МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ"

МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІНИ: "ОСНОВИ МІКРОПРОЦЕСОРНОЇ ТЕХНІКИ"

ДЛЯ СТУДЕНТІВ НАПРЯМКУ ПІДГОТОВКИ: 6.0924 – "ТЕЛЕКОМУНІКАЦІЇ" ТА 6.0912 – "АКУСТОТЕХНІКА"

КИЇВ, АВЕРС
2008

УДК 621.3.011 (075.8)
ББК 31.211я73 Г94

Укладачі: В. Б. Швайченко, доц.; К. А.Трапезон, ас.

Рецензенти: Дегтярьов В.В., доц., к.т.н.; Медведенко Б.И., проф., к.т.н.

МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ РОБІТ
З ДИСЦИПЛІНИ "ОСНОВИ МІКРОПРОЦЕСОРНОЇ ТЕХНІКИ"
/Склад. В. Б. Швайченко, К. А. Трапезон. - К.: Аверс, 2006, - 39 с.

Підписано до друку 20.02.2006. Формат 60x84/16.

РОЗПм.- друк. арк. 4,2. Тираж 200 прим. Зам. № 5-06

Видавництво ПЦ "АВЕРС", 03056, Київ, вул. Політехнічна, 16.

Свідоцтво про реєстрацію № 5933 від 24.07.1998 р.

ЗМІСТ

Вступ

Даний курс лабораторних робіт призначений для отримання початкових практичних навичок роботи з мікроконтролерами серії 68HC11 фірми Motorola.

В якості лабораторного стенду використовується налагоджувальний модуль HC11EVB, який працює на базі мікроконтролера MC68HC11E9. Цей налагоджувальний модуль дозволяє проводити налагодження програмного й апаратного забезпечення мікроконтролерів на базі BIC MC68HC11 серій AO, A1, A8, EO, E1, E2, E9, DO, D3. На лабораторному стенді також можна вивчати структури системи команд, схемотехнічне та програмне забезпечення, порядок функціонування й методи програмування МК BIC серії MC68HC11.

Загальні вимоги, щодо оформлення лабораторних робіт з курсу "Основи мікропроцесорної техніки"

(для студентів денної форми навчання)

Протокол лабораторної роботи складається з наступного:

1) Титульний аркуш протоколу до лабораторної роботи повинен містити наступну інформацію:

   - назва вузу, кафедри де проводиться лабораторна робота;
   - прізвище ім'я та по-батькові студента, що виконав лабораторну роботу;
   - прізвище керівника лабораторних занять.

2) Мета роботи

3) Відповіді на контрольні запитання

4) Схеми, таблиці, діаграми, текст програм, необхідні рисунки

5) Висновки за результатами виконання лабораторної роботи.

Лабораторна робота №1

Тема роботи – Вивчення архітектури мікроконтролера MC68HC11E9 та його основних параметрів, ознайомлення з побудовою розподілу пам'яті пристрою, вивчення особливостей регістрів загального та спеціального призначень.

Мета роботи – Ознайомитися з принципами побудови та роботи мікроконтролерів серії 68HC11 фірми Motorola в різних режимах. Вивчити основні параметри мікроконтролера MC68HC11E9 та призначення основних виводів пристрою.

Домашнє завдання

1. *Ознайомитися з описом мікроконтролера, його типовою структурою, принципами підключення зовнішньої пам'яті та різноманітних периферійних пристроїв.*

2. *Вивчити двійкову, десяткову та 16-кову системи числення.*

Теоретичні відомості

Мікроконтролер – це обчислювальна контрольно-вимірювальна або керуюча системи, основним пристроєм в яких є арифметичний пристрій (процесор). До складу даного мікроконтролера входять: ПЗП (постійний запам'ятовуючий пристрій); ОЗП (оперативний запам'ятовуючий пристрій); восьмиканальний 8-розрядний АЦП (аналого-цифровий перетворювач); асинхронний послідовний інтерфейс зв'язку; синхронний послідовний периферійний інтерфейс; основний 16-розрядний таймер із трьома вхідними й п'ятьма вихідними лініями, що підтримує переривання реального часу; 8-розрядний лічильник зовнішніх імпульсів для підрахунку зовнішніх імпульсів або виміру періодів зовнішніх сигналів.

Крім цього, до складу мікроконтролера входить схема автоматичного спостереження, яка призначена для захисту системи від помилок. Ця схема генерує системне скидання у випадку зупинки роботи або неприпустимо низької частоти тактового генератора.

Мікро контролер (МК) MC68HC11E9 належить до серії 8-розрядних мікроконтролерів серії 68HC11 з низьким енергоспоживанням. Ці МК мають внутрішню пам'ять та, до того ж, в них є можливість під'єднання зовнішньої пам'яті. Усі моделі даної серії містять 16-розрядний таймер, паралельні та послідовні порти переміщення даних.

MC68HC11E9 характеризується наступними основними параметрами:

- 8-розрядний процесор з арифметично-логічним пристроєм обробки даних;

- 12 кБ інтегрованої ПЗП (ROM);
- 512 байт інтегрованої ОЗП (RAM);
- 512 байт інтегрованої EEPROM;
- 16-розрядний таймерний блок;
- 38 виводів вводу/виводу даних загального призначення;
- 2 послідовних порти SCI та SPI;
- 8-розрядний АЦП;
- Живлення: 3-5 В;
- Тактова робоча частота: 2-3 МГц;

На рисунку 1 наведена структурна схема мікроконтролера.

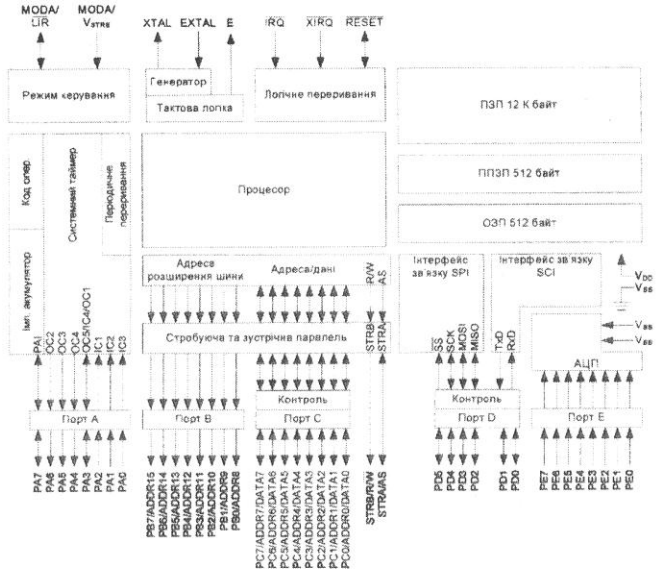

Рисунок 1 – Структурна схема мікроконтролера MC68HC11E9

Центральний процесор МК містить наступні регістри:

1) два 8-розрядних регістра – акумулятори A та B, які при обробці деяких команд можуть бути використані як 16-розрядний регістр D. Ці регістри

зазвичай використовуються для зберігання операндів чи результатів арифметичних розрахунків при роботі з даними;

2) два 16-розрядні індексні регістри IX та IY які насамперед мають бути задіяні при використанні індексного режиму адресації:
   - 16-розрядний регістр X використовується для індексного режиму адресації. Він містить 16-розрядні дані, з метою розрахунку ефективної адреси. Регістр X можна також використовувати як лічильник або регістр для тимчасового зберігання даних.
   - 16-розрядний регістр Y також можна використовувати для індексного режиму адресації, подібно до регістра X. Однак, всі команди, що використовують регістр Y, містять один додатковий байт коду, і час виконання збільшується на один додатковий цикл;

3) регістр ознак CCR. Цей регістр являє собою 8-розрядний регістр спеціального призначення, розряди якого відображають стан системи арифметично-логічного пристрою після виконання поточної команди. Ці розряди мають назви "ознак" (прапорців). Наведемо щодо них більш детальне пояснення:
   - Прапорець переносу/займу (C). Цей прапорець встановлюється, якщо протягом останньої арифметичної операції в арифметико-логічному пристрої (АЛП) було зафіксовано перенос. На стан цього прапорця також впливають команди зміщення й циклічного зміщення.
   - Прапорець переповнення (V). Прапорець переповнення встановлюється, якщо в результаті попередньої арифметичної операції було зафіксоване переповнення, у противному випадку прапорець скидається.
   - Прапорець нуля (Z). Прапорець нуля встановлюється в тому випадку, якщо результатом останньої арифметичної, логічної команди або команди пересилання даних був нуль, у противному випадку прапорець нуля скидається.
   - Прапорець негативного результату (N). Прапорець встановлюється, якщо результатом попередньої операції було негативне число, у противному випадку прапорець скидається. Число вважається негативним, якщо його старший значущий біт дорівнює одиниці.
   - Маска переривань (I). Біт заборони переривань може бути встановлено як апаратно, так і за допомогою команди в ході виконання програми. Встановлений біт забороняє усі переривання, що маскуються (як зовнішні, так і внутрішні).
   - Прапорець напівпереносу (H). Прапорець напівпереносу встановлюється в тому випадку, коли в ході виконання команд ADD, ABA або ADC в АЛП було зафіксовано перенос між третім і четвертим бітами, у всіх інших випадках прапорець скидається (див. детальне пояснення далі).

   - Маска переривань (X). Цей біт може бути встановлений тільки апаратно (за допомогою сигналів на виводах RESET/ або XIRO/). Скинути його можна програмно за допомогою - командою TAP або RTI.
   - Заборона режиму STOP (S). Біт забороняє виконання команди STOP, коли він встановлений, і дозволяє її виконання у зворотному випадку. Біт S повністю керується програмно. Якщо біт S встановлений, то команда STOP інтерпретується як NOP.

4) 16-розрядний покажчик стеку SP, що містить адресу наступного вільної ячейки стека. Стек представляє собою послідовність комірок пам'яті, організованих за принципом LIFO (останнім увійшов - першим вийшов), що дозволяє зберегти дані при перериваннях і викликах підпрограм. Щораз при додаванні в стек нового байта значення регістра SP зменшується, а при витягуванні зі стека -збільшується;

5) 16-розрядний програмний лічильник.

На рисунку 2 наведені регістри центрального процесора МК, які зазначені вище.

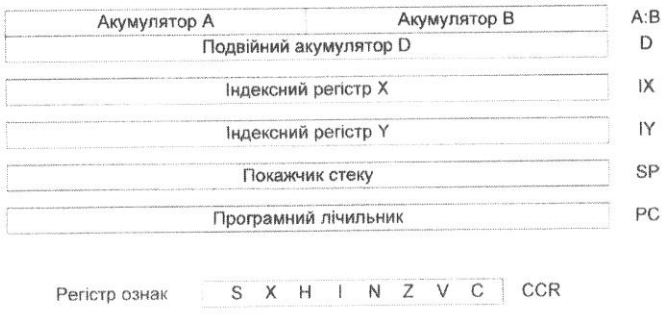

Рисунок 2 – Регістри МК

Карта пам'яті

Структура карти пам'яті характеризується наступним розподілом:
$0000 - $0047 – ОЗП користувача (54 байти);
$0048 - $0065 – адресний простір покажчика стеку монітора-програми Buffalo;
$0066 - $00C3 – змінні монітора-програми Buffalo;
$00C4 - $00FF – таблиця векторних переходів;
$6000 - $7FFF – простір для можливого додаткового ОЗП;

$B600 - $B7FF – 512 байт внутрішнього перепрограмуємого постійного запам'ятовуючого пристрою з електричним стиранням інформації (EEPROM).

Більш детальний розподіл наведено нижче:

Внутрішнє ОЗП	$0000 - $00FF
ОЗП користувача	$0100 - $01FF
PRU + Reg.DECODE	$1000 - $17FF
Не використовується	$1800 - $3FFF
FLIP-FLOP DECODE	$4000 - $5FFF
OPTIONAL 8K ОЗП	$6000 - $7FFF
Не використовується	$8000 - $97FF
TERMINAL ACIA	$9800 - $9FFF
Не використовується	$A000 - $B5FF
ППЗП	$B600 - $B7FF
Не використовується	$B800 - $BFFF

Процесор мікроконтролера виконує набір операцій над 8- або 16-розрядними операндами, розміщеними в регістрах і пам'яті. Команди мають довжину від 1 до 4 байтів: перший байт містить код операції, перед ним може йти префіксний байт, який вказує на звертання до одного з індексних регістрів X або Y (має значення $18, $1A або $CD), другий і третій байти забезпечують адресацію операнда. Два варіанти команд умовних розгалужень – за значенням біта BCLR, BSET (див. далі) з індексною адресацією, що використовують регістр X, містять п'ять байтів.

У більшості команд, що використовують пряму адресацію, може задаватися як 8-, так й 16-розрядна адреса. Однак у командах бітових операцій BCLR, BSET, BRCLR, BRSET використовуються тільки 8-розрядні адреси (коротка адресація в діапазоні $0000-SFFFF).

Для мнемокодів однакових команд, що виконуються за допомогою різних регістрів, вказівка регістра буде даватись у дужках, наприклад, LDA (A, B) - команди LDAA, LDAB завантаження вмісту регістрів A, B; PSH(X, Y) - команди завантаження в стек вмісту регістрів X, Y.

Команди пересилання данних здійснюють завантаження операндів з пам'яті в регістри, запис вмісту регістрів в пам'ять або можна організувати передачу даних між регістрами. При завантаженні даних (команди LD, LDA) використовуються усі способи адресації, крім відносної (див. далі), при записі в пам'ять (команди ST, STA) не використовується також безпосередня адресація. При завантаженні й запису в пам'ять вмісту регістра D (акумулятор подвійної розрядності), покажчика стека SP або індексних регістрів X, Y адресна комірка пам'яті містить старший байт (Dh, SPh, Xh або Yh), та осередок - молодший байт (Dl, SPl, Xl або Yl). Команди PSH виконують запис вмісту регістрів у стек, команди PUL - завантаження регістрів зі стеку. Команди очищення CLR роблять запис "0" у регістри A, B

або комірку пам'яті. Команди XGD(X,Y) виконують обмін даними між вмістом регістра D з індексними регістрами X або Y.

Комплекс налагодження

Комплекс програмно-апаратних засобів використовується з метою оцінки можливостей а також налагодження програм та схемотехніки мікроконтролерів серії 68HC11 фірми Motorola, що проектуються. Структурна схема модуля налагодження програм та схемотехнічних пристроїв на базі MC68HC11E9 наведена на рисунку 3.

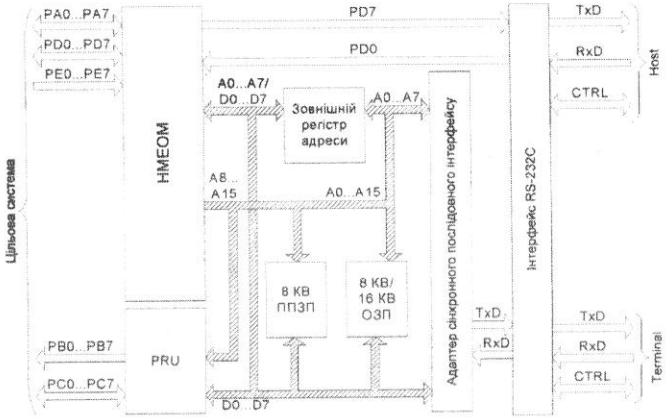

Рисунок 3 – Модуль налагодження програм

Контрольні питання

1. Що таке мікроконтролер?
2. Яким чином в програмах можуть бути представлені операнди?
3. Яку роль в мікроконтролері виконують регістри спеціального призначення?

4. Структурна схема 8-розрядного мікроконтролера. Призначення основних блоків.

5. Чим відрізняється індексний регістр IX від індексного регістра IY?

6. За якою технологією побудована стекова пам'ять в мікроконтролері?

7. Скільки в мікроконтролері паралельних портів, і які з них використовуються для вводу даних, а які для виводу? Яку особливість має порт A?

8. Яким чином розподілена пам'ять в мікроконтролері?

Лабораторна робота №2

Тема роботи – Вивчення системи команд програми-монітора Buffalo, синтаксису окремих команд а також отримання навиків щодо їх практичного застосування.

Мета роботи – Вивчити основні команди програми-монітора Buffalo, порядку їх запровадження та виконання. Ознайомитись з принципами організації контролю працездатності налагоджувального модулю.

Домашнє завдання

Вивчити команди програми-монітора Buffalo.

Теоретичні відомості

Крім існуючих команд програми-монітора, що будуть наведені нижче, термінал програми дозволяє застосувати наступні комбінації клавіш (так звані "гарячі клавіші"):

а) **^A** – вихід з діалогового режиму чи асемблера;

б) **^H** – повернення каретки;

в) **^W** – очікування чи останов екрану;

г) **DEL**– скидання команди;

д) **Enter** – введення команди чи повтор попередньої команди;

Особливість. Командний рядок виконується лише після натиснення клавіші Enter.

Основні команди програми-монітора Buffalo

1) **ASM [<адреса>]**

де

<адреса> - початкова адреса операції асемблірування.

Тлумачення. Ця команда визначає початок роботи інтерактивного асемблер-редактора.

Якщо адреса не вказана, то інтерактивний асемблер-редактор починає свою роботу за замовчуванням з адреси $0000.

Особливість. Усі невірні коди операцій на екрані висвічуються як ILLOP (illegal operation).

2) **BF <адреса 1> <адреса 2> <дані>**

де

<адреса 1> - початкова адреса пам'яті операції заповнення;

<адреса 2> - кінцева адреса пам'яті операції заповнення;

<дані> - 16-кові дані, якими буде заповнена визначена вище область пам'яті.

Тлумачення. Ця команда дозволяє користувачу визначену ним дозволену область ОЗП заповнити однаковими 16-ковими даними.

3) **MOVE <адреса 1> <адреса 2> [<цільова адреса>]**

де

<адреса 1> - початкова адреса пам'яті;

<адреса 2> - кінцева адреса пам'яті;

<цільова адреса> - початкова адреса заповнення даних, що переміщуються.

Тлумачення. Команда дозволяє скопіювати/перемістити вміст області пам'яті в область, що починається з цільової адреси відповідно.

Особливість. Якщо цільову адресу не вказано, то дані з комірок будуть переміщуватися через 1 байт.

Особливість. Повідомлення щодо виконання команди на екран не виводиться, а лише з'являється символ ">".

4) **MD [<адреса 1> [<адреса 2>]]**

де

<адреса 1> - початкова адреса пам'яті;

<адреса 2> - кінцева адреса пам'яті;

Тлумачення. Команда дозволяє переглянути область пам'яті з <адреси 1> до <адреси 2>.

Особливість. Якщо <адреса 2> не вказана, то в цьому випадку буде виведено 9 рядків по 16 біт починаючи з <адреси 1>.

Особливість. Якщо взагалі не вказано адрес, то буде виведено 9 рядків по 16 біт, починаючи з тої адреси, до якої було виконано останнє звернення.

5) **MM [<адреса>]**

де

<адреса> - адреса, з якої буде виконуватися перегляд (модифікація) пам'яті.

Особливість. З цієї адреси далі можна переміщувати дані (див. п.3), заповнювати даними (див. п.2).

6) **RM [P,Y,X,A,B,S]**

Тлумачення. Команда використовується для зміни:

- програмного лічильника PC;
- індексних регістрів IX,IY;
- акумуляторів A,B;
- покажчика стеку SP.

7) **CALL [<адреса>]**

де

<адреса> - початкова адреса, з якої виконується підпрограма користувача.

Тлумачення. Команда виклику підпрограми.

Особливість. Якщо <адреса> не вказана, то виконання підпрограми починається з адреси, що знаходиться у програмному лічильнику.

8) **T [<n>]**

де

n – кількість команд, що виконуються (у 16-ковій системі від 00 до FF).

Тлумачення. Це команда трасування, яка дозволяє виконувати програму по-командно. Виконання починається з наявної адреси програмного лічильника.

9) **G [<адреса>]**

де

<адреса> - початкова адреса, з якої починається виконання програми користувача.

Тлумачення. Команда запуску програми на виконання.

10) LOAD <T>

Тлумачення. Команда завантажує об'єктні дані (об'єктні коди програм) у форматі S-запису крізь основний порт T термінала на плату.

11) BULK

Тлумачення. Команда проводить операцію очищення ППЗП.

12) BULKALL

Тлумачення. Виконується очищення не лише ППЗП а й регістра конфігурацій CONFIG.

13) HELP

Тлумачення. Команда виводить на екран усі команди програми-монітора Buffalo зі стислим їх описом.

Робоче завдання

1. Виконати процедуру виводу на екран інформації щодо команд програми-монітора Buffalo.
2. Заповнити область комірок пам'яті $0000…$0010 числовими значеннями #55.
3. Змінити вміст комірки пам'яті, яку вказує викладач заняття.
4. Сформувати текстовий файл з програмою на мові Асемблера за допомогою текстового редактора:

```
LDX #05
LDAA #02
LDAB #03
MUL
STAA 0010
NOP
```

5. Виконати трасування програми

6. Ввести цю ж програму за допомогою команди ASM з адреси $0100 та повторити п.4 цього завдання.
7. Встановити точку останова на команді NOP за допомогою лістінга.
8. Запустити програму на виконання з адреси $0100.
9. Переглянути вміст комірок пам'яті, починаючи з адреси $0000.

Результати виконання робочого завдання відобразити у протоколі.

Контрольні питання

1. Призначення та основні функціональні можливості програми-монітору Buffalo.
2. Який адресний простір займає у зовнішньому ПЗП програма-монітор Buffalo?
3. Яке повідомлення виводиться на екран у випадку виконано невірної операції?
4. Перерахуйте та відповідно опишіть основні команди програми-монітора Buffalo.
5. Яку особливість обробки командних рядків програми на мові Асемблера має програма-монітор Buffalo.

Лабораторна робота №3

Тема роботи – Ознайомлення з особливостями застосування системи команд передачі даних мікроконтролера MC68HC11E9, вивчення режимів адресації.

Мета роботи – Вивчити основні команди передачі даних, що входять до системи команд мікроконтролера, порядку їх запровадження та виконання. Розібратися з режимами адресації, які використовуються при побудові програм на мові Асемблера.

Домашнє завдання

1. *Ознайомитися оглядово із основними групами команд, що утворюють систему команд мікроконтролера MC68HC11.*
2. *Вивчити режими адресації мікроконтролера.*
3. *Вивчити основні команди передачі даних.*

Теоретичні відомості

Основні команди передачі даних

Режими адресації

Мікроконтролер підтримує шість режимів адресації: безпосередню, пряму, розширену, індексну (з кожним із двох 16-розрядних індексних регістрів і восьмирозрядним зсувом), неявну та відносну. Для того, щоб реалізувати багатосторінкову карту команд, деяким командам потрібен додатковий байт перед самою командою, який називається відповідно префіксним байтом.

У наступних пунктах дається опис кожного режиму адресації. У цих пунктах термін "ефективна адреса" означає адресу в пам'яті, з якої байт зчитується, записується, або з якої почнеться процес виконання програми.

Безпосередня адресація (IMM)

При безпосередній адресації, операнд знаходиться в байті (або байтах), що містяться безпосередньо за командою. Число байт залежить від розміру регістра. Такі команди можуть складатися із двох, трьох або чотирьох (якщо потрібен байт префікса) байт.

Пряма адресація (DIR)

При прямому режимі адресації (іноді його називають ще режимом адресації нульової сторінки) молодший байт адреси операнда знаходиться в байті, що слідує за кодом команди, старший байт адреси при цьому вважається рівним $00. Пряма адресація дозволяє користувачеві використовувати комірки пам'яті з адресами $0000-$00FF за допомогою двохбайтних команд, зменшуючи при цому час виконання шляхом відмови від додаткового звертання до пам'яті. У більшості застосувань ця 256-байтна область резервується для даних. У мікроконтролері MC68HC11E9 можна в такий спосіб програмно налаштувати карту пам'яті, щоб на ці адреси відображалися внутрішній ОЗП або внутрішні регістри чи зовнішній адресний простір На платі, що досліджується, користувачу доступні тільки адреси з $0000 по $0035, а інше місце, з $0036 - $00FF займає програма-монітор BUFFALO. Також користувач може використовувати зовнішній ОЗП з адреси $3000 по адресу $DFFF у випадку розширеного режима адресації.

Розширена адресація (EXT)

При використанні режиму розширеної адресації другий і третій байти (що розміщені за кодом команди) містять абсолютні адреси операнда. Команди з розширеною адресацією можуть складатися із трьох або чотирьох (якщо є байт префікса) байт: один або два байти для коду команди й два байти для "ефективної" адреси.

Індексна адресація (IND, X; IND, Y)

У режимі індексної адресації для обчислення "ефективної" адреси використовується один з індексних регістрів (X або Y). У цьому випадку "ефективна" адреса складається з двох величин: поточного вмісту індексного регістра й восьмиразрядного беззнакового зсуву, що визначається кодом команди. Цей режим адресації дозволяє мати доступ до всіх 64 кБ адресного простору. Код команди складається зазвичай з двох або трьох (якщо є присутнім байт префікса) байт; один або два байти для коду команди плюс восьмирозрядний зсув.

Неявна адресація (INH)

Цей режим адресації означає, що вся інформація про операндах визначається кодом команди. А якщо є операнди, то це обов'язково регістри; при цьому посилання на пам'ять відсутнє. Команди з неявною адресацією як правило складаються з одного або двох байт. Таким чином команди з неявною адресацією або не мають операндів, або містять вказівку на

операнд у мнемоніці команди. До таких команд відносяться, наприклад, команди повернення з переривання (RTI), зупинки (STOP) і т.д. Неявний тип адресації мають команди роботи з даними в регістрах мікроконтролера, наприклад, встановлення прапорця переносу (SEC), збільшення числа, що зберігається в акумуляторі, на одиницю (INCA), та інші. Команди з неявною адресацією не вимагають звертання до пам'яті й мають довжину один байт.

Відносна адресація (REL)

Відносна адресація використовується в командах розгалуження. Якщо виконується умова розгалуження, то вміст восьмирозрядного знакового байта, що слідує за кодом команди (зсув), додається до вмісту програмного лічильника для формування ефективної адреси, на який і буде здійснено перехід; у протилежному випадку виконання триває з команди, що іде за командою розгалуження. Код команди складається зазвичай із двох байт.

Команди передачі даних реалізують: завантаження операндів з пам'яті даних у регістри; запис вмісту регістрів у пам'ять; операції передачі даних між регістрами.

У випадку завантаження операндів з пам'яті до регістрів можна застосовувати усі способи адресації, крім відносної. А при проведенні операцій запису вмісту регістрів до комірок пам'яті можна ще не використовувати безпосередню адресацію.

1) LDAA (opr)

Тлумачення. Ця команда реалізує операцію завантаження вмісту комірки пам'яті даних (ПД) у акумулятор-регістр А.
(M) → A

2) LDAB (opr)

Тлумачення. Ця команда реалізує операцію завантаження вмісту комірки пам'яті даних (ПД) у акумулятор-регістр В.
(M) → B

3) LDX (opr)

Тлумачення. Завантаження вмісту 8-розрядної комірки ПД до старшого байту індексного регістру IX.
$(M) \rightarrow X_h$

4) LDY (opr)

Тлумачення. Завантаження вмісту 8-розрядної комірки ПД до старшого байту індексного регістру IY.
$(M) \rightarrow Y_h$

5) STAA (opr)

Тлумачення. Пересилка вмісту акумулятора А до комірки ПД, адреса якої вказується після мнемоніки команди.
A → (M)

6) STAB (opr)

Тлумачення. Пересилка вмісту акумулятора В до комірки ПД, адреса якої вказується після мнемоніки команди.
B → (M)

7) STX (opr)

Тлумачення. Пересилка вмісту 16-розрядного індексного регістру IX у дві комірки ПД послідовно M_h; $M+1_l$.
$X \rightarrow (M)_h; (M+1)_l$

8) STY (opr)

Тлумачення. Пересилка вмісту 16-розрядного індексного регістру IY у дві комірки ПД послідовно M_h; $M+1_l$.
$Y \rightarrow (M)_h; (M+1)_l$

9) TAB

Тлумачення. Команда виконує операцію копіювання вмісту акумулятора А до акумулятора В.
A → B

10) TBA

Тлумачення. Команда виконує операцію копіювання вмісту акумулятора B до акумулятора A.
$B \rightarrow A$

11) CLR (opr)

Тлумачення. Відбувається стирання інформації в комірки ПД, адреса якої вказується.

12) CLRA

Тлумачення. Обнуління вмісту акумулятора A.

13) CLRB

Тлумачення. Обнуління вмісту акумулятора B.

14) CLRX

Тлумачення. Обнуління вмісту індексного регістра IX.

15) CLRY

Тлумачення. Обнуління вмісту індексного регістра IY.

16) PSHA

Тлумачення. Запис вмісту акумулятора A до стеку.
$A \rightarrow (SP)_l$

17) PSHB

Тлумачення. Запис вмісту акумулятора B до стеку.
$B \rightarrow (SP)_l$

Особливість При використанні команд зазначених у п.16 та п.17 покажчик стеку зміщується (змінюється позиція): SP-1 → SP (тобто покажчик стеку зменшується на одиницю).

18) PSHX

Тлумачення. Завантаження/передача вмісту 16-розрядного індексного регістру IX до стеку (першим передається молодший байт регістра).
$X \rightarrow (SP)_{l,h}$

19) PSHY

Тлумачення. Завантаження/передача вмісту 16-розрядного індексного регістру IY до стеку (першим передається молодший байт регістра).
$Y \rightarrow (SP)_{l,h}$

20) PULA

Тлумачення. Команда передачі вмісту стека SP до акумулятора A.
$(SP)_h \rightarrow A$

21) PULB

Тлумачення. Команда передачі вмісту стека SP до акумулятора B.
$(SP)_h \rightarrow B$

Особливість При використанні команд зазначених у п.20 та 21 покажчик стеку зміщується (змінюється позиція): SP+1 → SP (тобто покажчик стеку збільшується на одиницю).

22) PULX

Тлумачення. Передача даних зі стеку у 16-розрядний індексний регістр IX (першим передається старший байт).
$(SP)_{h,l} \rightarrow X$

23) PULY

Тлумачення. Передача даних зі стеку у 16-розрядний індексний регістр IY (першим передається старший байт).
$(SP)_{h,l} \rightarrow Y$

Особливість. При виконанні команд наведених у п.22 та п.23 покажчик стеку зміщується на 2 позиції (збільшується).

24) TAP

Тлумачення. Команда копіювання вмісту акумулятора А до регістра ознак CCR (порозрядно).
A → (CCR)
А точніше
- 0 розряд → ознака переносу C;
- 1 розряд → ознака переповнення V;
- 2 розряд → ознака нульового результату Z;
- 3 розряд → ознака знаку N;
- 4 розряд → ознака заборони переривань I;
- 5 розряд → ознака переносу між тетрадами H;
- 6 розряд → ознака X;
- 7 розряд → ознака S.

25) TPA

Тлумачення. Копіювання вмісту регістра ознак CCR до акумулятора А.
(CCR) → A

26) TXS

Тлумачення. Команда виконує пересилання даних з індексного регістру IX до стеку.
(X) → (SP)

27) TYS

Тлумачення. Команда виконує пересилання даних з індексного регістру IY до стеку.
(Y) → (SP)

28) TSX

Тлумачення. Пересилання даних зі стеку до індексного регістру IX.
(SP) → (X)

29) TSY

Тлумачення. Пересилання даних зі стеку до індексного регістру IY.
(SP) → (Y)

Особливість При виконанні команд зазначених в п.26÷29 не відбувається зміщення позиції покажчика стеку.

Основні команди зсуву даних

1) **ASL** {Арифметичний зсув ліворуч}

Тлумачення. Ця команда здійснює зсув усіх бітів акумулятора чи комірки пам'яті М на одну позицію вліво (мається на увазі біти вмісту). У біт "0" розміщується значення 0. А старший розряд акумулятора чи комірки пам'яті зміщується у ознаку С регістра CCR (ознака переносу).
C ← біт 7 ..... біт 0 ← 0

Формат: ASLA; ASLB; ASL (opr)

Особливість Ознака С регістру CCR встановлюється, якщо перед операцією зсуву, було встановлено старший розряд акумулятора або комірки пам'яті , інакше відбувається скидання ознаки.

2) **ASLD** {Арифметичний зсув ліворуч акумулятора D}

Тлумачення. Ця команда здійснює зсув усіх бітів акумулятора на одну позицію вліво (мається на увазі біти вмісту). У біт "0" розміщується значення 0. А старший розряд акумулятора чи комірки пам'яті зміщується у ознаку С регістра CCR (ознака переносу).
C ← біт 7 ..... біт 0 ← біт 7 ..... біт 0 ← 0
Акумулятор А Акумулятор В

3) **ASR** {Арифметичний зсув праворуч}

Тлумачення. Ця команда здійснює зсув усіх бітів акумулятора чи комірки пам'яті M на одну позицію вправо (мається на увазі біти вмісту). Біт 7 залишається незмінним . Біт 0 переміщується у біт C регістра ознак CCR.
біт 7 ..... біт 0 → 0

Формат: ASRA; ASRB; ASR (opr)

Особливість. Ознака C встановлюється, якщо перед операцією зсуву, було встановлено молодший розряд акумулятора або комірки пам'яті, інакше відбувається скидання ознаки.

4) **LSL** {Логічний зсув ліворуч}

Тлумачення. Ця команда здійснює зсув усіх бітів акумулятора чи комірки пам'яті M на одну позицію вліво (мається на увазі біти вмісту).
C ← біт 7 ..... біт 0 ← 0

Формат: LSLA; LSLB; LSL (opr)

5) **LSLD** { Логічний зсув ліворуч акумулятора D}

Тлумачення. Ця команда здійснює зсув усіх бітів акумулятора на одну позицію вліво (мається на увазі біти вмісту). У біт "0" розміщується значення 0. А старший розряд акумулятора чи комірки пам'яті зміщується у ознаку C регістра CCR (ознака переносу).
C ← біт 7 ..... біт 0 ← біт 7 ..... біт 0 ← 0
Акумулятор A Акумулятор B

Особливість. Ознака C встановлюється, якщо перед операцією зсуву, було встановлено старший розряд акумулятора, інакше відбувається скидання ознаки.

6) **LSR** { Логічний зсув праворуч}

Тлумачення. Ця команда здійснює зсув усіх бітів акумулятора чи комірки пам'яті M на одну позицію вправо (мається на увазі біти вмісту). Біт 7 залишається незмінним . Біт 0 переміщується у біт C регістра ознак CCR.
0 → біт 7 ..... біт 0 → C

Формат: LSRA; LSRB; LSR (opr)

Особливість. Ознака C встановлюється, якщо перед операцією зсуву, було встановлено молодший розряд акумулятора або комірки пам'яті, інакше відбувається скидання ознаки.

7) **LSRD** { Логічний зсув праворуч акумулятора D}

Тлумачення. Ця команда здійснює зсув усіх бітів акумулятора на одну позицію вліво (мається на увазі біти вмісту). У біт "0" розміщується значення 0. А старший розряд акумулятора чи комірки пам'яті зміщується у ознаку C регістра CCR (ознака переносу).
0 → біт 7 ..... біт 0 → біт 7 ..... біт 0 → 0
Акумулятор A Акумулятор B

Особливість. Ознака C встановлюється, якщо перед операцією зсуву, було встановлено молодший розряд акумулятора, інакше відбувається скидання ознаки.

8) **ROL** {Циклічний зсув ліворуч}

Тлумачення. Відбувається переміщення усіх бітів акумулятора або комірки пам'яті M на одну позицію вліво циклічно. У біт "C" регістра CCR заноситься вміст старшого розряду акумулятора чи комірки пам'яті.
C ← біт 7 ..... біт 0 ← C

Формат: ROLA; ROLB; ROL (opr)

Особливість. Ознака C встановлюється, якщо перед операцією циклічного зсуву, було встановлено старший розряд акумулятора або комірки пам'яті, інакше відбувається скидання ознаки.

9) **ROR** {Циклічний зсув праворуч}

Тлумачення. Відбувається переміщення усіх бітів акумулятора або комірки пам'яті M на одну позицію вправо циклічно. У біт "C" регістра CCR заноситься вміст молодшого розряду акумулятора чи комірки пам'яті.
C → біт 7 ..... біт 0 → C

Формат: RORA; RORB; ROR (opr)

Особливість. Ознака C встановлюється, якщо перед операцією циклічного зсуву, було встановлено старший розряд акумулятора або комірки пам'яті, інакше відбувається скидання ознаки.

10) **SEC**

Тлумачення. Встановлення ознаки C регістра CCR у положення 1.
C ← "1"

11) **CLC**

Тлумачення. Скидання ознаки C регістра CCR, тобто переведення її у положення 0.
C ← "0"

Робоче завдання

1. Написати програму на мові Асемблера за варіантом завдання, що вказується викладачем.
2. Продемонструвати програму викладачу та результати виконання програми, відповідно.
3. Занести до протоколу усі повідомлення, що виводяться програмою-монітором у діалогове вікно при виконанні програми, текст і лістинг програми.

Варіанти робочих завдань лабораторної роботи

1. Заповнити адреси $0000...$0015 значеннями #67. Обнулити вміст акумулятора A та індексного регістра IX.
2. Завантажити вміст регістра ознак у осередок $0020.
3. Здійснити обмін даними між стеком та індексним регістром IX. А також здійснити передачу вмісту акумулятора A до регістра ознак.

4. Занесіть у подвійний акумулятор D дані, що містяться у комірці з адресою $0030 і після цього скопіюйте його у покажчик стека. Завантажте в акумулятор A вміст осередку пам'яті, на яку вказує покажчик стека, а в акумулятор B – вміст комірки пам'яті, адреса якої на одиницю більше. Поміняйте місцями молодшу та старшу тетради акумулятора B.
5. Використовуючи режим індексної адресації завантажте в індексний регістр IY вміст осередка пам'яті з адресою $1234.
6. Повторити п.5, тільки занесіть дані з осередку, адреса якого на $13 менша ніж $1234. Вміст осередку скопіюйте також до індексного регістра IX.
7. Занесіть вміст комірки ПД $0031, $0032 до індексного регістра IY. А після цього вміст цього регістра скопіюйте до стеку.
8. Занесіть в акумулятор A вміст комірки пам'яті з номером $002F. Поміняйте місцями молодшу та старшу тетради акумулятора A.
9. Скопіювати вміст регістра ознак у комірку пам'яті з адресою $0014. Занесіть у регістр ознак вміст осередку, на який вказує покажчик стеку.
10. Занесіть в акумулятор B вміст комірки пам'яті з номером $003E. Поміняйте місцями молодшу та старшу тетради акумулятора B.

Контрольні питання

1. Які групи команд складають систему команд мікроконтролера MC68HC11?
2. Які режими адресації підтримує мікроконтролер MC68HC11?
3. Що таке "ефективна адреса" даних? Наведіть приклади команд, де використовується безпосередня адресація.
4. Яким чином "знаходиться" адреса операнда у випадку використання прямої адресації? Наведіть приклад рядка програми, де застосовано вказаний режим адресації.
5. Яким чином "знаходиться" адреса операнда у випадку використання розширеної адресації? Наведіть приклад рядка програми, де застосовано вказаний режим адресації.

6. Яким чином "знаходиться" адреса операнда у випадку використання індексної адресації? Наведіть приклад рядка програми, де застосовано вказаний режим адресації.

7. Яким чином "знаходиться" адреса операнда у випадку використання неявної адресації? Наведіть приклад рядка програми, де застосовано вказаний режим адресації.

8. В яких випадках в програмах використовується режим відносної адресації? Наведіть особливості застосування цього режиму адресації.

Лабораторна робота №4

Тема роботи – Ознайомлення з особливостями застосування на мові Асемблера системи команд керування програмою та процесором, вивчення команд умовного розгалуження.

Мета роботи – Вивчити основні команди керування програмою та процесором, отримати навички та вміння щодо застосування команд умовного розгалуження.

Домашнє завдання

1) *Ознайомитися оглядово із основними командами умовного розгалуження.*

2) *Вивчити основні команди керування програмою, що реалізуються на мові Асемблера.*

3) *Вивчити основні команди керування процесором мікроконтролера MC68HC11E9.*

Теоретичні відомості

Основні команди керування програмою та процесором

Команди умовного переходу, що використовується в тілі програми, дають можливість користувачу здійснити перехід до підпрограми, яка реалізована на мові Асемблера та вихід з неї відповідно. Крім цього у структурі цього блоку команд є такі, що допомагають здійснити виконання програмного переривання та реалізують вихід з цього переривання без втрати корисної інформації. Окрема частина команд призначена для переходу процесору мікроконтролера у різні режими функціонування.

Як було зазначено вище, відносна адресація використовується в командах розгалуження. Якщо виконується умова розгалуження, то вміст восьмирозрядного знакового байта, який слідує за кодом команди (зсув), додається до вмісту програмного лічильника для формування ефективної адреси, на яку і буде здійснений перехід; у протилежному випадку виконання триває з команди, що слідує за командою розгалуження. Код команди розгалужено складається як правило з двох байт.

1) **JMP (opr)**

Тлумачення. Ця команда реалізує операцію безумовного переходу. При виконанні цієї команди компілятором програми мікроконтролера, відбувається завантаження у програмний лічильник PC адреси наступної команди, що задається полем (opr). Іншими словами, адреса переходу та відповідно зміна керування програмою визначається полем операнда.
$addr → (PC)

2) JSR (opr)

Тлумачення. Ця команда реалізує операцію переходу до підпрограми за дійсною адресою. Тобто команда завантажує у програмний лічильник PC адресу початку підпрограми.

3) RTS

Тлумачення. Команда виходу з підпрограми. За цією командою відбувається вивантаження зі стеку вмісту програмного лічильника PC, яке було до виконання та передачі керування у підпрограму.

4) SWI

Тлумачення. Ця команда реалізує програмне переривання програми. Вона виконує завантаження у стек вміст програмного лічильника PC, а також дані що знаходяться в регістрах процесора IX, IY, A, B, CCR. При виконанні цієї команди заповнюються 9 комірок стекової пам'яті.

5) RTI

Тлумачення. Команда виходу з переривання. Вона реалізує відновлення вмістів регістрів CCR, A,B, IX, IY, PC зі стеку, які були до виконання запиту на переривання.

6) RSP

Тлумачення. Команда здійснює встановлення у регістрі стеку SP початкового значення $00FF.

$00FF → (SP)

7) NOP

Тлумачення. Команда не виконує ніяких дій. Проте відбувається пропущення байту (машинного циклу).

8) WAIT

Тлумачення. Перехід процесора мікроконтролера у режим очікування (зупинка процесора).

9) STOP

Тлумачення. Перехід процесора мікроконтролера у режим останова (зупинення ГТІ).

Особливість. Команди, що наведені у п.8 та п.9 дозволяють виконувати операцію переривання.

Основні команди умовного розгалуження

Ці команди використовують тільки режим відносної адресації. Якщо виконується умова розгалуження то адреса, куди буде здійснений перехід визначається як сума:

1) вмісту програмного лічильника PC;
2) числа $0002;
3) відносного зміщення у 1 байт (rel8 ⇒ 2 символи).

Якщо ж умова розгалуження не виконується то програмний лічильник переміщується на 2 позиції (2 байта, оскільки команди умовного розгалуження є командами на які використовується 2 машинні цикли процесора)

Формат команд умовного розгалудження: B_ _ rel8
– де 2 позиції заповнюються у відповідності с синтаксисом команди.

1) BCC (opr)

Тлумачення. Перехід за адресою, що вказується у полі (opr) та у відповідності з вище наведеним правилом, якщо ознака C у регістрі CCR (ознака переносу) прийме значення (C)=0.
Перехід за розрахованою адресою, якщо (C)="0"

2) BCS (opr)

Тлумачення. Перехід за адресою у випадку, якщо попередня арифметична операція призвела до переносу, тобто у регістрі ознак CCR прапорець C у положення "1".
Перехід за розрахованою адресою, якщо (C)="1"

3) BNE (opr)

Тлумачення. Перехід за адресою, у випадку коли результат попередньої операції ненульовий, тобто прапорець Z у регістрі ознак CCR приймає значення "0".
Перехід за розрахованою адресою, якщо (Z)="0"

4) BEQ (opr)

Тлумачення. Перехід здійснюється у випадку якщо результат попередньої операції нульовий, тобто прапорець Z у регістрі ознак CCR приймає значення "1".
Перехід за розрахованою адресою, якщо (Z)="1"

5) BPL (opr)

Тлумачення. Здійснюється перехід, якщо результат попередньої операції додатній, тобто прапорець N у регістрі ознак CCR приймає значення "0".
Перехід за розрахованою адресою, якщо (N)="0"

6) BMI (opr)

Тлумачення. Здійснюється перехід, якщо результат попередньої операції від'ємний, тобто прапорець N у регістрі ознак CCR приймає значення "1".
Перехід за розрахованою адресою, якщо (N)="1"

7) BVC (opr)

Тлумачення. Здійснюється перехід за розрахованою адресою, якщо результат попередньої операції не призвів до

переповнення, тобто ознака V у регістрі CCR приймає нульове значення.
Перехід за розрахованою адресою, якщо (V)="0"

8) BVS (opr)

Тлумачення. Перехід за адресою, якщо в результаті виконання попередньої операції виникло переповнення розрядів, тобто ознака V у регістрі CCR прийняла значення "1".
Перехід за розрахованою адресою, якщо (V)="1"

9) BRA (opr)

Тлумачення. Безумовний перехід за адресою, і крім цього вміст програмного лічильника PC змінюється наступним чином:
(PC)+2+rel8⇒(PC)

10) BRN (opr)

Тлумачення. Команда не виконуваного переходу. Іншими словами відбувається зміна вмісту програмного лічильника на 2 байти.

11) BGE (opr) {Перехід, якщо більше чи дорівнює}

Тлумачення. Перехід за визначеною адресою відбудеться якщо (ACC)≥(M) (за вмістом у двійкових цифрах).
(PC)+2+rel8⇒(PC)

12) BGT (opr) {Перехід, якщо більше }

Тлумачення. Перехід за визначеною адресою відбудеться якщо (ACC)>(M) (за вмістом у двійкових цифрах).
(PC)+2+rel8⇒(PC)

13) BLE (opr) {Перехід, якщо менше чи дорівнює}

Тлумачення. Перехід за визначеною адресою відбудеться якщо (ACC)≤(M) (за вмістом у двійкових цифрах).
(PC)+2+rel8⇒(PC)

14) BLO (opr) {Перехід, якщо менше }

Тлумачення. Перехід за визначеною адресою відбудеться якщо (ACC)<(M) (за вмістом у двійкових цифрах).
(PC)+2+rel8⇒(PC)

Основні команди, що дозволяють сформувати цикл у тілі програми

1) CBA

Тлумачення. Команда порівняння вмісту акумулятора А з вмістом акумулятора В. В даному випадку здійснюється віднімання операндів, які знаходяться у регістрах/акумуляторах А та В відповідно, без запису цього результату віднімання, але зі встановленням у регістрі CCR ознак N,Z у відповідності з отриманим значенням.

2) CMPA (opr)

Тлумачення. Здійснюється порівняння між вмістом акумулятора А та вмістом комірки пам'яті, адреса якої задається у полі (opr). Результатом цього порівняння є встановлення відповідної ознаки N,Z у регістрі CCR. Ця команда також дозволяє вказувати безпосередньо операнд для порівняння.

3) CMPB (opr)

Тлумачення. Здійснюється порівняння між вмістом акумулятора В та вмістом комірки пам'яті, адреса якої задається у полі (opr). Результатом цього порівняння є встановлення відповідної ознаки N,Z у регістрі CCR. Ця команда також дозволяє вказувати безпосередньо операнд для порівняння.

4) CPD (opr); CPX (opr); CPY (opr);

Тлумачення. Відбувається порівняння вмісту регістрів D, IX, IY з даними, що містяться у полі операнда. У полі (opr) можна вказати, або число (константу) у 16-ковій системі числення, або

адресу однієї комірки пам'яті (де міститься старший байт даних), а молодший байт міститься у сусідній комірці.

5) TST (opr)

Тлумачення. Команда тестування на знак, нульове значення вмісту комірки пам'яті, адреса якої задається у полі (opr). Результатом виконання цієї команди є встановлення у регістрі CCR відповідник ознак N,Z.

6) TSTA; TSTB;

Тлумачення. Команда тестування на знак, нульове значення вмісту акумулятора А, В. Результатом виконання цієї команди є встановлення у регістрі CCR відповідник ознак N,Z.

7) INC (opr)

Тлумачення. Здійснюється збільшення на одиницю вмісту комірки пам'яті даних, адреса якої задається командою.

8) INCA; INCB; INCX; INCY; INCS;

Тлумачення. Здійснюється збільшення на одиницю вмісту регістрів A,B,IX,IY,SP.

9) DEC (opr)

Тлумачення. Відбувається зменшення на одиницю вмісту комірки пам'яті даних, адреса якої задається командою.

10) DECA; DECB; DECX; DECY; DECS;

Тлумачення. Здійснюється зменшення на одиницю вмісту регістрів A,B,IX,IY,SP.

Робоче завдання

1. Написати програму на мові Асемблера за варіантом завдання, що вказується викладачем.

2. Продемонструвати програму викладачу та результати виконання програми, відповідно.

3. Занести до протоколу усі повідомлення, що виводяться програмою-монітором у діалогове вікно при виконанні програми, текст і лістинг програми.

Варіанти для виконання лабораторної роботи

1. Сформувати програму сортування комірок пам'яті $0090-$00BF за зростанням.

2. Сформувати на асемблері Basic - програму

```
10 LET A=7
20 FOR B=1 TO 8
30 IF B<4 THEN GO SUB 60
40 NEXT B
50 GO TO 80
60 LET A=A+B
70 RETURN
80 STOP
```

3. Напишіть програму підрахунку кількості комірок пам'яті зі значеннями, що відрізняються від #38.

4. Напишіть програму підрахунку кількості від'ємних та додатних чисел, вважаючи при цьому що нуль є нейтральним числом.

5. Нехай у комірках пам'яті $0050, $0051 містяться відповідно змінні A, B. Якщо вони упорядковані за зростанням, то виконати заповнення комірок пам'яті $0090-$00B0 значеннями #15, у противному випадку заповнити значеннями #36.

6. Нехай в комірці $0080 знаходиться деяке число. Перевірити, чи можна це число без остачі поділити на #03.

7. Нехай в комірці $0080 знаходиться деяке число. Перевірити, чи можна це число без остачі поділити на #0A.

8. Сформувати програму сортування значень комірок пам'яті визначеної області за зменшенням.