Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

# Лабораторна робота №1
## **Архітектура комп’ютера**

Виконали
студенти групи ІТ-01:

Гончаренко А. А
Чорній В. І

Перевірив:

Київ 2020

*Мета роботи:*

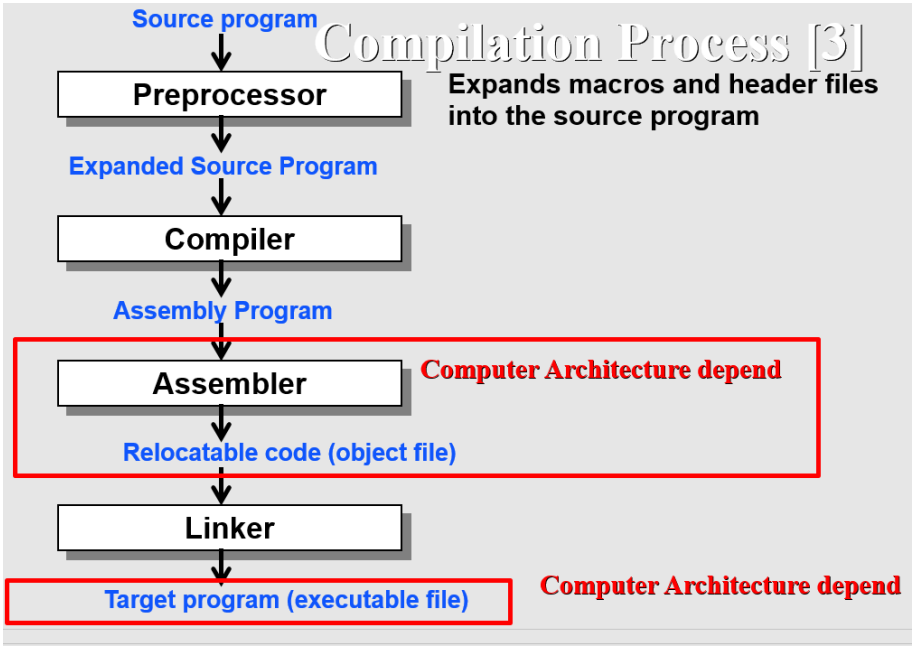
Набуття твердих навичок і знань технологічної основи розробки ПЗ на Асемблері, у ході якої застосовуються знання архітектури комп'ютерів.

*Хід роботи:*

1. Виконати повний технологічний цикл створення програми на Асемблері.

   Спочатку ознайомимось з повним технологічним циклом.

   Він виглядає так:




   Отже, спочатку нам потрібний вихідний код. Його ми візьмемо вже із завчасно підготовленого застосунку hw1.asm.

```
TITLE ЛР_1
;-------------------------------------------------------------------------------
;ЛР  №1.1
;-------------------------------------------------------------------------------
; Програмування 3. Системне програмування
; Завдання:              Основи розробки і налагодження
; ВУЗ:                  КНУУ "КПІ"
; Факультет:     ФІОТ
; Курс:                  1
; Група:         ІТ-01
;-------------------------------------------------------------------------------
; Автор:         Чорній В. Гончаренко А.
; Дата:          10/02/2021
;--------------------------------
                ;I.ЗАГОЛОВОК ПРОГРАМИ
IDEAL           ; Директива - тип Асемблера tasm
MODEL small     ; Директива - тип моделі пам'яті
STACK 256       ; Директива - розмір стеку
                ;II.МАКРОСИ
                ;III.ПОЧАТОК СЕГ arr_rnd1МЕНТУ ДАНИХ
DATASEG
exCode db 0
message db "Hello world!",10,13,'$';Рядок символів для виводу на екран
                ;VI. ПОЧАТОК СЕГМЕНТУ КОДУ
CODESEG
Start:
;--------------------------------- 1. Ініціалізація DS и ES-----------------------------------------
    mov ax,@data    ; @data ідентифікатор, що створюється директивою model
    mov ds, ax  ; Завантаження початку сегменту даних в регістр ds
    mov es, ax  ; Завантаження початку сегменту даних в регістр es
;---------------------------------2. Операція виводу на консоль---------------------------------
            ; Пересилання адреси рядка символів message в регістр dx
    mov dx, offset message
            ; Завантаження числа 09h до регістру ah
            ; (функція DOS 9h - команда виводу на консоль рядка)
    mov ah,09h
    int 21h     ; Виклик функції DOS 9h
;---------------------------------3. Операція зупинки програми, очікування натискання клавіш-----
    mov ah,01h
            ; Завантаження числа 01h до регістру ah
            ; (функція DOS 1h - команда очікування натискання клавіші...)
    int 21h     ; Виклик функції DOS 1h
            ; Завантаження числа 4ch до регістру ah
            ; (функція DOS 4ch - виходу з програми)
;---------------------------------4. Вихід з програми-----------------------------------------------
    mov ah,4ch
    mov al,[exCode] ; отримання коду виходу
    int 21h     ; виклик функції DOS 4ch
    end Start

DOS 4ch
    end Start
```

Наступний крок – асемблювання. Для виконання даного завдання ми повинні завантажити DOSBox, оскільки виконати це у звичайній консолі на Windows у нас не вийде.
Запускаємо DOSBox та вводимо наступні команди:

```
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount F D:/TASM
Drive F is mounted as local directory D:/TASM\
```

```
Z:\>F:

F:\>TASM HW1.ASM
Turbo Assembler  Version 3.2i  Copyright (c) 1988, 1992 Borland International
Serial No:   Tester:

Assembling file:   HW1.ASM
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  471k


F:\>_
```

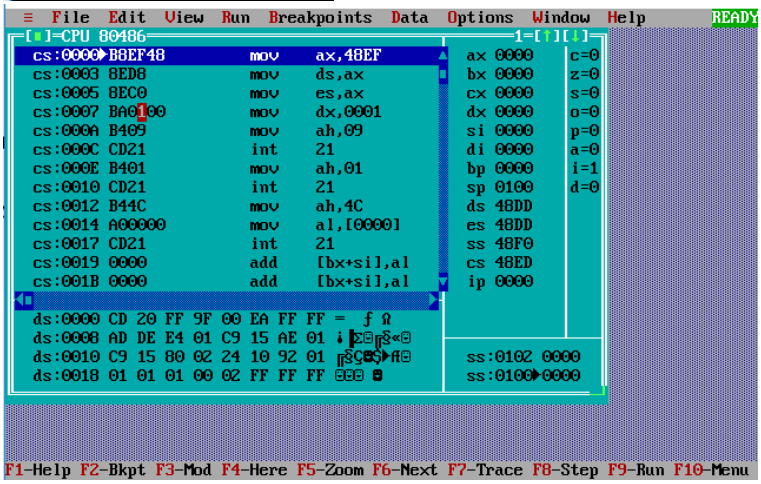
Команда mount прикріплює директорію з Асемблером до віртуального диску, на якому ми вже будемо працювати, а команда TASM власне виконує асемблювання.
Потім іде процес лінкування файлу hw1.obj командою tlink.

```
F:\>tlink hw1.obj
Turbo Link  Version 5.1 Copyright (c) 1992 Borland International
```

І тепер ми можемо або запустити дану програму, або запустити Turbo Debugger для налагодження програми.

```
F:\>hw1.exe
Hello world!
```

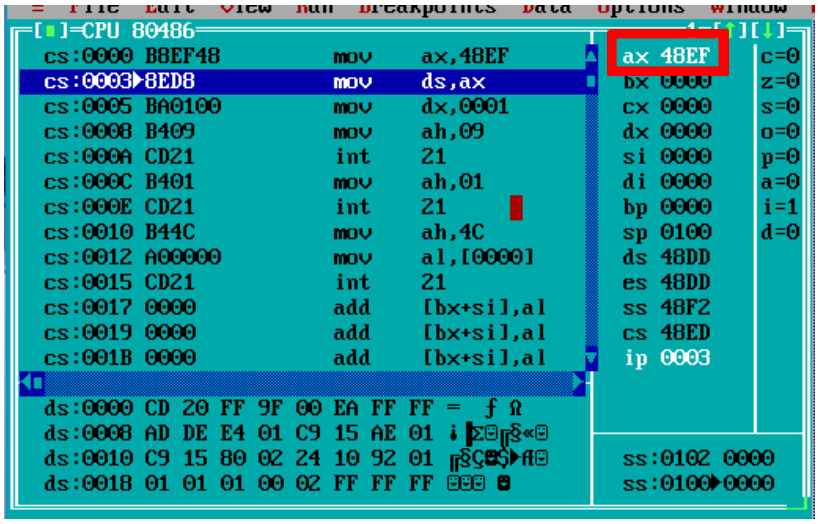
```
F:\>td hw1.exe_
```



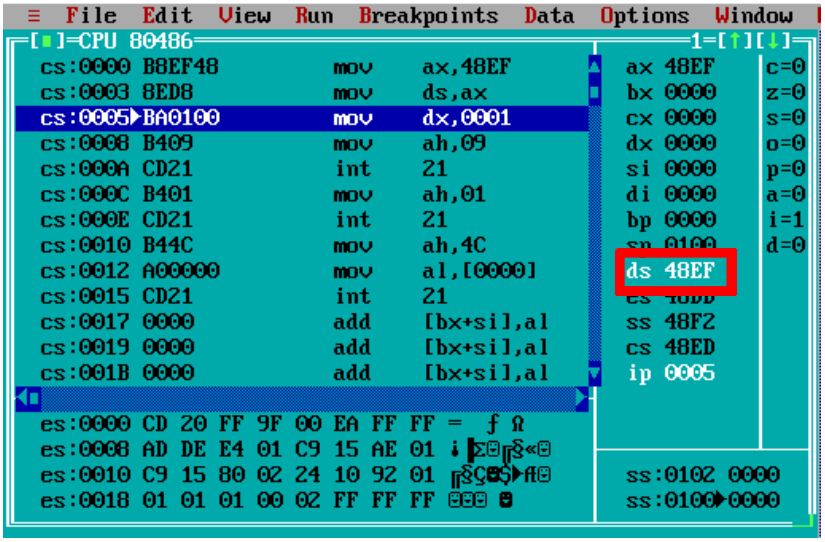

2. Доопрацювати вихідний код програми і вивести на консоль прізвища всіх студентів робочої бригади.
Відредагований код виглядає так:

```
TITLE laba1

IDEAL ;Обьявление типа асемблера - tasm
Model small ;Обьявление типа модели памяти
STACK 256 ;Обьявление размера стека

DATASEG
exCode db 0
messageHello db "Hello World!", 10, 13 ;Создание рядка символов для вывода
messageAndrey db "Honcharenko", 10, 13
messageVlad db "Chorniy", 10, 13, '$'

CODESEG
Start:
    mov ax, @data ;Инициализация ds
    mov ds, ax

    mov dx, offset messageHello ;Запись ссылки на первый рядок в регистр dx
    mov ah, 09h ;Запись функции вывода в консоль в регистр ah
    int 21h ;Вызов функции DOS 09h

    mov ah, 01h ;Запись числа операции 01h в регистр ah
    int 21h ;Вызов функции 01h для ожидания нажатия кнопки

    mov ah, 4ch ;Запись числа операции выхода из программы
    mov al, [exCode] ;Получение кода выхода
    int 21h ;Вызов функции 4ch
    end Start
```

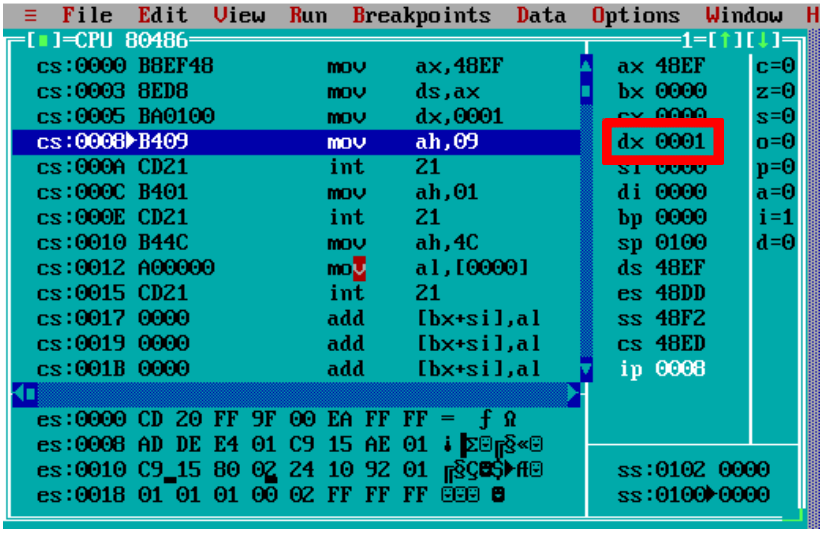
Для підтвердження результатів запустимо Turbo Debugger.



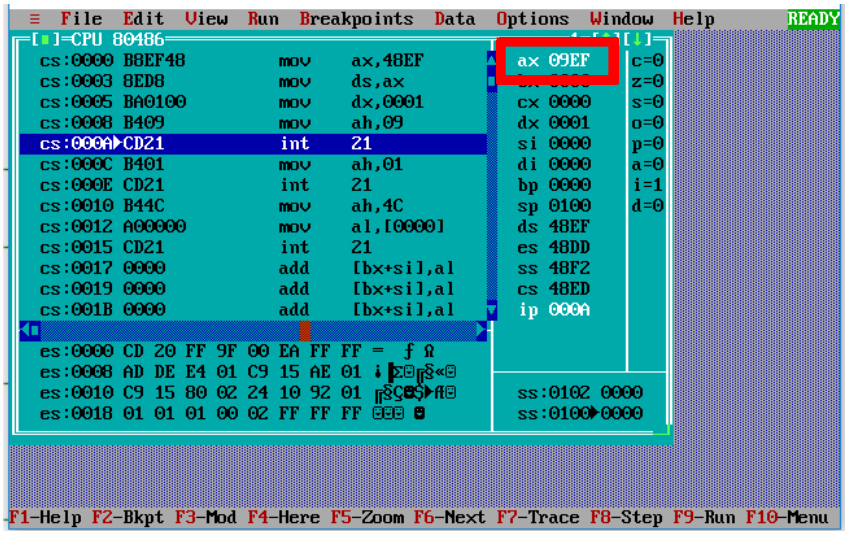

Запис в регістр ax посилання на дані.




Запис в регістр ds посилання на початок сегменту данних.

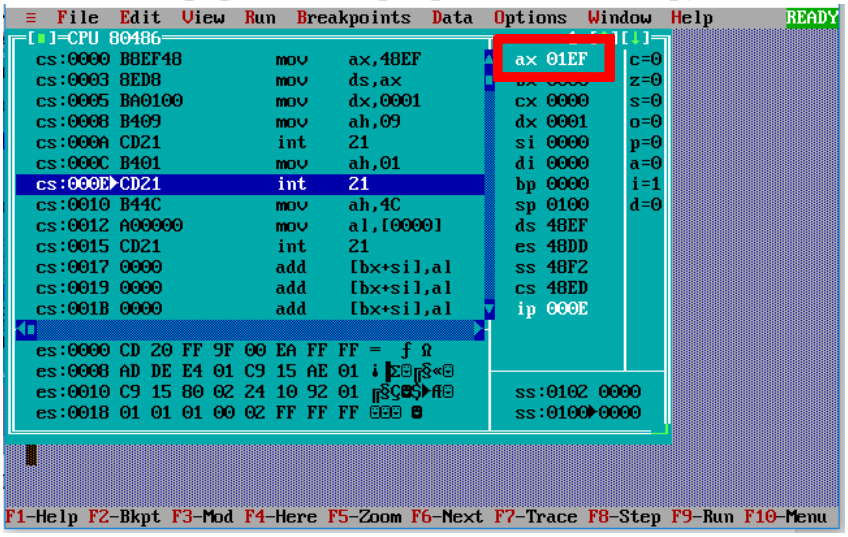



Записуємо в регістр dx зміщення змінної, у якій зберігаються дані. Цей регістр призначений для виводу даних в консоль.



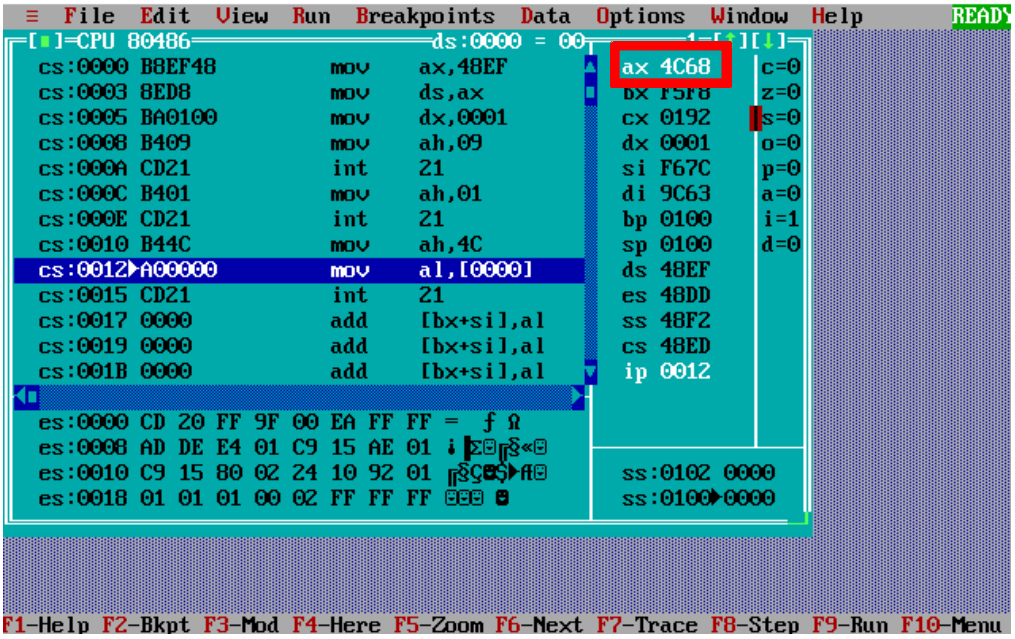

Записуємо в перші 2 байти регістру  ax число 09, яке пострібно для виведення даних в консоль.

Далі йде переривання програми і виклик функції DOS 09h.

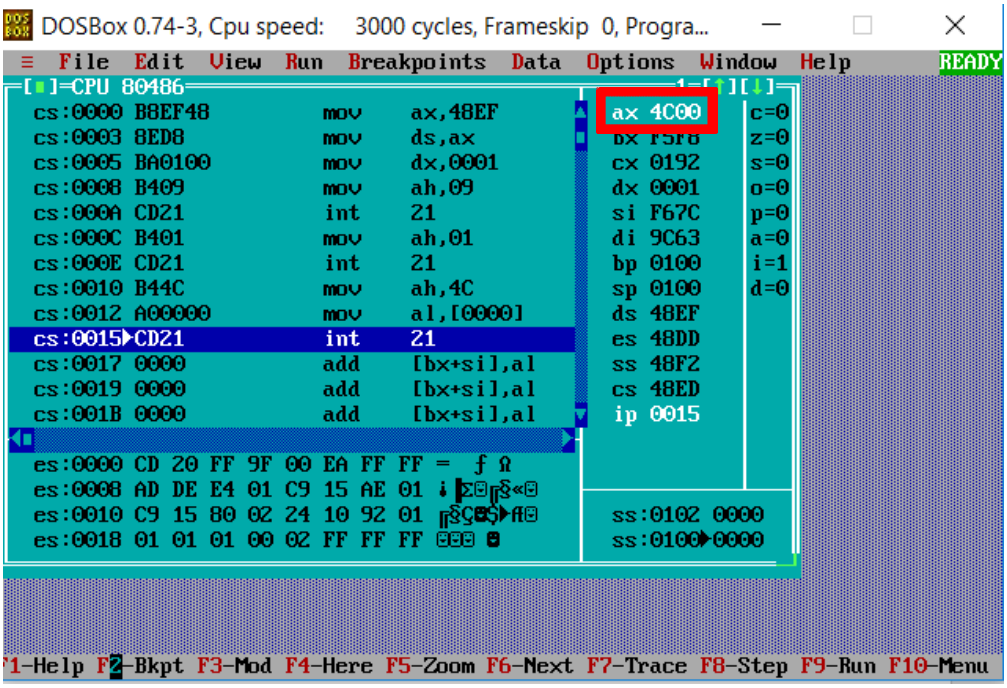



Записуємо у перших 2 байта регістру ax число 01, що слугує функцією очікування будь-якої клавіші.


Далі програма переривається і викликається функція DOS 01.




Записуємо у перших 2 байта регістру ax число 4C, що слугує функцією завершення програми.




Записуємо у останні 2 байта регістру ax число 00 (exCode), що обнуляє результуючий код програми.

Далі вдбувається переривання програми, виклик функції DOS 4c і в результаті її завершення.

3. Опис архітектурних елементів x86, що задіяні в програмі.
   Асемблер працює з Real Address Mode. У ньому пам'ять поділена на сегменти (загальний розмір – 1MB), і доступ до деякої комірки пам’яті здійснюється за допомогою адреси початку сегменту та зміщення у ньому. Всього сегментів ми використали 3: ES – додатковий сегмент, CS – сегмент коду та DS – сегмент даних.

Висновок:

В ході лабораторної було набуто твердих навичок і знань технологічної основи розробки ПЗ на Асемблері. І використано такі знання з архітектури комп’ютера, як знання сегментації даний у Real mode та правил звернення до довільної комірки пам’яті у цьому режимі процесора