Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

Лабораторна робота №5

з дисципліни «Архітектура комп'ютера»

Тема: «Управління ходом виконання програми на асемблері архітектури IA-32 (X86) у REAL ADRESS MODE»

Варіант-3

Виконали: студенти групи ІТ-01

Гончаренко А.А.
Чорній В.І.
Дмитрієва І.І.

Викладач: Бердник Ю.М.
Захищено з балом_____
Дата здачі: 22.04.2021

Київ-2020

Мета:

набути впевнених знань і навичок з розробки ПЗ на Асемблері, вивчити механізми управління потоком виконання у архітектурі IA-32 у real address mode.

Завдання:

1) Створити двовимірний масив array2Db, що складається з елементів в один байт, має розмір 16x16
2) Відсортувати масив
3) Записати всередину масиву квадрат 8x8, що заповнений датами народження студентів з групи

Хід роботи:

- скористатися результатами лабораторної роботи 4.
- Вивчити додаткові команди Асемблеру, що дають можливість керувати ходом програми.
- Вивчити додаткові переривання DOS і BIOS для відображення інформації на консолі.
- Закріпити вивчений матеріал шляхом виконання маніпуляцій з масивом, відповідно до свого варіанту
- Зберегти програму та зробити висновки щодо необхідних знань для виконання даної лабораторної роботи.

Для вирішення завдання був написаний код на Асемблері.
Посилання на GitHub:

- Гончаренко Андрій
- Дмитрієва Ірина
- Чорній Владислав

Реалізація лабораторної роботи:

Створюємо масив чисел 16х16, заповнений випадковими числами та 3 дати народження.

```
DATASEG

;Оголошення двовимірного експериментального масиву 16x16
array2Db        DW  '3', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '3', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '3', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '3', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '2', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '2', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '2', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '2', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '2', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '2', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '2', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                Dw  '2', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '2', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '3', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '3', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6'
                DW  '3', '4', '8', '9', '7', '3', '0', '3', '9', '0', '7', '6', '1', '0', '8', '6',13, 10,'$'

exCode   DB 0
Vladbrth dw '1','4','0','2','2','0','0','3'
Andrybrth dw '1','8','1','1','2','0','0','2'
Irabrth  dw '0','5','0','7','2','0','0','3'
```

Також для зручності створюємо макрос запису даних в регістр даних.

```
;-----------------------II.МАКРОСИ-------------------------------------
; Складний макрос для ініціалізації
MACRO M_Init        ; Початок макросу
    mov ax,@data     ; @data ідентифікатор, що створюються директивою model
    mov ds, ax   ; Завантаження початку сегменту даних в регістр ds
    mov es, ax   ; Завантаження початку сегменту даних в регістр es
ENDM M_Init
```

Сегмент коду має наступний вигляд

```
;----------------------VI. ПОЧАТОК СЕГМЕНТУ КОДУ------------------
Start:
M_Init
call sort  ; сортуємо масив
xor si,si
lea ax,[array2Db]    ; записуємо в регістр початок масиву
mov dx,ax
mov ah, 09h
int 21h              ; виводисо відсортований масив
mov ah,01h
int 21h              ; чекаємо на введення клавіші
lea si,[array2Db]
add si,66h               ; переміщуємось на координати (3,3)
mov cx,2
add_to_array:             ; вставляємо прямокутник з датами 8х6
    lea di,[Vladbrth]    ; записуємо в регістр початок сегменту памяті дат
    push cx              ; запамятовуємо кількість ітерацій
    call add_brth         ; записуємо 1 дату
    add si,10h               ; переміщуємось на рядок
    call add_brth            ; записуємо 2 дату
    add si,10h
    call add_brth            ; записуємо 3 дату
    pop cx                   ; повертаємо в регістр кількість ітерцій, що залишилися
    add si,10h
loop add_to_array


lea di,[Vladbrth]         ; додаємо до прямокутника 8х6 ще 2 дати
call add_brth
add si,10h
call add_brth

mov ah, 09h               ; виводимо масив з датами
int 21h
mov ah,01h
int 21h

mov ah,4ch                ;завершення програми
mov al,[exCode]
int 21h
```

У цьому сегменті ми викликаємо процедуру sort, яка сортує масив за допомогою Bubble sort. Далі викликаються функції виводу на екран та очікування клавіші, щоб можна було побачити результат. Наступний крок це запис масиву 8 на 8 у вже відсортований масив. Внутрішній масив заповнюється датами з сегменту даних за допомогою процедур add_brth. Тут знову можна побачити виклик функцій переривання програми, а саме вивід масиву та очікування на ввід клавіші. В кінці реалізовано вихід з програми.
Процедура сорт виглядає так:

```
;--------------------Процедури----------------------------------------
proc sort
mov bp,256  ; записуємо кількість ітерацій зовнішнього циклу
C:
    mov cx, 256 ; записуємо кількість ітерацій внутрішнього циклу
    dec cx      ; зменшуємо на одиницю кількість ітерцій внутрішнього циклу
    lea bx, [array2Db] ; записуємо у регістр початок сегменту памяті, де зберігається масив
    mov si, 510             ; переходимо в кінець масиву
B:
    mov ax, [bx + si]  ; записуємо в регістр останній елемент масиву
    mov di, si
    A:
        sub di, 2
        mov dx, [bx + di]        ; переходимом до наступного елементу(рахуючи з кінця)

        cmp ax, dx               ; порівняння цих 2 елементів
        jng next                  ; якщо перший елемент не більший, пропускаємо наступні 2 кроки
            mov [bx+di], ax      ; міняємо місцями 2 елементи
            mov [bx+si], dx
     next:
     sub si, 2                   ; переходимо да наступного елементу
     loop B
        xor si, si               ; анулюємо індексні регістри щоб прейти до наступного кроку
         xor di, di
         dec bp                  ; позначаємо, що ітерацію завершено
         cmp bp,0                 ; порівнюємо кількість операцій з 0
         jne c                    ; поки кількість ітерацій не буде рівним нулю, повторюєм цикл


ret
endp sort
```

Процедура add_brth реалізована наступним чином:

```
proc add_brth
mov cx,8                ; задаємо кількість ітерацій
do_it:
mov ax,[ds:di]              ; записуємо в регістр цифру з дати
mov [ds:si],ax              ; записуємо цю цифру в масив
add si,2                    ; переходимо до наступної цифри
add di,2
loop do_it
ret
endp add_brth
```

Для асемблювання, лінкування, запуску програми та турбодебагера, створено бат-файл.

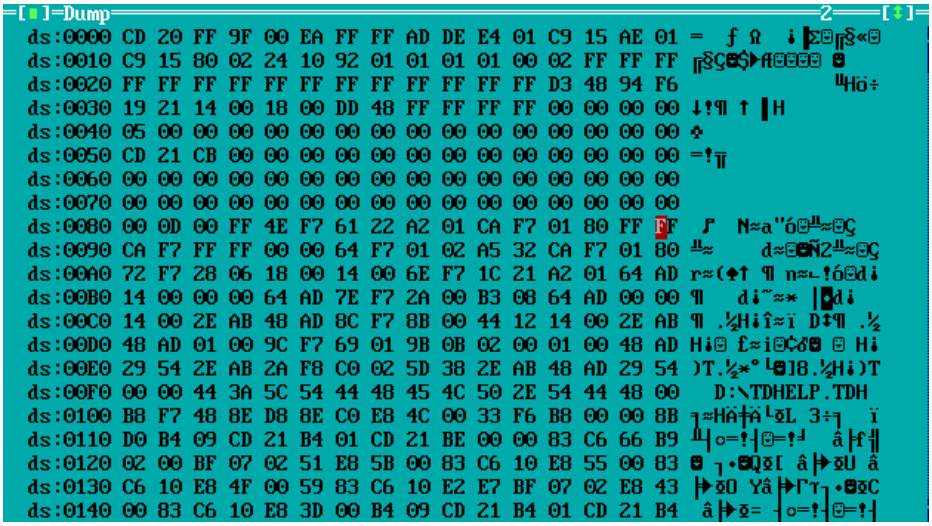
```
Файл  Правка  Формат  Вид  Справка
@ECHO OFF
REM 1 Assembling
CLS
TASM LAB5.asm
IF ERRORLEVEL 1 GOTO exit
REM PAUSE

REM 1 Linking
TLINK LAB5.obj
IF ERRORLEVEL 1 GOTO exit
REM PAUSE
LAB5.EXE
REM PAUSE
TD LAB5.EXE
:exit
ECHO ON
```

Результат програми в консолі:

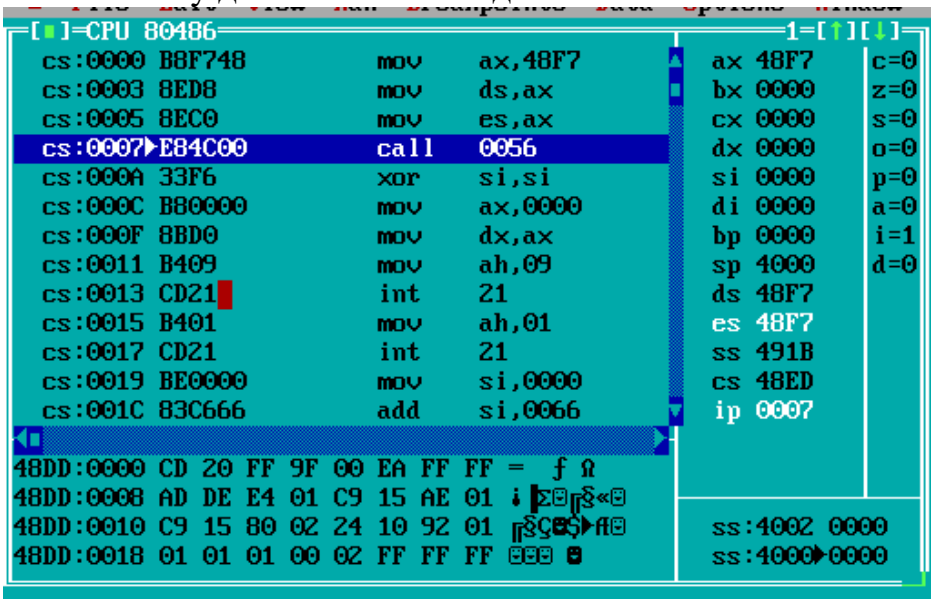
```
Serial No:   Tester:

Assembling file:   LAB5.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  468k

Turbo Link  Version 5.1 Copyright (c) 1992 Borland International
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 7 7 7 7 7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 1 4 0 2 2 0 0 3 8 8 8 8 8 7 7 7 1 8 1 1 2 0 0 2 7 7 7 7 7
7 7 7 0 5 0 7 2 0 0 3 7 7 7 7 7 6 6 6 1 4 0 2 2 0 0 3 6 6 6 6 6 6 6 6 1 8 1 1 2
0 0 2 6 6 6 6 6 4 4 4 0 5 0 7 2 0 0 3 4 4 4 4 4 3 3 3 1 4 0 2 2 0 0 3 3 3 3 3 3
3 3 3 1 8 1 1 2 0 0 2 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

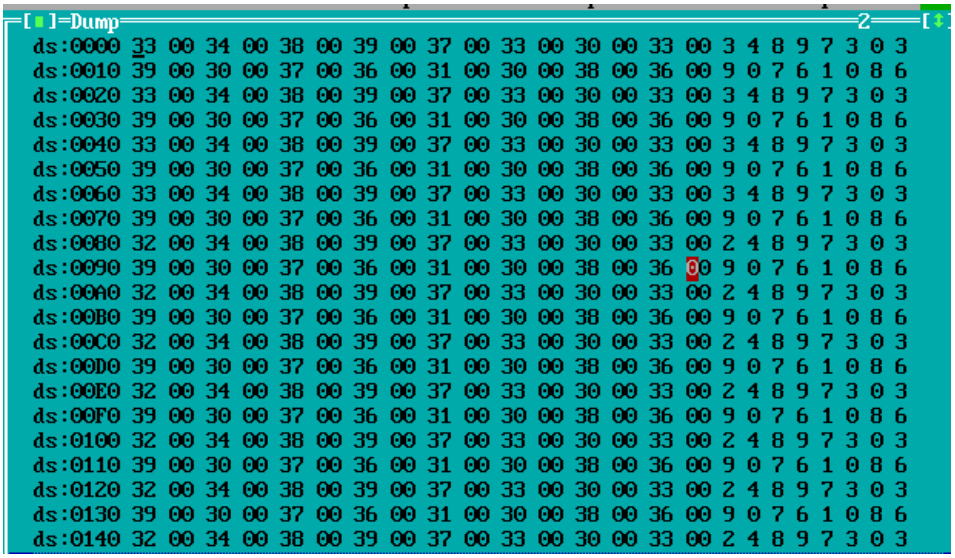
Відслідкуємо виконання програми у турбодебагері:

При запуску дамп виглядає наступним чином.

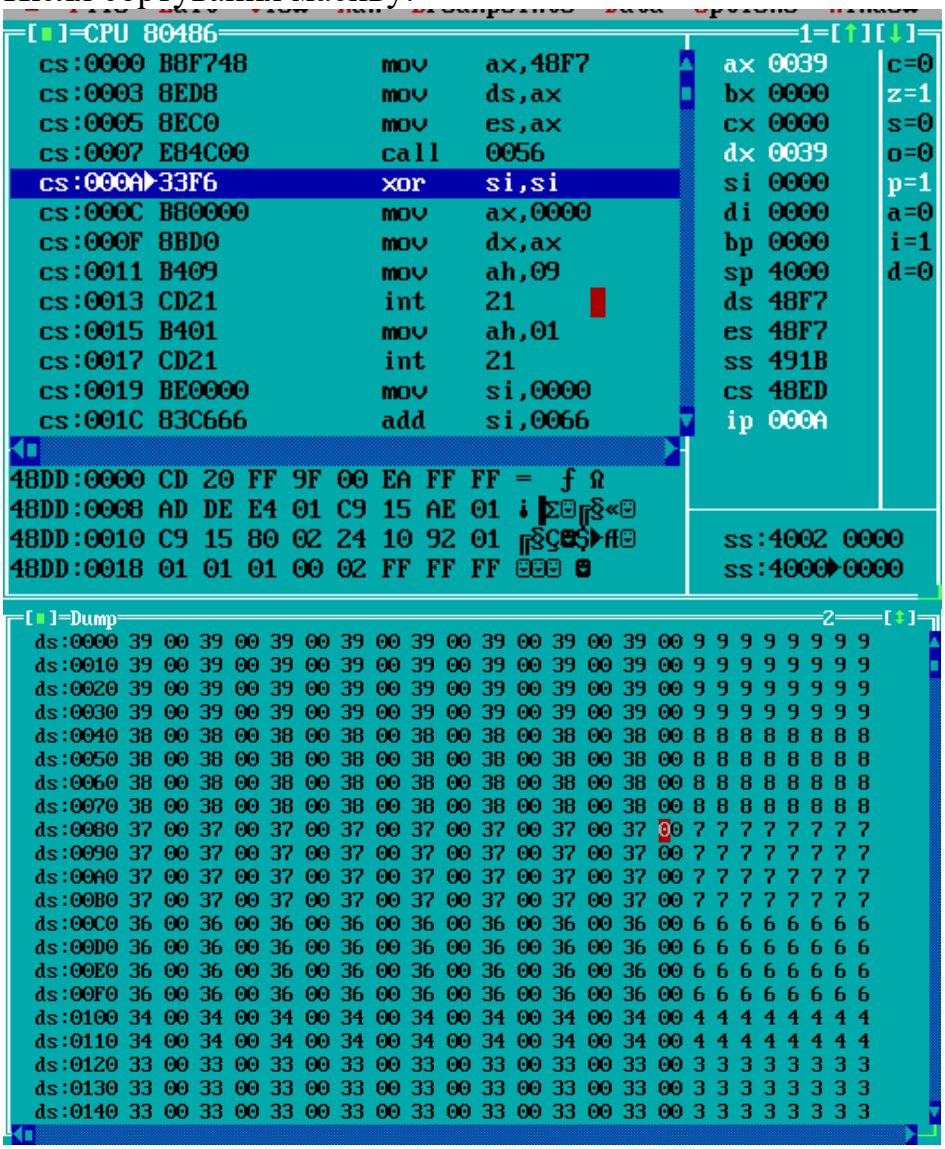


Після запису даних в сегмент даних:

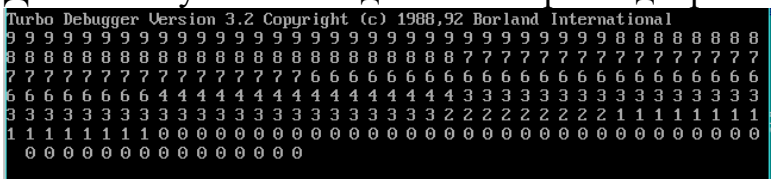




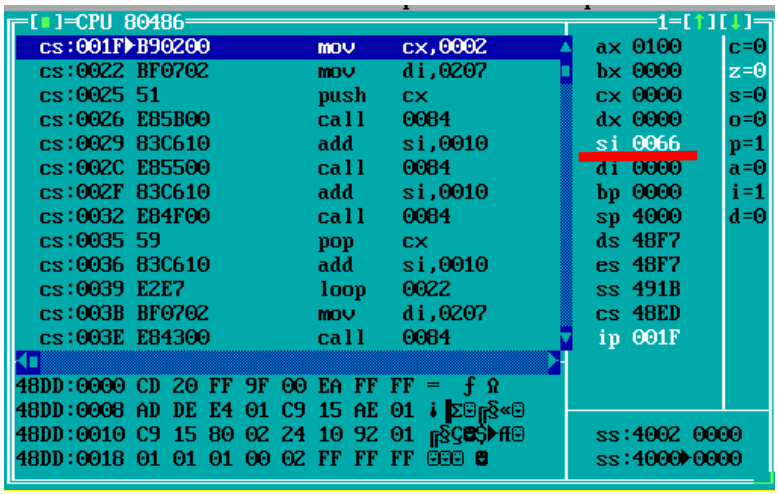
Після сортування масиву:



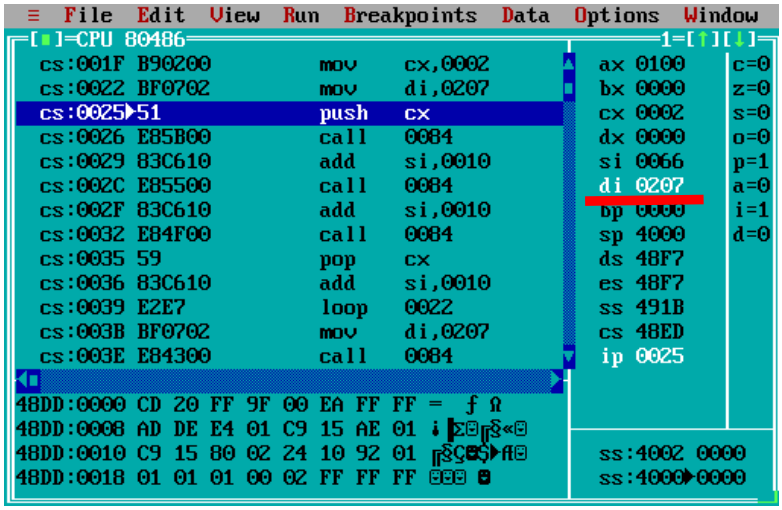
Далі виконується виведення на екран відсортованого масиву:



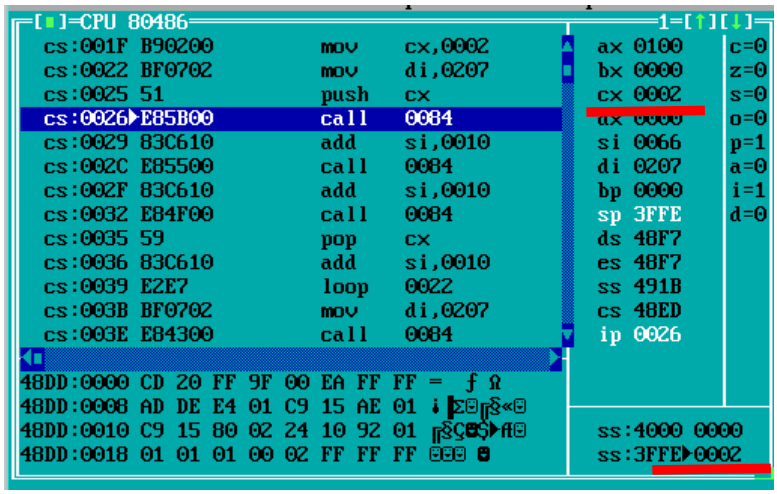
Далі переходимо відповідно до варіанту до координат (3,3) в масиві 16x16.



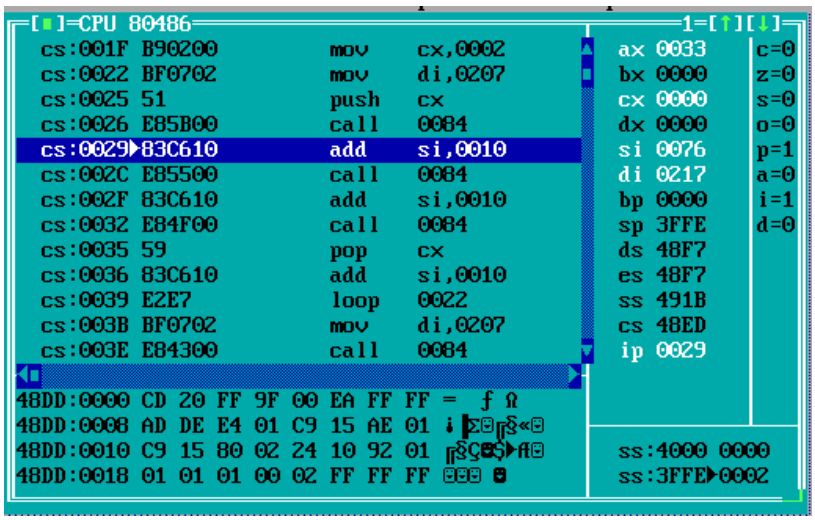
Запис в регістр di початок сегменту пам’яті, де зберігаються дати.

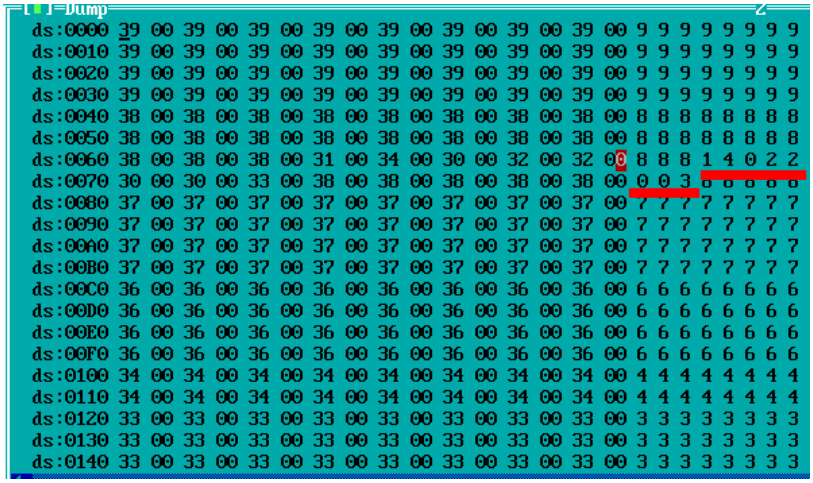


Записуємо в стек значення регістру cx:

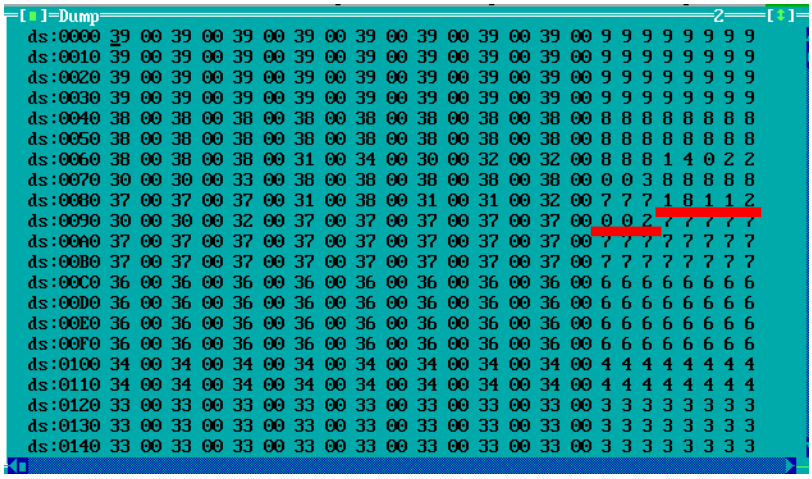


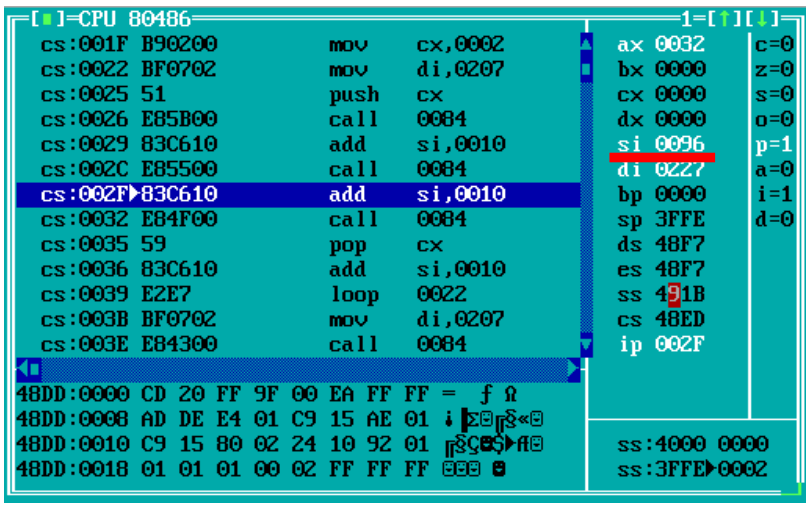
Виклик процедури запису в масив.



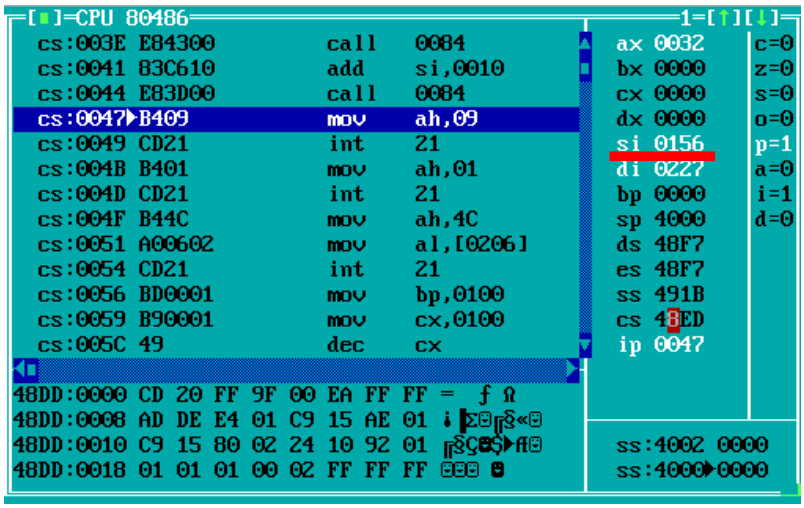


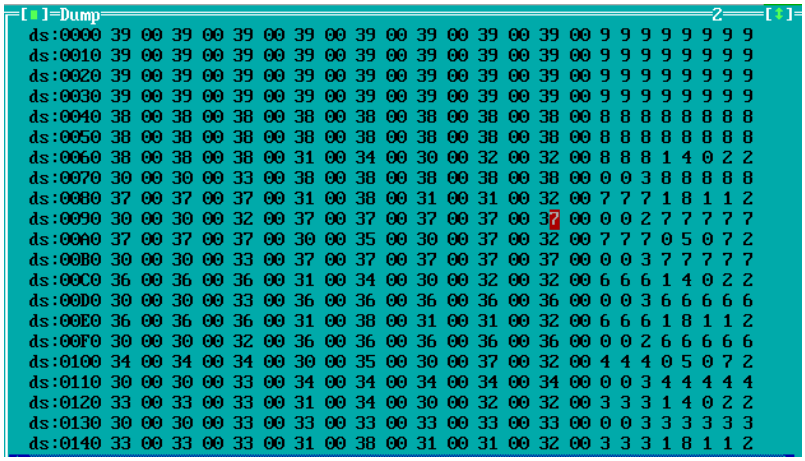
Зміщуємось на рядок і знову викликаємо функцію запису.



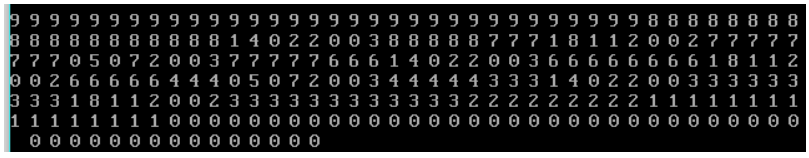


Після повного запису масиву 8x8

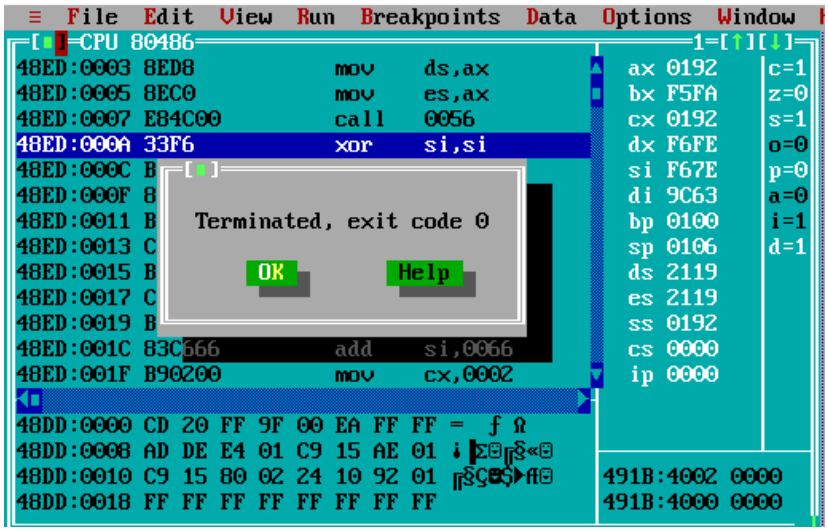




Далі виконуються функції виводу на екран та запит на введення клавіші.



І нарешті завершення програми.



Висновки:

В ході лабораторної було набуто твердих навичок і знань технологічної основи розробки ПЗ на Асемблері, закріплено знання механізмів адресації, навичок щодо управління потоком виконання та вивчено функції виводу даних у консоль в архітектурі IA-32 у real address mode.