

## Лабораторна робота №7. Дослідження роботи комп'ютера. Емулятор EMU8086

**Мета роботи:** Ознайомитись із програмними способами емуляції роботи ПК. Вивчити Емулятор Emu8086.

### 1. Теоретичні відомості

#### 1.1 Регістри процесора 8086

Процесор містить 14 швидкодіючих 16-розрядних чарунок пам'яті, які називаються регістрами (див. таблицю 10.1).

Регістри поділяють на регістри загального призначення, сегментні, індексні, вказівники та регістр стану процесора і регістр вказівник команд. До регістрів загального призначення (AX, BX, CX, DX) можна звертатися як безпосередньо, так і побайтово (до молодшого та старшого розряду окремо), наприклад, регістр AX складається з молодшого розряду AL та старшого — AH. Індексні регістри (SI, DI), призначені для зберігання індексів при роботі з рядковими даними. Регістри-вказівники (BP, SP) — використовуються для роботи зі стеком.

Сегментні регістри використовуються для зберігання адрес відповідних сегментів пам'яті. Сегментом називається область пам'яті, яка починається з адреси кратної 16 та містить дані однієї структури. Розрізняють три головні сегменти програми: сегмент коду (CS) — містить інструкції програми; сегмент даних (DS) — містить дані, задані програмістом; сегмент стеку (SS) — область пам'яті, доступ до якої організований за принципом LIFO (last in, first out — останній зайшов, перший вийшов). Також існує розширений сегмент (ES), який використовується, наприклад, при роботі з рядковими даними.

Регістр стану (регістр прапорів) визначає стан процесору після виконання кожної команди. Для мікропроцесорів Intel 8086 активними є 9 біт з 16. Ці біти називають прапорами стану. Регістр-вказівник команд (IP) містить адресу команди (в сегменті коду). Під час виконання програми значення вказівника збільшується на розмір виконаної команди. Існують команди, які змінюють значення IP для реалізації переходів всередині програми.

Табл. 1 – Регістри процесору Intel 8086

Ім'я	English	Призначення
<i>Регістри загального призначення</i>		
AX	Accumulator	Акумулятор (AH, AL), надшвидкий регістр, зв'язаний безпосередньо з АЛП (арифметико-

		логічним пристроєм) мікропроцесора.
BX	Base	Базовий регістр ( <i>BH, BL</i> ), використовується при розширеній адресації.
CX	Counter	Регістр-лічильник ( <i>CH, CL</i> ), керує числом повторень у циклах.
DX	Data	Регістр даних ( <i>DH, DL</i> ), використовується для введення/виведення даних та для обчислень з подвійною точністю, або великими числами при недостатній розрядності акумулятора.
Індексні регістри		
SI	Source Index	Індексування джерела ( <i>SI</i> )/приймача( <i>DI</i> ), використовуються при обробці рядкових даних та індексування масивів даних.
DI	Destination Index	
Регістри – вказівники		
BP	Base Pointer	Вказівник бази
SP	Stack Pointer	Вказівник стека
Сегментні регістри		
CS	Code Segment	Сегмент коду, не можна змінити напряму.
SS	Stack Segment	Сегмент стеку.
DS	Data Segment	Сегмент даних.
ES	Extension Segment	Розширений сегмент даних.
Вказівник команди		
IP	Instruction Pointer	Зміщення команд, програмно не доступний.
Регістр прапорів		
FLAGS		Інформація про поточний стан процесора.

## 1.2 Синтаксис Асемблера

Програма на мові Асемблера є послідовністю операторів, що описують виконувані дії. Оператором (рядком) початкової програми може бути або команда, або псевдооператор (директива) Асемблера.

Команди представляють коротку нотацію (запис) системи команд. У деякому керівництві вони називаються машинними командами, оскільки саме вони повідомляють процесор, які дії необхідно виконувати. На відміну від команд псевдооператори повідомляють (транслятору), що йому робити з командами і даними, які вводяться в програму.

Команда може включати до 4-х полів наступного виду:

[мітка:] мнемокод [операнд] [; коментарій]

Приклад команди з усіма полями:

exit: MOV CX, DX ; помістити вміст регістра DX в регістр CX

Оскільки в [ ] вказуються необов'язкові поля, отже, команда обов'язково повинна містити мнемокод виконуваної дії. Поля можуть набиратися у будь-якому місці рядка, але відділяйте поля один від одного хоч би одним пропуском і, якщо хочете розібратися у своїй програмі по витіканню часу, потурбуйтеся про читабельність, що найчастіше забезпечується за рахунок позиціонування полів.

Мітка, команда та операнд відокремлюється один від одного символом пробілу або табуляції, коментар починається від символу «;» і до кінця рядка. Коментар не відноситься до машинного коду і ігнорується транслятором. Мітка може містити: літери латинської абетки, цифри (не може бути першим символом), спеціальні символи: ? @ \_ \$. Мітки в основному використовують для команд передачі керування і вони не є обов'язковими.

Мова Асемблера може містити змінні, які визначаються за допомогою директив: DB – визначає байт, DW – слово, DD – подвійне слово. Синтаксис:

ім'я DB значення

Вимоги до імені змінної такі ж, як і для мітки. Значення може бути: числовим (14, 4Bh, 1000112b), декілька чисел, строковим («рядок») та масивом, елементи якого розділяються комою, оператор «?» (який задає невизначену змінну) та DUP (дублювання даних вказану кількість раз). Змінна являє собою область пам'яті, яка помічена певним ім'ям. Зазвичай всі змінні розміщують в сегменті даних.

text_string	db	'Dobrogo dnya!'	; задавання рядку символів
number	dw	7	; number розміром 2 байти та значенням 7
tab	db	1,2,3,4,5,6,7,8,9	; визначення табличних даних (масиву)
null	db	?	; задавання невизначеної змінної
table_512	dw	512 dup(0)	; масив з 512 слів заповнених нулями.
mas	db	8 dup ('a', 'b')	; масив: 8 разів повторюється 'ab'

В якості операнда в команді може фігурувати константа, яка може вводитися в наступних формах:

- двійковою, як послідовність цифр 0 і 1, що закінчуються буквою B, наприклад, 10111010B;
- десятковою, в звичній десятковій системі числення з необов'язковою буквою D на кінці, наприклад, 129d або просто 129;
- шістнадцятиричною, як послідовність цифр від 0 до 9 і букв від A до F, що закінчується буквою H. Якщо шістнадцятирична константа розпочинається з букви, то така константа доповнюється першим символом - цифрою від 0 до 9, наприклад, 0E23h (в даному випадку перша цифра інформує Асемблер про те, що

E23 число, а не ідентифікатор або змінна);

- літералом, у вигляді рядка букв, цифр і інших символів, поміщеного в лапки або апострофи.

Мнемокоди можуть мати від 2 до 6 букв, при трансляції мнемокод перетвориться в числове значення по таблиці тієї, що перекодувала (усередині транслятора). Мнемокоди мають жорсткий формат, що передбачає 1,2 або відсутність операндів. Якщо операндів 2, вони відділяються один від одного коми.

Не можна використати в якості міток імена регістрів і мнемокоди, крім того мітка повинна розпочинатися з букви, але може містити цифри і спеціальні символи : ?, @, /, \_, \$ і точку, проте точка може бути тільки першим символом мітки.

Важливою особливістю машинних команд є те, що вони не можуть маніпулювати одночасно 2-мя операндами, що знаходяться в оперативній пам'яті (ОЗП). Це означає, що в команді тільки 1 операнд може вказувати на осередок ОЗУ, інший операнд має бути або регістром, або безпосереднім значенням. З цієї причини можливі наступні поєднання операндів в команді:

- регістр – регістр;
- регістр – пам'ять;
- пам'ять – регістр;
- регістр – безпосередній операнд;
- пам'ять – безпосередній операнд;
- сегментний регістр – регістр.

Для команд характерно, що за наявності двох операндів перший з них є приймачем, а другий – джерелом. Результат операції зберігається за першою адресою, ось чому перший операнд ніколи не може бути безпосереднім операндом або, інакше кажучи, константою.

### 1.3 Команда MOV– команда пересилки даних

Команда записується наступним чином:

MOV приймач, джерело

Дія: копіює вміст джерела в приймач, джерело не змінюється.

Приймач може бути регістром, пам'яттю. Джерело – регістром, пам'яттю, безпосереднім значенням.

MOV AX, 7 ; в регістр AX буде занесене число 7.

MOV AX, ABC ; в регістр AX буде занесене значення за адресою ABC.

MOV ABC, 82 ; за адресою ABC буде занесене число 82.

MOV DS, BX ; регістр DS матиме значення регістра BX

## 1.4 Стек

Стек – це структура пам'яті, яка використовується для тимчасового зберігання інформації. Програма може помістити дані в стек (PUSH) або забрати їх звідти (POP). Наприклад:

PUSH AX	; у стек розмістити вміст регістру AX
PUSH abc	; у стек розмістити значення змінної abc
PUSH 1234h	; у стек розмістити число 1234h
POP BX	; значення зі стеку розмістити у BX
POP [1234h]	; значення зі стеку розмістити за адресою 1234h
POP qwerty	; значення зі стеку розмістити у змінну qwerty

## 1.5 Програмний емулятор (віртуальний ПК) Emu8086

Для створення виконуємої програми на мові Асемблера необхідне певне програмне забезпечення: компілятор, компоновщик (редактор зв'язків) та налагоджувач (дебагер). Існує велика кількість пакетів подібного програмного забезпечення.

Програма пишеться у будь-якому текстовому редакторі та зберігається з довільним ім'ям та розширенням ASM. Такий файл називається вихідним модулем. За допомогою компілятора вихідний модуль перетворюється в об'єктний (файл з тим же ім'ям та розширенням OBJ), паралельно може створюватися і файл лістингу, який має розширення LST. Об'єктний модуль за допомогою редактора зв'язків перетворюється на виконуєму програму з розширенням COM або EXE.

Emu8086 поєднує в собі потужний редактор вихідного коду, асемблер, дізасемблер, програмний емулятор (віртуальний ПК) з відладчиком і поетапне навчання.

Візуальний інтерфейс дуже простий в роботі. Ви можете спостерігати регістри, прапори і пам'ять під час виконання вашої програми.

Арифметико-логічний пристрій (АЛП) показує внутрішню роботу центрального процесора (CPU).

Емулятор виконує програми на віртуальному ПК, який повністю виключає можливість доступу з вашої програми до реальних апаратних засобів, таким як жорсткі диски і пам'ять. Так як ваш код асемблера виконується на віртуальній машині, то налагодження стає більш легкою.

## 2 Завдання до лабораторної роботи

Розробити програму на Асемблері для виконання наступних завдань:

2.1 За адресою рівному дню і місяцю Вашого народження (наприклад, 23 травня – 2305) занести рік Вашого народження, представивши його як шістнадцятиричне число.

2.2 Визначити змінну const, присвоївши їй день і місяць Вашого народження у форматі описаному вище. Відправити це значення до стека. Розмістити це значення зі стека у регістр CX.

2.3 Призначити змінній fio Ваше прізвище, ім'я, по батькові. У пам'яті відразу після fio розмістити символ, код якого в ASCII-кодах визначити, як Ваш варіант плюс 14. А потім через пропуск дату Вашого народження у форматі ddmmuuuu (наприклад, 23051987).

2.4 Визначити адресу fio. Визначити адресу сегмента, зміщення та фізичну адресу кожного слова fio. Розмістити в регістр AL середню букву ПІБ (значення округлити до цілого). Дані внести в таблицю:

Слово в рядку	Адреса сегмента	Зміщення	Фізична адреса

У звіті до кожного завдання повинен бути скріншот середовища Emu8086, в якому повинні відображатись текст програми, регістри, стек та данні з оперативної пам'яті.

## 3 Хід роботи роботи

3.1 Ознайомитись з теоретичною частиною.

3.2 Написати та налагодити програму для завдання у середовищі Emu8086.

3.3 Записати очікувані результати.

### Приклад програми

```
; дата народження - 23/05/1987
org 100h
mov [2305h], 1987h
push const
pop cx
; мій варіант - 30
lea bx, fio
mov [bx+22], 44
mov [bx+23], ' '
```

```

mov [bx+24], 0523h
mov [bx+26], 8719h
; середня буква ПІВ має номер 11 і є «а»
mov al, [bx+11]
ret
const dw 2305d
fio db "Sidorov Ivan Petrovich"

```

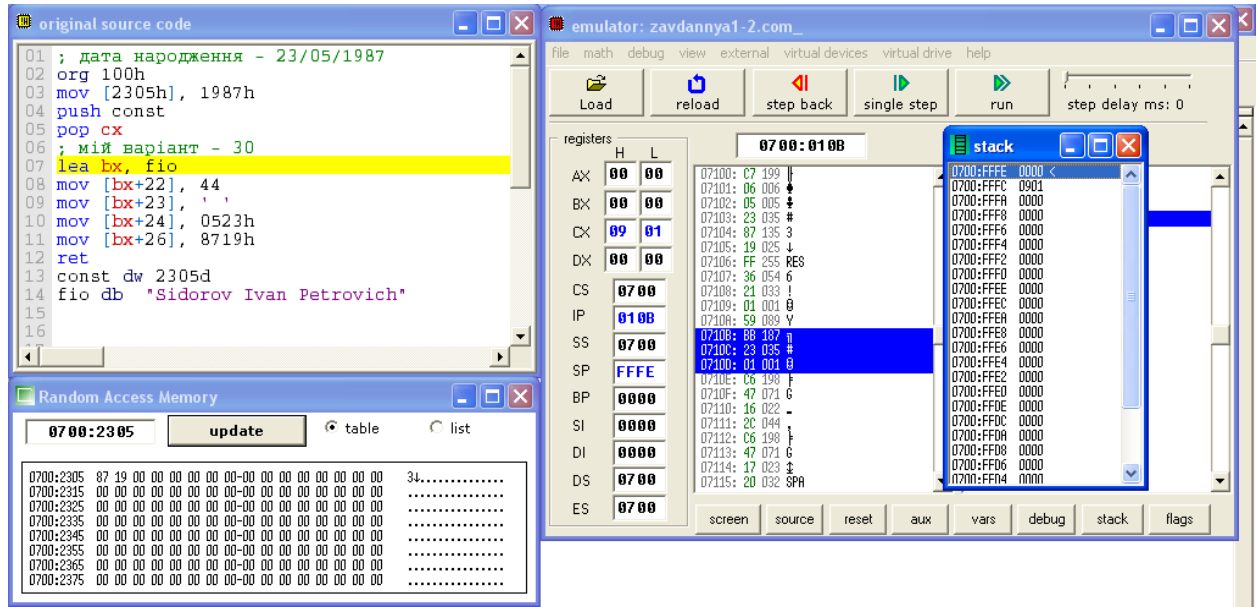


Рис.1. Скріншот виконання програми до пунктів 2.1-2.2

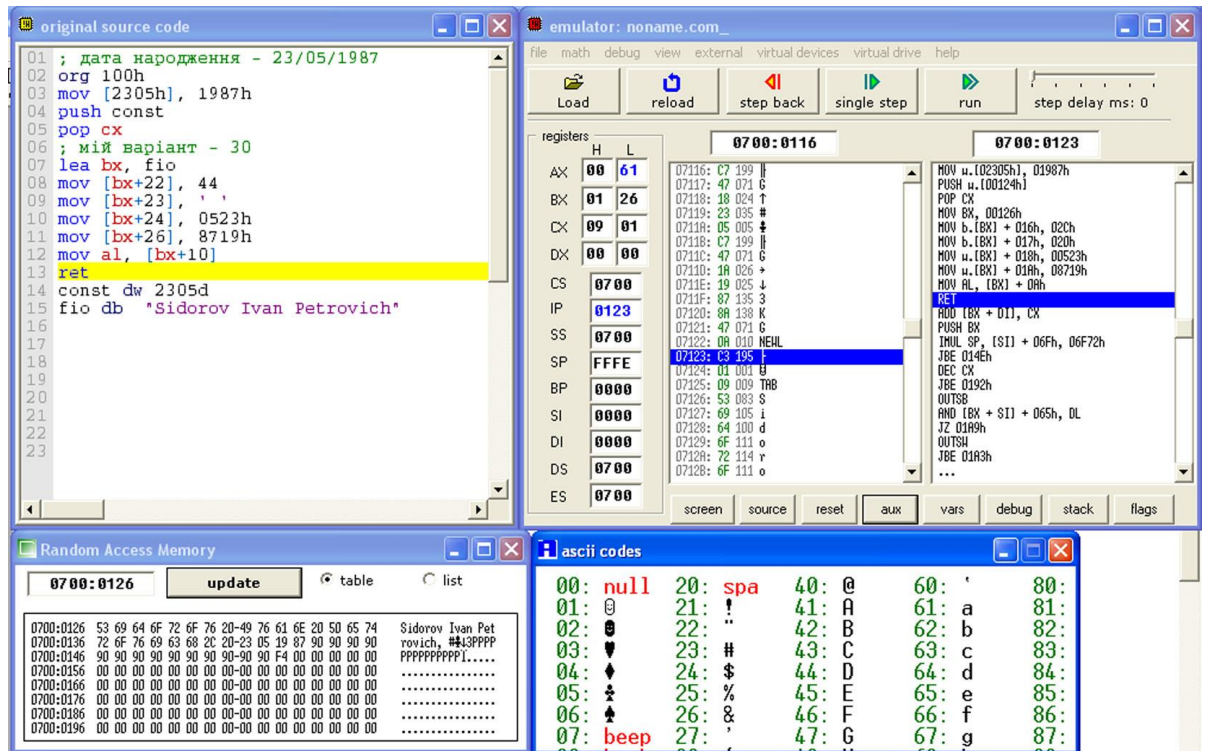


Рис. 2. Скріншот виконання програми до пунктів 2.3-2.5

Табл. 2. Виконання завдання до пункту 2.4

Слово в рядку	Адреса сегмента	Зміщення	Фізична адреса
Sidorov	0700	0126	07126
Ivan	0700	012E	0712E
Petrovich	0700	0133	07133

#### 4 Вміст звіту

Звіт повинен містити наступне:

- 4.1 Індивідуальне завдання.
- 4.2 Основний текст програми з коментарями до опису команд у завданні.
- 4.3 Заповнену таблицю.
- 4.4 Скріншоти програми, реєстрів, пам'яті.
- 4.5 Висновок по роботі.

#### 5 Контрольні запитання

1. Для чого використовують програмний емулятор Emu8086?
2. Роботу якого пристрою можна спостерігати при роботі в Emu8086?
3. Перелічить основні види реєстрів Intel8086.
4. Перелічить реєстри загального призначення Intel8086.
5. Перелічить індексні реєстри Intel8086.
6. Перелічить реєстри–вказівники Intel8086.
7. Перелічить сегментні реєстри Intel8086.
8. Для чого потрібен реєстр стану (реєстр прапорів)?
9. Вкажіть структуру команди Асемблера.
10. Які бувають директиви визначення змінних, масивів тощо?
11. Які можуть бути числові дані в Асемблері?
12. Яким чином визначають рядок?
13. Чим можуть буди операнди в команді? Перерахуйте можливі їх поєднання в команді.
14. Опишіть команду MOV.
15. Опишіть команди стека.