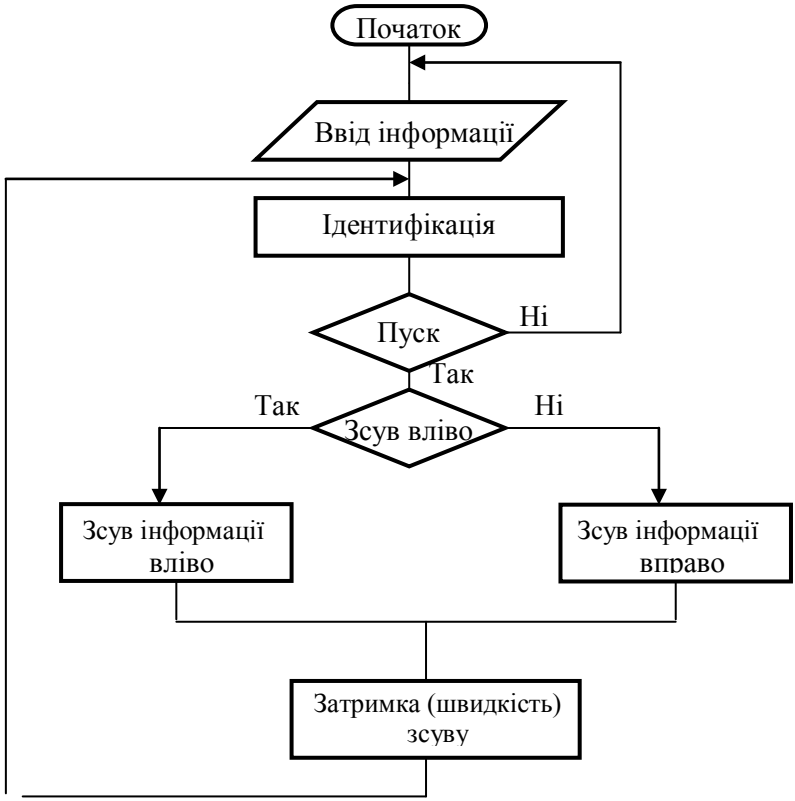


Рисунок 5.1 - Схема алгоритму керування портом виводу

3.4 Вказівки до виконання п.2.4 завдання

Зміни в програмі насамперед стосуються вводу інформації та її виводу на індикатори. При цьому треба пам'ятати, що операції ***IN*** та ***OUT*** використовують однобайтовий регістр-акумулятор. Зсув інформації в двобайтовому регістрі вліво доцільно здійснювати за допомогою команди ***DAD***, а вправо – комбінуючи команди зсуву через прапорець перенесення та пересилання між регістрами.

4 Зміст звіту

Звіт повинен мати наступні матеріал:
- мету роботи;
- позначення та зміст команд звернення до підпрограми і повернення з неї, команд циклічного зсуву;
- схему алгоритму програми керування;
- текст програми керування індикатором;
- текст змін програми для одночасного використання двох портів R та L ;
- аналіз отриманих результатів та висновки.

5 Контрольні запитання

- ✓ Який зміст команд ***CALL*** і ***RET***?
- ✓ Який зміст команд циклічного зсуву?
- ✓ Що таке адреса повернення? Де вона зберігається і як використовується?
- ✓ Як може виконуватися аналіз окремих бітів слова?
- ✓ У чому полягає різниця та схожість прийомів зсуву однобайтової та двобайтової інформації?
- ✓ Як реалізується зміна величини затримки операцією з порту вводу?

Лабораторна робота № 6 "Розробка програми управління технологічними процесами за допомогою графічного інтерпретатора портів виводу"

1 Мета роботи

Метою даної роботи є одержання навичок складання програм керування технологічними процесами за допомогою графічного інтерпретатора портів виводу (ГІПВ).

2 Завдання до лабораторної роботи

2.1 Вивчити основні функції ГІПВ (таблиця Л.1).
2.2 Вивчити константи для завдання кольорів (таблиця Л.2).
2.3 Одержати завдання у викладача.
2.4 Скласти та запустити програму на виконання.

3 Методичні вказівки до виконання лабораторної роботи

3.1 Вказівки до виконання п.2.1 завдання

Графічний інтерпретатор портів виводу описаний у Додатку Л. Основні функції ГІПВ наведені у таблиці Л.1.

3.2 Вказівки до виконання п.2.2 завдання

Робота з кольором (константи, що завдають колір) наведені у таблиці Л.2.

3.2 Вказівки до виконання п.п.2.3 та 2.4 завдання

Одержавши від викладача завдання на лабораторну роботу, треба спочатку структурувати майбутнє графічне зображення на окремі графічні об’єкти, а вже потіп приступати до реалізації програми, керуючись Додатком Л.

Робота з графічним інтерпретатором здійснюється через порт 05.

Значення, що надходять на цей порт, сприймаються (інтерпретуються) ГІПВ як деякі керуючі команди.

Кожна така команда займає 2 байти, тому може бути передана на виконання тільки за 2 кроки: спочатку на порт 05h посилається перший байт команди, а потім другий байт. Перший байт містить код команди, а другий байт - її параметри.

4 Зміст звіту

- Ціль роботи.
- Текст програми.
- Результат роботи програми.
- Висновки.

5 Контрольні запитання

- ✓ Для чого призначений ГІПВ?
- ✓ Чому інформація на ГІПВ передається в 2 прийоми?
- ✓ Які основні функції виконує ГІПВ?

Література

1 Гилмор Ч. Введение в микропроцессорную технику. – М.: Мир, 1989.

2 Григорьев В.Л. Программирование однокристальных микропроцессоров. – М: Энергоатомиздат, 1987.

3 Лихтциндер В.Я., Кузнецов В.Н. Микропроцессоры и вычислительные устройства в радиотехнике. – К.: Вища школа, 1988.

4 Майоров В.Г., Гаврилов А.И. Практический курс программирования микропроцессорных систем. – М.: Машиностроение, 1989.

5 Проектирование микропроцессорной электронно-вычислительной аппаратуры: Справочник / В.Г. Артюхов, А.А. Будняк, В.Ю. Лапий и др. – К.: Техника, 1988.

6 Самофалов К.Г., Викторов О.В. Микропроцессоры. – К.: Техника, 1989.

7 Справочник по микропроцессорным устройствам / А.А. Молчанов, В.И. Корнейчук, В.П. Тарасенко. – К.: Техника, 1987.

8 Сташин В.В. Проектирование цифровых устройств на однокристальных микропроцессорах. – М.: Энергоатомиздат, 1990.

9 Токхайм Р. Микропроцессоры: Курс и упражнения. – М.: Энергоатомиздат, 1988.

10 Шевкопляс Б.В. Микропроцессорные структуры. Инженерные решения: Справочник. – М.: Радио и связь, 1990

11 Щелкунов Н.Н., Дианов А.П. Микропроцессорные средства и системы. – М: Радио и связь, 1989.

Додаток А Архітектура типової системи на базі мікропроцесора І8080

Функції, які реалізуються мікропроцесором, визначаються його структурою (сукупністю елементів, які складають МП, та зв'язків між ними) і послідовністю управляючих слів (команд), що надходять із запам'ятовуючого пристрою на входи МП. Для комплексної характеристики можливостей МП використовують поняття архітектури.

Архітектура МП – це його логічна організація, яка визначається можливостями МП з апаратної та програмної реалізації функцій, необхідних для будування мікро-ЕОМ (МПС).

Поняття архітектури МП відображає його структуру, способи звертання до всіх доступних для користувача елементів структури, а також систему команд, режими адресації, формати та довжину даних, тобто архітектура будь-якої обчислювальної системи – це її представлення з точки зору користувача.

Структурна схема ЦПЕ представлена на рисунку А.1. До його складу входять:

- блок регістрів;
- блок Арифметико-логічного пристрою (АЛП);
- блок керування та синхронізації.

В свою чергу, блок регістрів складається з:

- регістрів загального призначення (РЗП) (***B, C, D, E, H, L***);
- регістрової пари ***W-Z;***
- програмного лічильника ***PC;***
- покажчика стека ***SP;***
- регістра адреси ***PA***.

До складу блока АЛП входять:

- сам АЛП;
- акумулятор ***A***;
- регістр ознак (прапорців) ***F;***
- буферні регістри ***БР1, БР2.***

Блок керування та синхронізації складається з:

- пристрою керування;
- регістра команд;
- дешифратора команд.

Внутрішні регістри мають наступне призначення:

A - регістр-акумулятор або накопичувач, призначений для збереження результату, а також здійснення вводу та виводу даних по командах ***IN*** та **OUT;**

РЗП - регістри загального призначення (**У**, **C**, **D**, **E**, **H**, **L**), які виконують роль надоперативної пам'яті ЦПЕ; можуть використовуватися як окремо, так і парами (тобто в якості 16-розрядних регістрів). В операціях звернень до пам'яті парний регістр HL використовується особливо: у ньому зберігається (записується) адреса ОЗП, по якій відбувається звернення;

PC - 16-розрядний програмний лічильник (лічильник команд), який визначає адресу ОЗП чи ПЗП, по якій зберігається чергова команда;

SP - 16-розрядний покажчик стека, який визначає поточну адресу вершини стека в ОЗП;

F - 8-розрядний регістр ознак (прапорців), розряди (прапорці) якого відповідають різноманітним ознакам і використовуються при виконанні операцій умовних переходів, умовного виклику підпрограм та умовного повернення з підпрограм. В результаті виконання окремих команд ознаки можуть змінюватися (встановлюватися в одиницю).

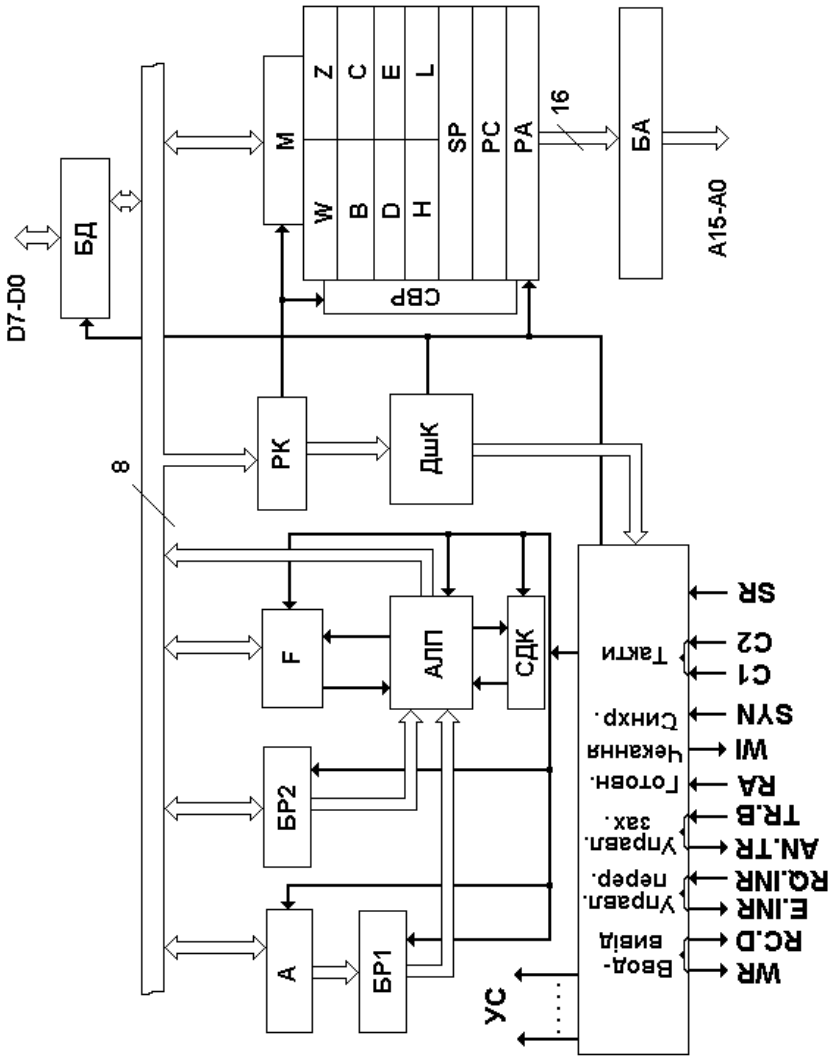


Рисунок А.1 - Структурна схема ЦПЕ І8080

Структурна схема МПС представлена на рисунку А.2 й містить мінімальну сукупність елементів, які повинні бути у складі будь-якої мікро-ЕОМ.

Типова МПС складається з наступних основних блоків:

- центрального процесорного елемента (**ЦПЕ**) І8080;
- системного контролера (**СК**);
- генератора тактових імпульсів (**ГТІ**);
- постійного запам'ятовуючого пристрою (**ПЗП**);
- оперативного запам'ятовуючого пристрою (**ОЗП**);
- портів уводу;
- портів виводу.

Системний контролер призначений для зберігання слова стану та виробки додаткових сигналів керування.

Генератор тактових імпульсів призначений для формування двох серій імпульсів синхронізації та сигналу скидання.

Постійний запам'ятовуючий пристрій зберігає незмінну в процесі роботи МПС інформацію. Працює тільки у режимі читання.

Оперативний запам'ятовуючий пристрій дозволяє змінювати інформацію. **ПЗП** і **ОЗП** разом складають основну пам'ять МПС.

Порти вводу-виводу - це пристрої, за допомогою яких інформація надходить до МПС або виводиться з неї. Тобто це пристрої для зв'язку МПС із зовнішнім світом.

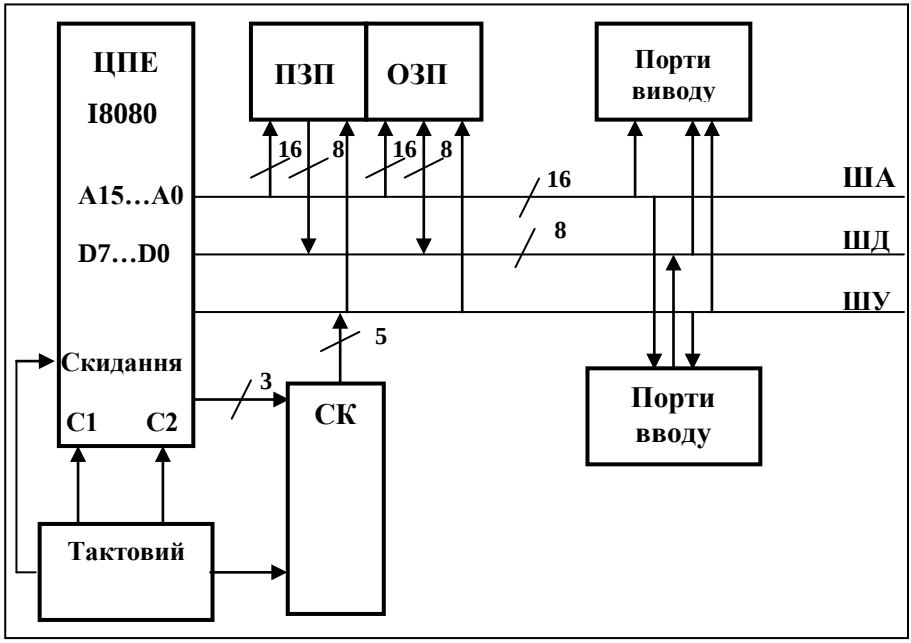


Рисунок А.2 - Структурна схема мікропроцесорної системи

Додаток Б - Емулятор мікропроцесорної системи

Емулятор мікропроцесорної системи (ЕМПС) - це програмний комплекс, призначений для моделювання роботи мікропроцесорних систем (МПС) на ПЕОМ старших поколінь та навчання студентів основам програмування на мові Асемблера мікропроцесора I8080. МПС, яка моделюється, складається з центрального процесорного елементу, ОЗП та портів вводу-виводу.

Крім того, програма містить графічний інтерпретатор портів виводу (ГІПВ), що імітує керування зовнішнім приладом за допомогою порту виводу 05. Програма містить детальну довідку як по роботі з емулятором, так і по командам мікропроцесора I8080, яка при запущеному додатку „Эмулятор CPU I8080” викликається натисканням кнопки F1.

ЕМПС виконує наступні функції:

- створення програм у текстовому редакторі та збереження їх до файлу з виконанням усіх функцій будь-якого текстового редактора: друк файлу, робота з блоками, пошук та заміна, тощо;
- компіляція та запуск програми користувача;
- покрокове виконання програми та встановлення точок зупинки у вказаних пунктах програми;
- можливість контролю та зміни стану процесора (РЗП, РС та SP), змісту ОЗП та портів вводу-виводу за допомогою повно-функціонального налагоджувача;
- перевірка синтаксису програми;
- робота з ГІПВ.

Редактор ЕМПС

На рисунку Б.1 наведений зовнішній вигляд редактора ЕМПС.

Текстовий редактор програми "Емулятор CPU I8080" містить в собі усі стандартні механізми, що потрібні для редагування тексту: копіювання, вставка та вирізання тексту. Заголовок вікна містить ім’я відкритого файлу. Символ * показує, що зміст поточного файла був змінений. В текст програми можна вставляти коментарі, що починаються з символу ";" та продовжуються до кінця строки.

При виборі пункту меню ***Сервис → Опции редактора...,*** настройки редактора можна змінити. У нижній частині редактора міститься вікно повідомлень компілятора. Користувач може змінювати розміри цього вікна, закривати його, а також очищувати список повідомлень вибором відповідного пункту спливаючого меню .

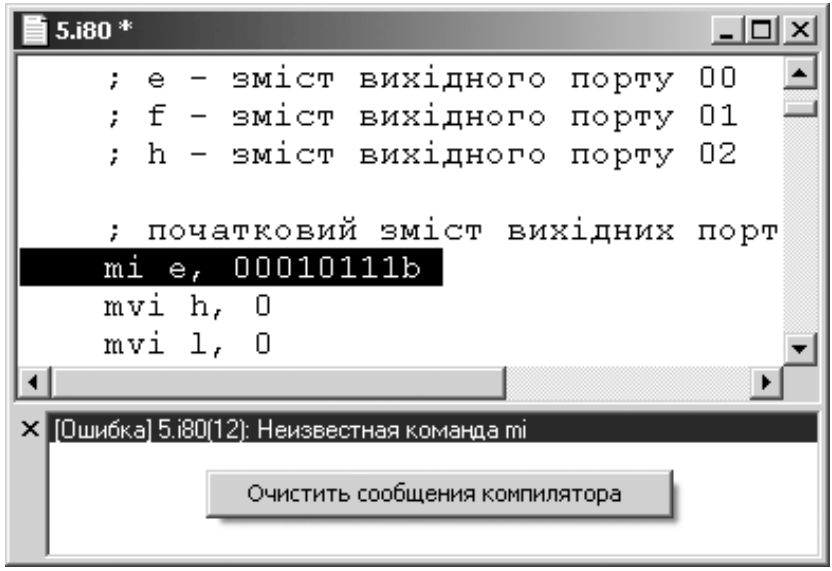


Рисунок Б.1 - Вікно Редактора

Команди редактора:

Ctrl+X – вирізання фрагменту до буфера обміну;
Ctrl+C – копіювання виділеного фрагменту до буфера обміну;
Ctrl+V – вставка змісту буфера обміну;
Ctrl+Del – вилучання фрагменту програми;
Ctrl+A – виділення усієї програми;
Ctrl+Z – відміна останньої операції.
Ctrl+F – пошук тексту у програмі;
Ctrl+R – заміна попередньо знайденого тексту.

Директиви компілятора:

ЕМПС має ряд додаткових директив, необхідних для вірної організації коду. Розглянемо ці директиви:

org адреса - встановлення адреси формування коду програми. Директив ORG в тілі програми може бути декілька.

мітка: equ значення - визначення констант. Мітці надається значення, вказане після директиви. При завданні значень констант можна використовувати як число, так і внутрішню змінну ЕМПС $, яка має значення поточної адреси. При використанні $, можна до $ додавати або віднімати деяке значення.

.startup - визначає точку входження до програми. У тілі програми може зустрічатися лише один раз.

include "ім'я файла" - підключення до тіла програми зовнішнього файла з текстом процедур або описів констант. Файл буде вставлений після описаної директиви. Мітки в файлах не повинні повторюватися, а також у допоміжних файлах не повинна зустрічатися директива ***.startup.***

[мітка] db значення - розміщує у пам'яті дані, що перелічені після директиви. Значення необхідно записувати через кому. Дані будуть розміщені послідовно, розмірність - байт.

[мітка] db N dup (значення) - заповнює ***N*** чарунок пам'яті заданим значенням. Якщо замість значення вказати "?", то ЕМПС зарезервує місце завдовжки ***N.***

Директива ***dw*** має такий синтаксис, як і ***db***, але розмірність - слово.

Значення внутрішньої змінної $ можна використовувати як константу при визначенні команд пересилки даних, а також у командах передачі управління. До $ можна як додавати, так і віднімати деяке значення, що не повинне перевищувати 65535.

Налагоджувач ЕМПС

Налагоджувач використовується для емуляції МПС I8080 (рис. Б.2). Він складається з 8 основних частин:

1. **Список команд.** Цей список поділений на 3 стовпця: адреса першого байта команди, код команди, а також сама команда.

Зелена стрілка вказує на чергову команду, що буде виконуватися. При виборі пункту меню ***Выполнить → Добавить точку зупину,*** натисканні клавіші ***F5*** або при натисканні мишею на першу чарунку потрібного рядку можна додавати або вилучати точку зупинки. Переміщуватися по списку можна за допомогою клавіш "↑" та "↓", а також за допомогою скролера.

2. **Зміст ОП.** Показує значення кожного байта ОП. Переміщуватися по списку можна тими ж способами, що й в попередньому списку.
3. **Такти та дозвіл переривань.** Відображає кількість тактів, на протязі яких виконувалася програма, а також значення сигналу дозволу переривань. Спливаюче меню в даній панелі містить пункт "Изменить INTE", що служить для зміни значення INTE.
4. **РЗП.** Панель відображає поточний стан регістрів. Вибираючи відповідний пункт спливаючого меню, можна змінювати формат відображення, збільшувати, зменшувати на одиницю, очищати, а також заносити власне значення до кожного регістра. Змінювати значення регістра можна також подвійним натисканням на відповідний регістр.
5. **Прапорці.** Панель відображає поточний стан прапорців. Вибираючи відповідний пункт спливаючого меню, можна змінювати значення кожного прапорця. Це також можна робити подвійним натисканням на відповідний регістр.
6. **Порти вводу.** Панель відображає значення останніх даних, що поступали на порти вводу. Спливаюче меню містить пункти для зміни даних, а також формату їх відображення на панелі. Переміщуватися за списком можна за допомогою клавіш "↑" та "↓", а також за допомогою скролера.
7. **Порти виводу.** Панель відображає значення останніх даних, що поступали на порти виводу. Спливаюче меню містить пункти для зміни формату відображення даних. Переміщуватися за списком можна тими ж способами, що й в попередньому пункті.

При виборі пункту меню ***Сервис → Опции отладчика...***, опції налагоджувача можна змінювати.

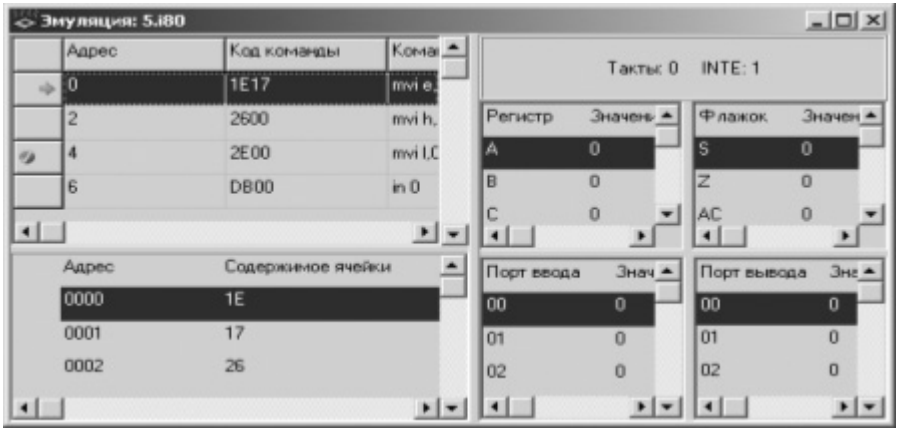


Рисунок Б.2 – Вікно Налагоджувача

Команди налагоджувача:

F4 – запуск програми до позиції курсору;

F5 – встановлення контрольної точки;

F7 – трасування за однією командою;

F8 – трасування за командою без входу до підпрограми;

F9 – запуск програми без перекомпіляції;

Ctrl+F2 – зупин програми (встановлення PC на початок програми).

Графічний інтерпретатор портів виводу описаний у Додатку Л.

Меню ЕМПС

Файл:

Новый - (елемент по умовчанню; виконується при подвійному натисканні напису “Файл”) - створює новий файл;

Открыть - відкриває файл для перегляду та редагування;

Сохранить - збереження поточного файлу;

Сохранить как... - збереження файлу під іншим ім’ям;

Закрыть - закриває активне вікно;

Закрыть всё - закриває усі вікна додатку, крім головного;

Предварительный просмотр - перегляд та друк змісту активного вікна;

Печать - безпосередній друк змісту активного вікна;
Выход - завершення роботи програми.

Правка:

Отменить - відміна останньої операції у текстовому редакторі;
Вырезать - знищення виділеного фрагменту до буфера обміну;
Копировать - копіювання фрагменту до буфера обміну;
Вставить - вставка з буфера обміну в позицію курсору;
Удалить - вилучання фрагменту;
Выделить всё - виділення всього тексту програми.

Поиск:

Найти... - пошук фрагменту тексту;
Заменить... - пошук фрагменту та його заміна.

Вид:

Панель инструментов - настройка панелі інструментів:
Стандартная - стандартний вид панелі інструментів;
Скрыть(Отобразить) - керування видимістю панелі;
Настроить... - настройка панелі інструментів;
Строка статуса - керування видимістю рядка статусу.

Выполнить:

Компилировать... - компіляція програми;
Проверить синтаксис... - перевірка синтаксису програми;
Выполнить - компіляція та запуск програми;
Пауза - пауза при виконанні програми;
Остановить - завершити виконання програми;
Шаг вне - покрокове виконання програми без заходу у процедури;
Шаг внутрь - покрокове виконання програми із заходом у процедури;
Выполнить до курсора - виконання програми до курсору;
Добавить точку останова - додавання в програму точки зупинки.

Вставка:

Заголовок - вставка до програми стандартного шаблону:

```
; Лабораторная работа №
; ()

org 0
.startup   .
```

Сервис:

Опции редактора... - зміна опцій редактора;
Опции отладчика... - зміна опцій налагоджувача.

Окно:

Каскад - розташування вікон каскадом;
Упорядочить - упорядкування вікон:
По горизонтали.
По вертикали.
Упорядочить значки - упорядкування значків згорнутих вікон;
Свернуть всё - згорнути усі вікна, крім головного;
Восстановить всё - відновлення раніше згорнутих вікон.

Помощь:

Справка - запуск довідки програми "Эмулятор CPU I8080", командам 8-ми розрядного Асемблера та директивам компілятора;
О программе... - інформація про версію та авторів програми.

Додаток В – Система команд мікропроцесора I8080

Система команд однокриштального мікропроцесора I8080 має команди трьох форматів. Перший байт команди містить інформацію про формат команди, код операції, вид адресації та про регістри або регістрові пари, якщо вони приймають участь у виконанні операцій.

У двобайтових командах другий байт (***B2***) містить 8-розрядний операнд або 8-розрядну адресу пристрою вводу чи виводу.

У трибайтових командах другий та третій байти (***B2***, ***B3***) містять 16-розрядні адреси (у командах із прямою адресацією пам'яті) або 16-розрядні операнди (у командах завантаження регістрових пар або покажчика стека). Другий байт трибайтової команди містить молодший байт числа, третій – старший.

У таблиці В.1 наведені мнемонічні позначення та опис команд мікропроцесора i8080. При цьому використовуються наступні умовні позначення :

r - регістр загального призначення;
rp - пара регістрів загального призначення;
PC - програмний лічильник (лічильник команд);
SP - покажчик стека;
M - чарунка пам'яті;
B2,B3 - другий та третій байти команд

Таблиця В.1 - Система команд мікропроцесора I8080

Команда	Довжина команди, байт	Число тактів	Опис команди	Ознаки
1	2	3	4	5
1 Команди пересилання даних				
MOV r1,r2	1	5	Пересилання даних із регістра **r**2 до регістра **r**1	-
MOV M,r (MOV r,M)	1	7	Пересилання даних із регістра **r** до пам'яті за адресою, що зберігається у регістровій парі **H-L** (із пам'яті до регістру **r**)	-
XCHG	1	4	Обмін даними між парами регістрів **H-L** і **D-E**	-

Продовження таблиці В.1

1	2	3	4	5
MVI r,<B2> (MVI M,<B2>)	2	7 (10)	Занесення байта даних до регістру **r** (до пам'яті)	-
LXI rp <2байта>	3	10	Занесення двох байтів даних у пару регістрів **(B-C, D-E, H-L, SP**). Третій байт команди заноситься в старший регістр, а другий - у молодший	-
LDAX rp	1	7	Завантаження в накопичувач вмісту чарунки, яка опосередковано адресується парою регістрів **rp (B-C, D-E)**	-
LDA <адреса>	3	13	Завантаження накопичувача вмістом чарунки за вказаною адресою. 2-й байт команди - молодший байт адреси, 3-й байт - старший	-
STAX rp	1	7	Занесення вмісту накопичувача до чарунки, яка опосередковано адресується парою **rp**	-
STA, <адреса>	3	13	Занесення вмісту накопичувача до чарунки за вказаною адресою	-
LHLD <адреса>	3	16	Завантаження регістра **L** вмістом чарунки за вказаною адресою, а регістр **H** - чарунки з адресою на одиницю більше	-
SHLD <адреса>	3	16	Занесення вмісту регістрів **H** і **L** до пам'яті (аналогічно команді **LHLD**)	-
2 Арифметичні команди				
ADD r (ADD M)	1	4 (7)	Складання змісту регістра **r** (чарунки пам'яті) і накопичувача	Z,S,P,C
ADC r (ADC M)	1	4 (7)	Складання змісту регістра **r** (чарунки пам'яті) і накопичувача з бітом перенесення	Z,S,P, C,AC

Продовження таблиці В.1

1	2	3	4	5
SUB r (SUB M)	1	4 (7)	Віднімання змісту регістра **r** (чарунки пам'яті) від змісту накопичувача	Z,S,P, C^1,AC^2
SBB r (SBB M)	1	4 (7)	Віднімання змісту регістра **r** (чарунки пам'яті) та біта перенесення від змісту накопичувача	Z,S,P, C^1,AC^2
ADI ,<байт>	2	7	Складання байта зі змістом накопичувача	Z,S,P,C, AC
ACI ,<байт>	2	7	Складання байта зі змістом накопичувача та бітом перенесення	Z,S,P,C, AC
SUI,<байт>	2	7	Віднімання байта із змісту накопичувача	Z,S,P, C^1,AC^2
SBI,<байт>	2	7	Віднімання байта команди і біта перенесення від змісту накопичувача	Z,S,P, C^1,AC^2
DAD rp	1	10	Складання змісту пари регістрів **rp** (**B-C,D-E,H-L,SP**) зі змістом пари регістрів **H-L**	C
INR r (INR M)	1	5 (10)	Збільшення змісту регістра **r** (чарунки пам'яті) на одиницю	Z,S,P, AC
DCR r (DCR M)	1	5 (10)	Зменішення змісту регістра **r** (чарунки пам'яті) на одиницю	Z,S,P, AC^2
INX rp (DCX rp)	1	5	Збільшення (зменьшення) змісту пари регістрів **rp** (**B-C,D-E,H-L,SP**) на одиницю	-
DAA	1	4	Перетворення змісту накопичувача в двійково-десятичний код	Z,S,P,C, AC
3 Логічні команди				
ANA r (ANA M)	1	4 (7)	Порозрядне 'І' над змістом регістра **r** (чарунки пам'яті) і накопичувача	Z,S,P, C=0, AC=0
XRA r (XRA M)	1	4 (7)	Порозрядне виключаюче 'АБО' над змістом регістра **r** (чарунки пам'яті) і накопичувача	Z,S,P, C=0, AC=0

Продовження таблиці В.1

1	2	3	4	5
ORA r (ORA M)	1	4 (7)	Порозрядне ‘АБО’ над змістом регістра **r** (чарунки пам’яті) і накопичувача	Z,S,P, C=0, AC=0
CMP r (CMP M)	1	4 (7)	Порівняння змісту регістра **r (**чарунки пам’яті) і накопичувача	(Z,S,P, C,AC)[3]
ANI,<байт>	2	7	Порозрядне ‘І’ над змістом накопичувача і байтом	Z,S,P, C=0, AC=0
XRI,<байт>	2	7	Порозрядне виключаюче ‘АБО’ над змістом накопичувача і байтом	Z,S,P, C=0, AC=0
ORI,<байт>	2	7	Порозрядне ‘АБО’ над змістом накопичувача і байтом	Z,S,P, C=0, AC=0
CPI,<байт>	2	7	Порівняння байта зі змістом накопичувача	(Z,S,P, C,AC)[3]
RLC (RRC)	1	4	Циклічний зсув змісту накопичувача вліво (вправо)	C[4]
RAL (RAR)	1	4	Циклічний зсув змісту накопичувача вліво (вправо) через перенесення	C[4]
CMA	1	4	Порозрядне інвертування накопичувача	-
STC	1	4	Встановлення ознаки перенесення в одиницю	C=1
CMC	1	4	Інвертування ознаки перенесення С	$C=\overline{C}$
4 Команди переходів				
PCHL	1	5	Занесення змісту регістрів **H**, **L** в лічильник команд (зміст **H** – в старший байт, **L** – в молодший)	-
JMP,<адреса>	3	10	Безумовний перехід по вказаній адресі	-
JC/(JNC), <адреса>	3	10	Перехід при наявності (відсутності) перенесення	-
JZ/(JNZ), <адреса>	3	10	Перехід при наявності (відсутності) нуля	-
JP/(JM), <адреса>	3	10	Перехід при плюсі (мінусі)	-

Продовження таблиці В.1

1	2	3	4	5
JPE/(JPO), <адреса>	3	10	Перехід при парності (непарності)	-
CALL,<адреса>	3	17	Виклик підпрограми	-
CC/(CNC), <адреса>	3	11 (17)	Виклик підпрограми при наявності (відсутності) перенесення	-
CZ/(CNZ), <адреса>	3	11 (17)	Виклик підпрограми при наявності (відсутності) нуля	-
CP/(CM), <адреса>	3	11 (17)	Виклик підпрограми при плюсі (мінусі)	-
CPE/(CPO), <адреса>	3	11 (17)	Виклик підпрограми при парності (непарності)	-
RET	1	10	Повернення з підпрограми	-
RC/(RNC)	1	5 (11)	Повернення при наявності (відсутності) перенесення	-
RZ/(RNZ)	1	5 (11)	Повернення при наявності (відсутності) нуля	-
RP/(RM)	1	5 (11)	Повернення при плюсі (мінусі)	-
RPE/(RPO)	1	5 (11)	Повернення при парності (непарності)	-
RST,<номер>	1	11	Повторне запущення з адреси 8 х номер (0,8,…,56)	-
5 Команди вводу-виводу та управління				
IN,<порт>	2	10	Ввод даних із вказаного порту до накопичувача	-
OUT,<порт>	2	10	Вивід даних із накопичувача до вказаного порту	-
PUSH rp	1	11	Занесення змісту пари регістрів **rp** (**B-C,D-E,H-L, PSW**) до стека	-
POP rp	1	10	Видання даних зі стека в пару регістрів **rp** (**B-C,D-E, H-L,PSW**)	(Z,S,P,CAC)[6]
XTHL	1	18	Обмін даними між верхівкою стека та парою регістрів **H-L**	-
SPHL	1	5	Занесення в покажчик стека змісту регістрів **H-L**	-
DI/EI	1	5	Заборонити/дозволити переривання	-
NOP	1	4	Порожня операція	-
HLT	1	7	Зупиню	-

Примітки:

1 встановлюється при наявності займу до старшого розряду, у протилежному випадку скидається;

2 встановлюється при наявності займу зі старших чотирьох розрядів в молодші, у протилежному випадку скидається;

3

- **Z** встановлюється, якщо зміст регістра та байта даних дорівнює змісту накопичувача ;
- **S,C,** якщо зміст регістра або байта даних більше змісту накопичувача;
- **AC,** якщо зміст молодших чотирьох розрядів регістра й байта даних більше змісту молодших чотирьох розрядів накопичувача;
- **P,** якщо байт різниці між змістом накопичувала та змістом регістра або байта даних містить парне число одиниць;

4 стан ознаки дорівнює значенню висунутого з накопичувача двійкового розряду;

5 у знаменнику дробу вказано кількість тактів при виконанні умов, в чисельнику – при невиконанні;

6 за командою ***POP PSW*** ознаки установлюються відповідно до значення розрядів слова, яке занесено до стека, при інших значеннях гр ознаки не змінюються

Додаток Д - Призначення та склад регістра ознак

Регістр ознак використовується для зберігання ознак результатів операцій. В МП І8080 з цією метою використовуються тригери умов (прапорці), які для зручності об'єднано у спеціальний регістр. При читанні ознак на шині даних у байті прапорці розміщуються таким чином:

D7	D6	D5	D4	D3	D2	D1	D0
S	Z	0	AC	0	P	1	C

де **S** - ознака знаку: якщо результат < 0, то S=1;
Z- ознака нуля: якщо результат = 0, то Z=1;
AC - ознака допоміжного перенесення: AC=1, якщо є пере-несення із розряду D3 до розряду D4;
P- ознака парності: якщо у двійковому коді результату кількість одиниць є парне число, то P=1;
C - ознака перенесення зі старшого розряду: у випадку перенесення C=1.

Кожен із перелічених п'яти розрядів є ознакою (прапорцем), а інші розряди мають фіксовані значення.

За допомогою регістра ознак реалізуються розгалужені алгоритми.

Додаток Ж - Способи організації часових затримок

При організації часової затримки використовують дані табл.В.1 про час виконання тієї чи іншої команди. В залежності від потрібної тривалості затримки доцільно розділити всі способи організації затримки на дві групи:

- способи організації затримки малої тривалості;
- способи організації затримки великої тривалості;

Для обох груп затримка кратна тривалості одного машинного такту $t_{зад}$=0,5 мкс.

Для організації затримок малої тривалості використовуються команди, які не змінюють інформацію та стан ЦПЕ. Рекомендується використовувати команди NOP; MOV A,A; ADI 0 і пару команд PUSH-POP, які складають у ланцюжок.

Для організації затримок великої тривалості використовують однокаскадний або багатокаскадний лічильник. Типова схема однокаскадного лічильника показана на рисунку Ж.1.

Час затримки в машинних тактах розраховується за формулою:

$$N_{зад}=T_{вст}+ N_в(T_{рах}+T_{дод.\ в})+T_{дод.\ к} \qquad (В.1)$$

де $N_в$ - кількість циклів лічильника;

$T_{вст}$ - кількість тактів виконання команди встановлення початкового стану лічильника;

$T_{рах}$ - мінімальна затримка у циклі рахування;

$T_{дод.\ в}$ - додаткова затримка у циклі рахування;

$T_{дод.\ к}$ - додаткова затримка після закінчення циклу;

Вибір величин, що входять до цього виразу, здійснюється методом підбору за даними (кількість тактів) табл.В.1.

Текст програми, що реалізує схему однокаскадного лічильника, наведено у таблиці Ж.1.

Таблиця Ж.1 - Текст програми однокаскадного лічильника

Адреса ОЗП	Код команди	Мітка	Мнемоніка	Кільк. тактів	Формат	Текст коментарю
04A0 04A1	06 C8H		**MVI B,N_B**	7	2	Встановлення пачатк. стану
04A2	00	**M1:**	**NOP**	4	1	Додаткова затримка у циклі
04A3	05		**DCR B**	5	1	Мінімальна затримка лічильника
04A4 04A5 04A6	C2 A2 04		**JNZ M1**	10	3	
04A7	00		**NOP**	4	1	Додаткова затримка після закінчення рахування
04A8	7F		**MOV A,A**	5	1	

При $N_в$=200 дана програма реалізує затримку в машинних тактах, яка дорівнює

$$N_{зад}=7+200*(15+4)+9=3816$$

Звідси час затримки:

$$T_{зад}=t_{зад}*N_{зад}=0,5*3816=1908мкс.$$

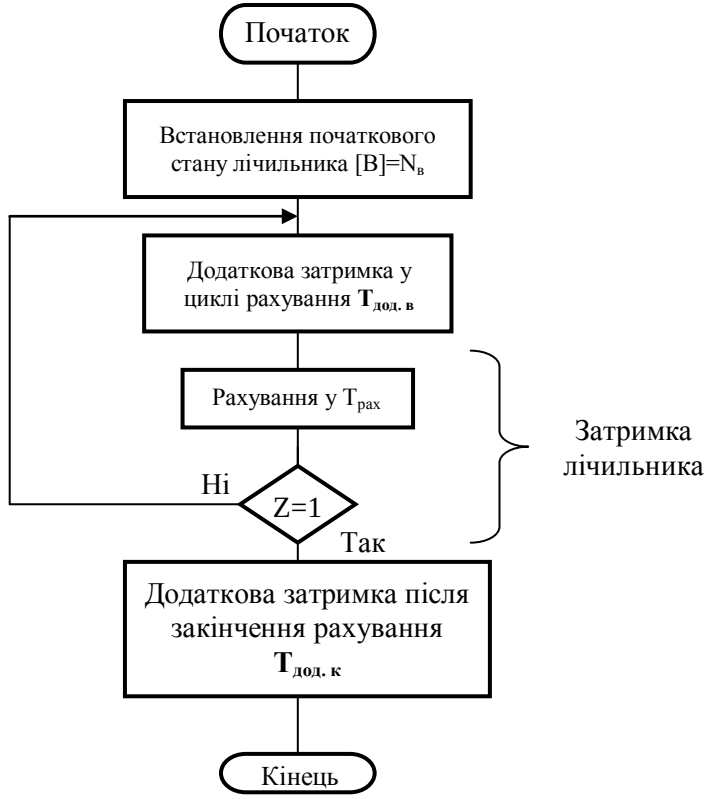


Рисунок Ж.1 - Типова схема однокаскадного лічильника.

Використовуючи багатокаскадні лічильники, можливо отримати практично будь-яку затримку.

Додаток К - Приклад реалізації логічної функції

Нехай задана принципова схема логічного функціонального вузла (рисунок К.1). Програмна реалізація даного пристрою складається з наступних етапів.

К.1 Складання та перетворення логічної функції.

$$Y=\overline{X1*X2}+X3*X4=X1*X2*\overline{X3*X4}=X1*X2*(\overline{X3}+\overline{X4})=$$

$$=X1*X2*\overline{X3}+X1*X2*\overline{X4}=Y1+Y2$$

К.2 Складання таблиці істинності отриманої функції.

Таблиця К.1 - Таблиця істинності логічного функціонального вузла

X1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
X2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
X3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
X4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Y1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
Y2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

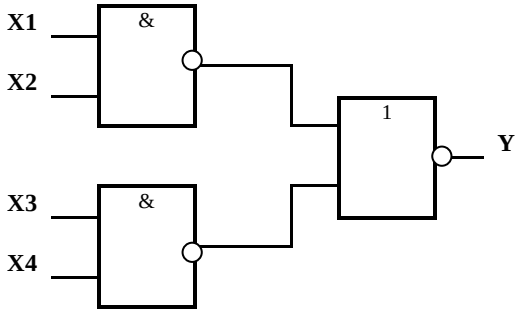


Рисунок К.1 - Схема функціонального вузла

К.3 Складання схеми алгоритму, що реалізує логічну функцію.

Схема повинна складатися з ланцюжка блоків, в яких відбувається обчислення значення відповідної частини функції з наступною перевіркою на рівність результату одиниці (рисунок К.2).

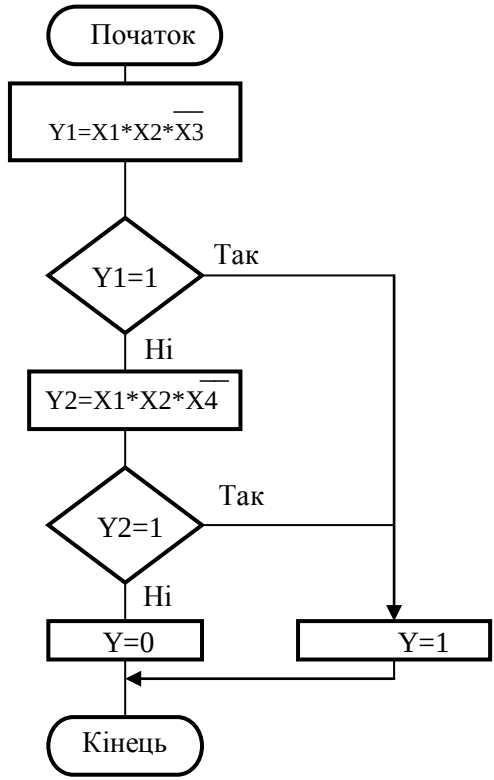


Рисунок К.2 - Схема програми, що реалізує логічний пристрій

К.4 Складання тексту програми.

Цей етап для кожного частини функції здійснюється у такій послідовності.

К.4.1 Маскування розрядів, які не використовуються, шляхом логічного множення на “маску” – двійкове число, в якому на місці

розрядів, які не використовуються, стоять нулі, а на інших місцях – одиниці.

Таблиця К.2 - Вихідна інформація

	X7	X6	X5	X4	X3	X2	X1	X0
Маска Y1	0	0	0	0	1	1	1	0
Маска Y2	0	0	0	1	0	1	1	0

К.4.2 Ознакою умовного переходу до аналізу наступної частки обираємо прапорець Z, для цього замінимо перевірку Y1=1 і Y2=1 перевіркою Y1=0 і Y2=0.

Проведемо складання по модулю 2 "маскованої" вхідної інформації з константою, отриманою зі своєї "маски" шляхом заміни одиниць на нулі у розрядах, що містять $\overline{X_i}$.

К.4.3 Здійснюється перехід по прапорцю Z.

Текст програми, що реалізує отриману логічну функцію, наведений у табл.Д.3.

Таблиця К.3- Текст програми

Мітка	Мнемоніка	Кільк. тактів	Формат	Текст коментарю
FUNC:	**IN PORT X**	10	2	Ввід {X} у A
	MOV C,A	5	1	Занесення X до C
	ANI MASK1	7	2	Обчислення Y1
	XRI CONST1	7	2	
	JZ M1	10	3	Перехід доY=1, тому що Y1=1
	MOV A,C	5	1	Відновлення X у A
	ANI MASK2	7	2	Обчислення Y2
	XRI CONST2	7	2	
	JZ M1	10	3	Перехід доY=1, тому що Y2=1
	XRA A	4	1	Формування Y=0
	JMP M2	10	3	
M1:	**MVI A,FF**	7	2	A=11111111
M2:	**OUT PORT Y**	10	2	Вивід Y
	RET	10	1	Повернення

Додаток Л Графічний інтерпретатор портів виводу

Графічний інтерпретатор портів виводу (ГІПВ) призначений для складання програм імітації управління технологічними процесами. ГІПВ має наступний вигляд:

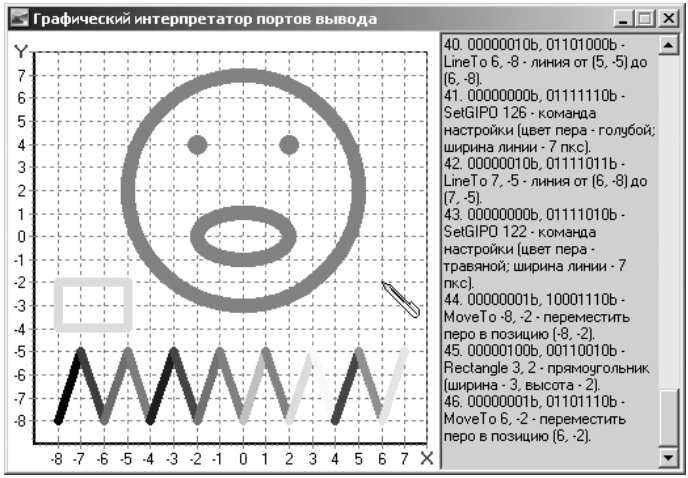


Рисунок Л.1 – Вікно ГІПВ

Для того, щоб при виконанні програми відображувався ГІПВ, необхідно в опціях налагоджувача задати опцію „***Отображать графический интерпретатор портов вывода***”. По умовчанню ГІПВ не відображається.

У лівій частині вікна знаходиться графічна область для рисування всіляких графічних примітивів, у правій – список команд, що поступили на ГІПВ, з їх описом.

Робота з ГІПВ здійснюється через порт **05**. Значення, що поступають на цей порт, сприймаються (інтерпретуються) ГІПВ як деякі керуючі команди. Кожна така команда займає 2 байти, тому може бути передана на виконання тільки за 2 кроки: спочатку на порт 05h посилається перший байт команди, а потім другий байт. Перший байт містить код команди (дана версія ГІПВ підтримує 7 команд), а другий – її параметри.

Система команд ГІПВ наведена в таблиці Л.1.

Таблиця Л.1 - Система команд ГІПВ

Команда	Виклик	Коментар
1	2	3

SetGIPO W, C	`mvi a,00h`
	`out 05h`
	`mov a,B2`
	`out 05h`
		Команда настроювання ГІПВ. Молодший напівбайт числа ***B2*** задає колір пера, старший напівбайт – ширину лінії. Для завдання кольору є 16 констант (див. нижче), ширина лінії відповідно може бути від 1 пкс до 15 пкс. Якщо старший напівбайт числа ***B2*** буде дорівнювати нулю, то ширина лінії пера встановиться в 1 пкс. Наприклад, наступний фрагмент програми
		`mvi a,00h`
		`out 05h`
		`mvi a,01111110b`
		`out 05h`
		встановлює ширину лінії в 7 пкс (0111) та голубий колір пера (1110).

MoveTo X, Y	`mvi a,01h`
	`out 05h`
	`mov a,B2`
	`out 05h`
		Переміщення пера до зазначеної позиції. Молодший напівбайт числа ***B2*** задає координату по осі **X** (від -8 до 7), старший напівбайт – координату по осі **Y** (від -8 до 7).
		Наприклад,
		`mvi a,01h`
		`out 05h`
		`mvi a,01101110b`
		`out 05h`
		переміщує перо в точку X = 6 (0110), Y = -2 (1110).

Продовження таблиці Л.1

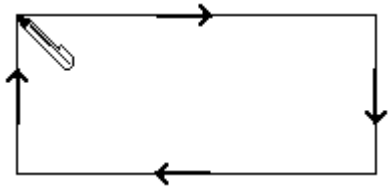
1	2	3

LineTo X, Y	`mvi a,02h`
	`out 05h`
	`mov a,B2`
	`out 05h`
		Рисування лінії з поточної позиції пера в точку, що вказана другим байтом команди. Як і для команди MoveTo, молодший напівбайт числа ***B2*** задає координату по осі **X**, старший напівбайт – координату по осі **Y**. Наприклад,
		
		`mvi a,02h`
		`out 05h`
		`mvi a,01111011b`
		`out 05h`
		
		рисує лінію від поточної точки в точку X = 7 (0111), Y = -5 (1011).

PutPixel	`mvi a,03h`
	`out 05h`
	`mov a,B2`
	`out 05h`
		Використовуючи поточні значення для ширини лінії та кольору пера, в поточній позиції ставиться точка. Значення другого байту команди ***B2*** не має значення. Але його слід посилати на порт 05, інакше команда не буде виконана. Наприклад,
		
		`mvi a,03h`
		`out 05h`
		`mvi a,0`
		`out 05h`
		
		ставить точку в поточну позицію пера.

Продовження таблиці Л.1

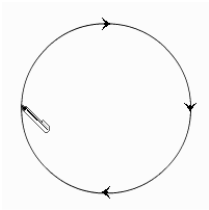
1	2	3

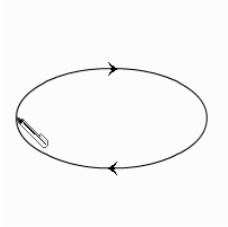
Rectangle W, H	`mvi a,04h`
	`out 05h`
	`mov a,B2`
	`out 05h`
		Рисування прямокутника з шириною, що міститься в старшому напівбайті числа ***B2*** (від 0 до 15), та з висотою, що міститься в молодшому напівбайті числа ***B2*** (від 0 до 15). Прямокутник рисується з поточної позиції пера в напрямку, вказаному на рисунку. Таким чином, після виконання команди, перо повертається в початкове положення.
		 
		Наприклад,
		`mvi a,04h`
		`out 05h`
		`mvi a,00110010b`
		`out 05h`
		рисує прямокутник з шириною - 3 та висотою - 2.

Circle R	`mvi a,05h`
	`out 05h`
	`mov a,B2`
	`out 05h`
		Рисування кола з радіусом, що міститься у молодшому напівбайті числа ***B2*** (від 0 до 15). Значення старшого напівбайта числа ***B2*** не має значення. Коло рисується з поточної позиції пера у напрямку, що вказаний на рисунку. Тому, як у випадку виконання команди **Rectangle**, перо повертається у початкове положення.

Продовження таблиці Л.1

1	2	3

		
		Наприклад, програма
		`mvi a,05h`
		`out 05h`
		`mvi a,00000101b`
		`out 05h`
		рисує коло з радіусом - 5.

Ellipse a, b	`mvi a,`**`06h`**
	`out 05h`
	`mov a,`*`B2`*
	`out 05h`
		Рисування еліпса з радіусами, що містяться у напівбайтах числа ***B2*** (від 0 до 15). Еліпс рисується з поточної позиції пера у напрямку, що вказаний на рисунку. Тому, як і у випадку виконання команд ***Rectangle*** та ***Circle,*** перо повертається у початкове положення.
		 
		Наприклад, програма
		`mvi a,06h`
		`out 05h`
		`mvi a,00110101b`
		`out 05h`
		рисує еліпс з радіусами - 3 та 5.

Колір задається константами, що наведені у таблиці Л.12.

Таблиця Л.2 - Список констант для завдання кольорів

Константа	Колір	Константа	Колір
0	чорний	8	сріблистий
1	малиновий	9	червоний
2	зелений	10	трав'яний
3	маслиновий	11	жовтий
4	темно-синій	12	синій
5	бузковий	13	рожевий
6	бірюзовий	14	блакитний
7	сірий	15	білий

Розглянемо фрагмент програми, результат роботи якої можна побачити на рисунку Л.2.

```
org 0
circle:
mvi a,05h
out 05h
mvi a,1b
out 05h
ret

org 10
.startup
mvi a,00h
out 05h
mvi a,11000000b
out 05h
mvi a,1h
out 5h
mvi a,10000111b
out 5h
mvi a,02h
out 05h
mvi a,10010110b
out 05h
mvi a,01h

out 05h
mvi a,10010101b
out 05h
call circle
mvi a,1h
out 05h
mvi a,10110111b
out 05h
call circle
mvi a,1h
out 05h
mvi a,11010101b
out 05h
call circle
mvi a,1h
out 05h
```

```
mvi a,11110110b
out 05h
mvi a,2h
out 05h
mvi a,00000111b
out 05h
mvi a,2h
out 05h
mvi a,01110111b
out 05h
mvi a,01h
out 05h
mvi a,11000110b
out 05h
mvi a,2h
out 05h
mvi a,11000001b
out 05h
mvi a,01h
out 05h
mvi a,10110000b
out 05h
call circle
mvi a,01h
out 05h
mvi a,11110000b
out 05h
call circle
mvi a,01h
out 05h
mvi a,00110000b
out 05h
call circle
mvi a,01h
out 05h
mvi a,10111011b
out 05h
call circle
mvi a,01h
out 05h
mvi a,11111011b
out 05h
call circle
```

```
mvi a,01h
out 05h
mvi a,00111011b
out 05h
call circle
mvi a,01h
out 05h
mvi a,00000001b
out 05h
mvi a,02h
out 05h
mvi a,00000100b
out 05h
mvi a,02h
out 05h
mvi a,00010101b
out 05h
mvi a,02h
out 05h
mvi a,01110101b
out 05h
mvi a,01h
out 05h
mvi a,01000001b
out 05h
mvi a,02h
out 05h
mvi a,01000010b
out 05h
mvi a,02h
out 05h
mvi a,01010011b
out 05h
mvi a,02h
out 05h
mvi a,01110011b
out 05h
mvi a,01h
out 05h
mvi a,00111100b
out 05h
mvi a,02h
out 05h
```

```
mvi a,00101101b
out 05h
mvi a,02h
out 05h
mvi a,10001101b
out 05h
mvi a,01h
out 05h
```

```
mvi a,11001010b
out 05h
mvi a,02h
out 05h
mvi a,11001000b
out 05h

hlt
```

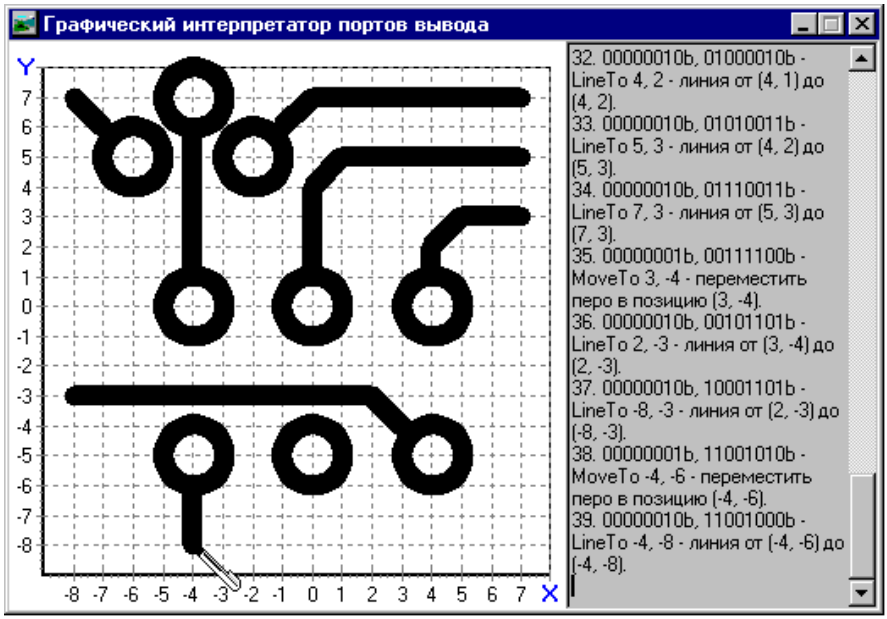


Рисунок Л.2 – Результат роботи програми

Спливаюче меню інтерпретатора містить один пункт – "Очистить графический интерпретатор", що дозволяє очистити графічну область.

Додаток М Варіанти завдань для самостійної роботи

На мові Асемблера мікропроцесора I8080 скласти програми, які реалізують наступні завдання :

1. У масиві чисел, розташованому з адреси 0150 Н до 0200 Н, встановити в "1" старші і в" 0" молодші розряди чисел.
2. Підрахувати, скільки разів у масиві чисел з адресами 0200-02FF зустрічаються числа, менші, ніж перший елемент масиву.
3. У масиві чисел, які зберігаються в ОЗП у чарунках з адресами з 0500 до 05DE, замінити від'ємні числа нулем.
4. Підрахувати число додатних елементів у масиві з адресами 0600-06AE.
5. Здійснити обмін інформації між масивами, розташованими з адрес 0400 і 0500, які вміщають по 21 елементу.
6. Програма формування тимчасової затримки тривалістю 0,3 с (f = 1,2 МГц).
7. Зсунути вміст чарунки пам'яті з адресою 040F на число бітів, яке визначається вмістом регістру **D**.
8. Скласти два числа, які знаходяться у чарунках з адресами 0100Н та 0200Н, і вивести молодший напівбайт у порт 01, а старший напівбайт - порт 02.
9. Масив чисел у чарунках ОЗП з адресами з 0500 до 05CD переписати у ті ж самі чарунки у зворотному порядку.
10. З масиву чисел із адресами з 0400 до 04CD видавати вміст чарунок із непарними адресами у вихідний порт, помножуючи їх на 2.
11. Переписати 10 послідових байтів із одної області пам'яті (починаючи з адреси 2200 Н) у другу (з адреси 2300 Н).
12. Програма пересилки у старший розряд вихідного порту логічної одиниці з частотою 100 Гц (f = 2 МГц).
13. Масив чисел розмірністю 120, розташований, починаючи з адреси 0400, вивести на вихідні порти 01 (молодший напівбайт) і 02 (старший напівбайт).
14. Замінити у масиві з адресами 0500Н-05ССН числа, більші 9, числом 10.
15. В масиві чисел із адресами з 0500Н до 05FFH числа, які зберігаються у чарунках із парними адресами, замінити на 0.