НАВЧАЛЬНО-МЕТОДИЧНА ЛІТЕРАТУРА

Тернопільський національний технічний університет імені Івана Пулюя

Кафедра автоматизації технологічних процесів і виробництв

Лабораторна робота № 1 з курсу ”Мікропроцесорні та програмні засоби автоматизації”

Програмування МП і8086 з використанням програмного емулятора Emu8086

Тернопіль 2018

Методичні вказівки до лабораторної роботи №1. Програмування МП і8086 з використанням програмного емулятора Emu8086 з курсу «Мікропроцесорні та програмні засоби автоматизації». Медвідь В.Р., Пісьціо В.П., - Тернопіль: ТНТУ, 2018 - 8 с.

Відповідальні за випуск доцент, к.т.н. Медвідь В.Р.,
асистент Пісьціо В.П.

Для студентів напряму: 151 "Автоматизація та комп'ютерно-інтегровані технології"

Лабораторна робота №1

ПРОГРАМУВАННЯ МП i8086 З ВИКОРИСТАННЯМ ПРОГРАМНОГО ЕМУЛЯТОРА “emu8086”

1. РОБОТА ЕМУЛЯТОРА emu8086

Для виконання роботи використовується програмний емулятор мікропроцесора emu8086. Вікно програми має слідуючий вигляд (рис. 1):

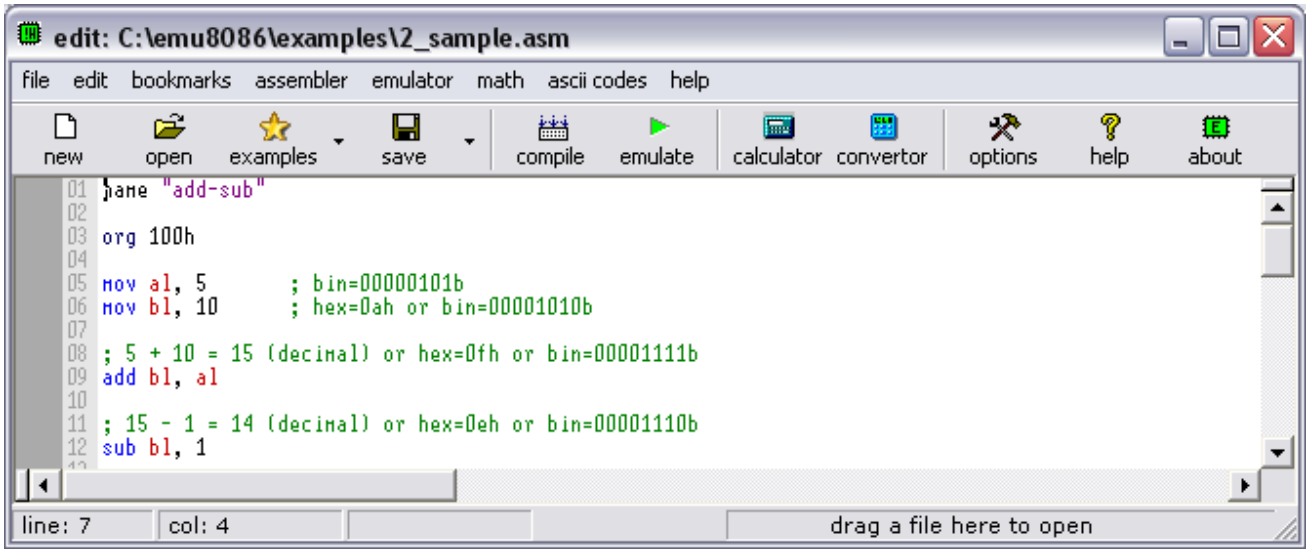

Рис.1

Текст програми набирається в полі редактора порядково.

Коментарі записуються після знаку ";".

Числа допускається записувати в будь-якій системі числення, при цьому після числа ставиться позначення: **h - шістнадцяткове, b - бінарне, o - вісімкове, наприклад**: 1ah, 101b, 71o.

Числа без позначення вважаються десятковими.

Якщо в записі шістнадцяткового числа старшим розрядом є літера (A..F), то перед нею необхідно поставити «0».

Для створення циклів, умовних і безумовних переходів використовуються команди LOOP, JMP, JA, JC і т.п. і мітки виду l**abel:**

В кінці програми рекомендується ставити команду **RET**.

Повний список і опис усіх команд процесора можна знайти в меню help: Documentation and tutorials / 8086 Instruction Set.

Для запуску набраної програми слід натиснути на кнопку **emulate** на панелі інструментів. При цьому відкривається вікно емулятора (рис.2):

У вікні емулятора можна управляти режимом роботи програми: кнопками
single step дозволяє використовувати покроковий режим,
run - запустити програму на виконання.

У лівій частині вікна виводяться значення всіх регістрів загального призначення, їх можна переглядати і змінювати по ходу виконання програми; крім того, подвійне клацання миші по віконцях зі значенням регістра дозволяє вивести на екран вікно розширеного перегляду значень регістрів в різних кодуваннях.

У центральній частині вікна емулятора знаходяться номери і вміст комірок пам'яті (підсвічуються комірки, відповідні наступному рядку програми), в правій частині - оброблений текст виконуваної програми.

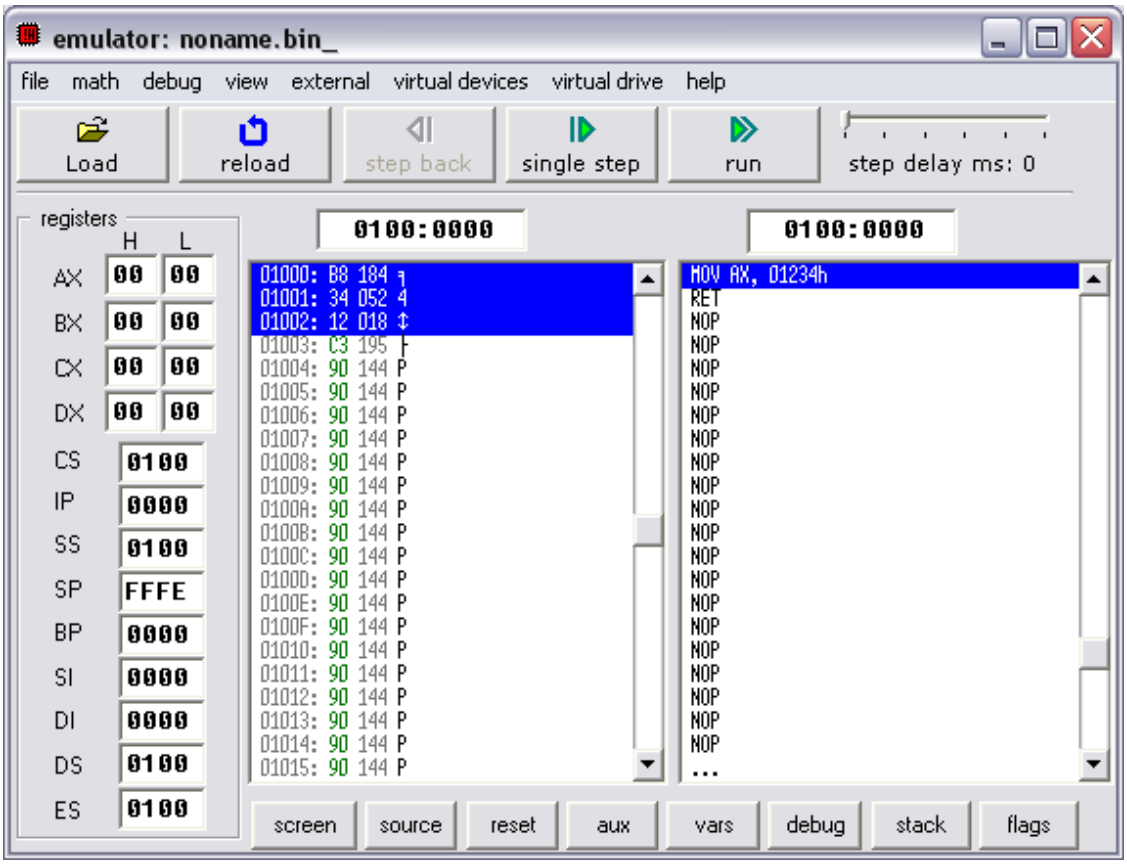

Рис.2

Вміст регістру флагів можна переглядати і змінювати за допомогою кнопки **flags** в лівому нижньому кутку.

За допомогою регулятора **step delay** можна встановлювати затримку між виконанням кроків програми.

2. Пслідовність виконання роботи

Ознайомлення з работою емулятора:

1) запустіть программу emu8086.exe.

В вікні привітання виберіть варіант **new**, далі виберіть **empty workspace**.

2) для прикладу, напишіть п**рограму завантаження числа 1234h в регістр AX**:

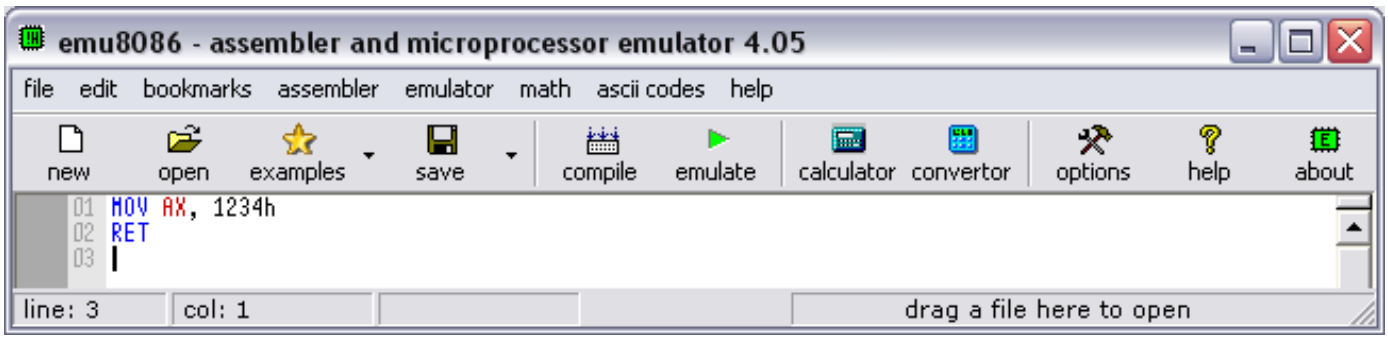

Рис.3

3) запустіть її натисканням **клавіші F5** або кнопки **emulate** на панелі інструментів, відкриються вікно емулятора (рис.2) і коду програми (рис.4):

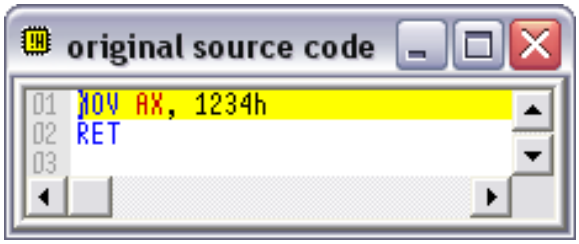

Рис.4

4) запустіть програму на виконання в **крововому режимі** і простежте за зміною значення регістрів AX і IP (**вміст змінених при виконанні команди регістрів виділяється синім кольором**), потім перезавантажте програму натисканням кнопки **reload** і виконайте її за допомогою кнопки **run.**

5) перейдіть до вмісту регістра флагів, натиснувши кнопку **flags**; змініть значення флагу паритету PF на 1.

II. ЗАВДАННЯ для виконання лабораторної роботи «ПРОГРАМУВАННЯ МП i8086 З ВИКОРИСТАННЯМ ПРОГРАМНОГО ЕМУЛЯТОРА “emu8086”»

1. Виконується на самостійній підготовці

1.1. Ознайомитися з роботою программного емулятора emu8086 мікропроцесора i8086, користуючись методичними вказівками.

1.2. Написати на самостійній підготовці програми відповідно до Вашого варіанту, що виконують наступні дії:

ВАРІАНТ 1

Команда завантаження і пересилання даних (команда MOV)

1) завантажити регістр CX числом ABCDh,
2) переслати вміст CX в регістр AX,
3) завантажити комірку пам'яті поточного сегменту без зміщення (зміщення 0) числом FFh,
4) задати в якості бази сегменту DS = 200h, попередньо записавши це число в регістр DX; завантажити комірку 02100h числом AAh (зміщення 100h),
5) записати в регістр BX зміщення 200h і завантажити комірку 02200h числом BBh (зміщення [BX]),
6) записати в регістр SI індекс 20 і завантажити комірку 02220h числом CCh, використовуючи сегмент з базою DS = 200h і зміщення [BX + SI],
7) завантажити комірку 02222h числом DDh, використовуючи сегмент з базою DS = 200h і зміщення [BX + SI + 2].

Одноопернадні команди

1) завантажити регістр DX числом 5,
2) збільшити вміст регістру DX на одиницю (команда INC),
3) перенести вміст регістра DX в комірку пам'яті 01100h,
4) зменшити вміст комірки 01100h на одиницю (команда DEC),
5) інвертувати вміст регістру DX (команда NOT).

ВАРІАНТ 2

Двоопернадні команди

а) Арифметичні операції:

1) додати числа 1Bh і 1Ch, використовуючи регістри BX і CX, результат записати в CX (команда ADD),
2) відняти від вмісту регістра CX константу 10b, результат записати в AX (команда SUB),
3) вміст регістру AX розділити на 10h, попередньо записавши дільник в регістр BX (команда DIV)

4) залишок від ділення (знаходиться в регістрі DX) помножити на 2 (команда MUL).

б) Логічні операції:
1) записати в комірку пам'яті 01100h число 101101b, в регістр AX число 110001b, виконати для них логічне множення, результат записати в комірку 01100h (команда AND),
2) записати в регістр AX число 111010b, виконати роз'єднання між вмістом AX і комірки пам'яті 01100h, результат записати в регістр AX (команда OR),
3) виконати операцію "сума по модулю 2" для вмісту регістра AX і числа 001001b (команда XOR).

ВАРІАНТ 3

Команди зсуву

1) завантажити в регістр BX число 25h, помножити його на 2, використовуючи операцію арифметичного зсуву (команда SAL),
2) завантажити в комірку пам'яті 01200h число 40h, розділити його на 4, використовуючи операцію арифметичного зсуву (команда SAR).

Команди управління програмою

а) Цикл (команда LOOP)
1) завантажити в регістр CX число 10 (кількість повторень циклу),
2) встановити в якості бази сегменту DS = 0100h, занести в регістр BX зміщення 0101h,
3) встановити мітку початку виконання циклу виду label:
4) тіло циклу: занести число AAh в комірку пам'яті, зміщення якої щодо бази DS зазначено в регістрі BX, і збільшити вміст BX на 1,
5) після запису тіла циклу перейти по мітці label.
Після виконання програми число AAh має бути записано в комірку з 01101h по 0110Ah.

ВАРІАНТ 4

Команди управління програмою

б) Умовний перехід (команда JC)
1) завантажити число FFh в регістр AL,
2) збільшити вміст регістру AL на AAh,
3) якщо біт перенесення не дорівнює одиниці, записати в регістр BX число ABCDh, інакше завершити програму (перехід по мітці до кінця програми).

в) Безумовний перехід (команда JMP)
1) завантажити в ккомірку 01100h число C3h (код команди RET),
2) написати програму додавання чисел Ah і Bh, використовуючи регістри DL і DH,
3) для закінчення виконання програми перейти за адресою 01100h.

2. Виконуєтсья в лабораторіі

Приклад

Приклад використання команди ВВІД/ВИВІД ПО ПРЕРИВАННЮ INT 10H (AH=0):
Це **команда встановлення режиму дисплея**.
В AL вказується номер режиму
AL=0: Текстовий чорно-білий 40x25
AL=1: Текстовий кольоровий 40x25
AL=2: Текстовый чорно-білий 80x25
AL=3: Текстовий кольоровий 80x25
AL=4: Графічний кольоровий 320x200
Наприклад, для встановлення кольорового графічного режиму необхідно:

```
mov ah,0
mov al,4
```

```
int 10h
```

Завдання 1

Записати та виконати команди в кроковому режимі роботи :

```
org 100h      ;задати адресу наступної команди

              ; встановити режим відео
mov ax, 3     ; задаються параметри  дисплею 80x25, 16 colors, 8 pages (ah=0, al=3)
int 10h       ; виконується команда переривання
```

Після виконання кожної з команд записати вміст регістрів МП, які змінюються в результаті їх виконання, в табл.1.

Таблиця 1

Регістр	IP	SP	AH	AL	BH	BL	CH	CL	DH	DL	CS	DS	SS	ES	BP	SI	DI	IF
Команда 1																		
Команда 2																		
Команда 3																		
Команда n																		

Завдання 2

Записати та виконати команди в кроковому режимі роботи:

```
org 100h
                    ; встановити значення сегментного регістру DS
mov   ax, 0b800h    ; записати число в регістр ax
mov   ds, ax        ; скопіювати вміст регістру ax в регістр ds
```

Записати вміст регістрів МП після виконання кожної з команд (табл.1).

Завдання 3

Записати та виконати команди додавання та віднімання в кроковому режимі роботи:

```
org 100h
mov al, 5           ; bin=00000101b
mov bl, 10          ; hex=0ah or bin=00001010b
                    ; 5 + 10 = 15 (decimal) or hex=0fh or bin=00001111b
add bl, al
                    ; 15 - 1 = 14 (decimal) or hex=0eh or bin=00001110b
sub bl, 1
```

Записати вміст регістрів МП після виконання кожної з команд (табл.1).

Завдання 4

Записати та виконати підпрограму, що використовує команду TEST для перевірки значення потрібного біту з регістра, який множиться на 1 (всі решта бітів - на нуль). Виконує операцію логічного множення двох операндів. Не змінює значення регістру.:

```
  mov bl, 8
  test bl, 10000000b
  jz zero
  mov bl, 1
zero:  int 21h
```

Записати вміст регістрів МП після виконання кожної з команд (табл.1).

Записати коментар для кожної з команд.

Завдання 5

Записати та виконати команди, які оперують з числами, записаними в різних системах числення.

```
org 100h
                    ; завантаження двійкового числа:
                    ; (hex: 5h)
mov al, 00000101b
                    ; завантаження шістнадцяткового числа
mov bl, 0ah
                    ; завантаження вісімкового числа
                    ; (hex: 8h)
mov cl, 10o
                    ; 5 + 10 = 15 (0fh)
add al, bl          ; додавання двох чисел, результат записується в перший регістр
                    ; 15 - 8 = 7
sub al, cl          ; віднімання друго числа від першого, результат записується в
                    ;перший регістр.
```

Записати вміст регістрів МП після виконання кожної з команд (табл.1).

Завдання 6

Записати та виконати арифметичні та логічні команди.

```
org   100h
mov   ah, 09h
mov   al, 05h
                          ; al = al + ah =
                          ; = 09h + 05h = 0eh
add   al, ah
xor   ah, ah
```

Записати вміст регістрів МП, що змінюються в результаті виконання команд (табл.1).

Завдання 7

Записати та виконати команди програмування портів.

```
    org  0100h
start:
    mov dx,0fffeh       ;задає адресу регістра керуючого слова порта К580ВВ55
    mov al,099h         ;керуюче слово порта
    out dx,al           ;вивід керуючого слова
L1:                     ;початок циклу
    mov dl,0eah         ;в регістр DX задається адреса регістра стану К580ВВ79
    in al,dx            ;читання стану
    mov dl,00fah        ;зміна адреси на адресу порта B К580ВВ55
    out dx,al           ;вивід стану на світлодіоди
    jmp L1              ;перехід на початок циклу
hlt
```

Записати вміст регістрів МП після виконання кожної з команд (табл.1).

Література.

1. Проектирование цифровых устройств на однокристальных микроконтроллерах / В. В. Сташин и др. - М.: Энергоатомиздат, 1990. – 224 с.
2. Микропроцессоры / Под ред. Преснухина Л.Н., т. 1, 2, 3. - М.: Высшая школа, 1986.
3. Бойко Н.П., Стеклов В.К. Системы автоматического управления на базе микро-ЭВМ. - К.: Техника, 1989. - 182 с.