

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 архітектурі IA-32  
(X86) у REAL ADDRESS MODE»

Варіант-3

Виконали: студенти групи IT-01

Гончаренко А.А.

Чорній В.І.

Дмитрієва І.І.

Викладач: Бердник Ю.М.

Захищено з балом \_\_\_\_\_

Дата здачі: 14.04.2021

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 16x16, 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 '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 ; виводимо 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 прямокутних з дати 8x6
    lea di,[Vladbrth] ; записуємо в регістр початок сегменту пам'яті дат
    push cx ; запам'ятовуємо 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 ітерацій, що записалися
    add si,10h
loop add_to_array

lea di,[Vladbrth] ; додаємо до прямокутника 8x6 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 ітерацій внутрішнього циклу
    lea bx, [array2Db] ; записуємо у регістр початок сегменту пам'яті, 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] ; переходим 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 до наступного елементу
    loop B
    xor si, si ; анулюємо індексні регістри щоб перейти до наступного кроку
    xor di, di
    dec bp ; позначаємо, що ітерація завершена
    cmp bp,0 ; порівнюємо кількість операцій з 0
    jne c ; поки кількість ітерацій не буде рівня 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, лінування, 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

```

Serial No:    Tester:

Serial No:    Tester:

При запуску дампу виглядає наступним чином.

При запуску дампу виглядає наступним чином.

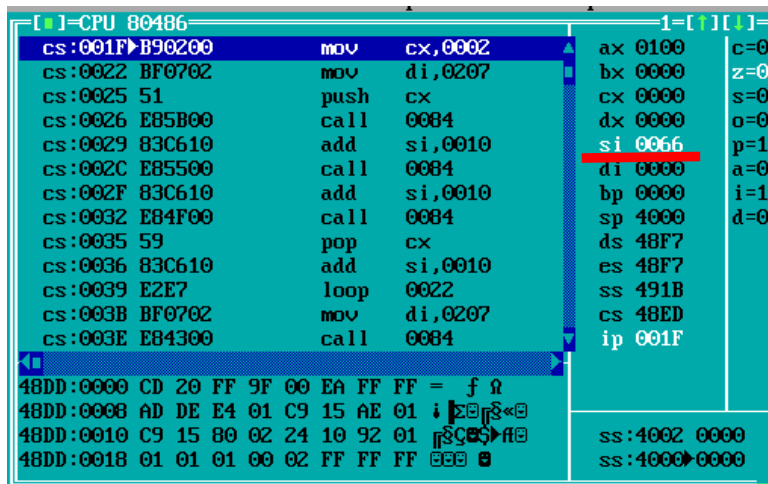
[illegible]

Після запису даних в сегмент даних:

[ ] - CPU 80486		1 = [ ] [ ]	
cs:0000 B8F748	mov ax,48F7	ax 48F7	c=0
cs:0003 8ED8	mov ds,ax	bx 0000	z=0
cs:0005 8EC0	mov es,ax	cx 0000	s=0
cs:0007 E84C00	call 0056	dx 0000	o=0
cs:000A 33F6	xor si,si	si 0000	p=0
cs:000C B80000	mov ax,0000	di 0000	a=0
cs:000F 8BD0	mov dx,ax	bp 0000	i=1
cs:0011 B409	mov ah,09	sp 4000	d=0
cs:0013 CD21	int 21	ds 48F7	
cs:0015 B401	mov ah,01	es 48F7	
cs:0017 CD21	int 21	ss 491B	
cs:0019 BE0000	mov si,0000	cs 48ED	
cs:001C 83C666	add si,0066	ip 0007	
<div> <div>48DD:0000 CD 20 FF 9F 00 EA FF FF = f 0</div> <div>48DD:0008 AD DE E4 01 C9 15 AE 01 ; 00 00 00</div> <div>48DD:0010 C9 15 80 02 24 10 92 01 ; 00 00 00</div> <div>48DD:0018 01 01 01 00 02 FF FF FF 00 00</div> </div>			
		ss:4002 0000	
		ss:4000 0000	

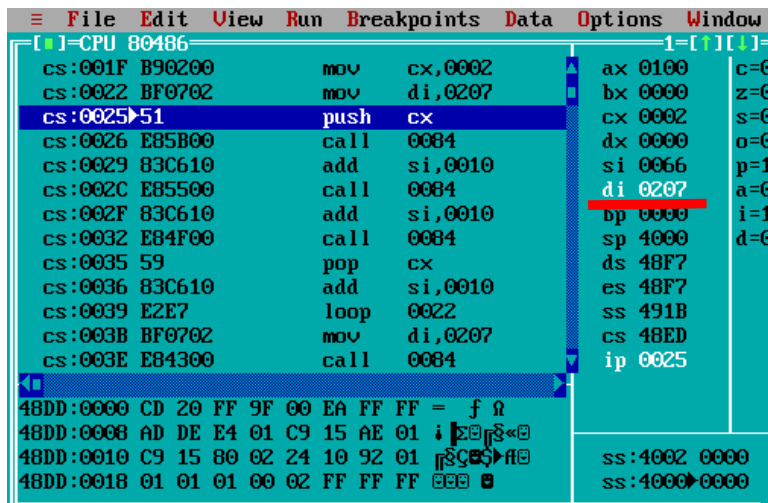


Далі переходимо відповідно до варіанту до координат (3,3) в масиві 16х16.



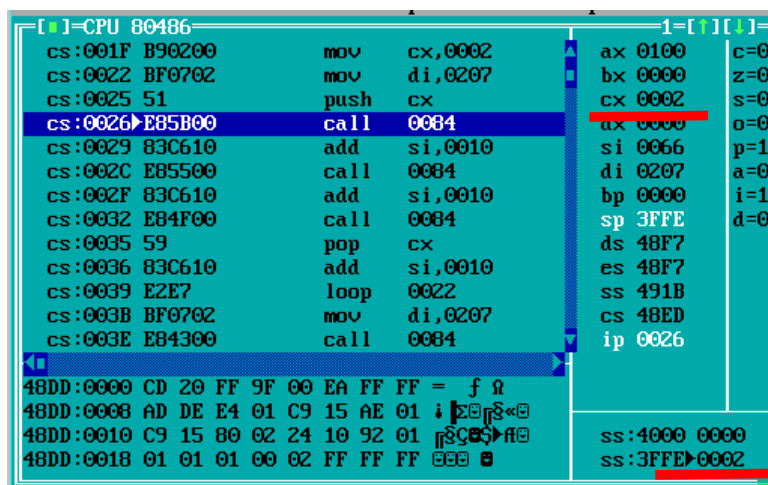
CPU 80486		1=[ ]	
cs:001F B90200	mov cx,0002	ax 0100	c=0
cs:0022 BF0702	mov di,0207	bx 0000	z=0
cs:0025 51	push cx	cx 0000	s=0
cs:0026 E85B00	call 0084	dx 0000	o=0
cs:0029 83C610	add si,0010	si 0066	p=1
cs:002C E85500	call 0084	di 0000	a=0
cs:002F 83C610	add si,0010	bp 0000	i=1
cs:0032 E84F00	call 0084	sp 4000	d=0
cs:0035 59	pop cx	ds 48F7	
cs:0036 83C610	add si,0010	es 48F7	
cs:0039 E2E7	loop 0022	ss 491B	
cs:003B BF0702	mov di,0207	cs 48ED	
cs:003E E84300	call 0084	ip 001F	
48DD:0000 CD 20 FF 9F 00 EA FF FF = f 0		ss:4002 0000	
48DD:0008 AD DE E4 01 C9 15 AE 01 i 20 00 00		ss:4000 0000	
48DD:0010 C9 15 80 02 24 10 92 01 00 00 00 00			
48DD:0018 01 01 01 00 02 FF FF FF 00 00 00			

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



CPU 80486		1=[ ]	
cs:001F B90200	mov cx,0002	ax 0100	c=0
cs:0022 BF0702	mov di,0207	bx 0000	z=0
cs:0025 51	push cx	cx 0002	s=0
cs:0026 E85B00	call 0084	dx 0000	o=0
cs:0029 83C610	add si,0010	si 0066	p=1
cs:002C E85500	call 0084	di 0207	a=0
cs:002F 83C610	add si,0010	bp 0000	i=1
cs:0032 E84F00	call 0084	sp 4000	d=0
cs:0035 59	pop cx	ds 48F7	
cs:0036 83C610	add si,0010	es 48F7	
cs:0039 E2E7	loop 0022	ss 491B	
cs:003B BF0702	mov di,0207	cs 48ED	
cs:003E E84300	call 0084	ip 0025	
48DD:0000 CD 20 FF 9F 00 EA FF FF = f 0		ss:4002 0000	
48DD:0008 AD DE E4 01 C9 15 AE 01 i 20 00 00		ss:4000 0000	
48DD:0010 C9 15 80 02 24 10 92 01 00 00 00 00			
48DD:0018 01 01 01 00 02 FF FF FF 00 00 00			

Записуємо в стек значення регістру cx:



CPU 80486		1=[ ]	
cs:001F B90200	mov cx,0002	ax 0100	c=0
cs:0022 BF0702	mov di,0207	bx 0000	z=0
cs:0025 51	push cx	cx 0002	s=0
cs:0026 E85B00	call 0084	dx 0000	o=0
cs:0029 83C610	add si,0010	si 0066	p=1
cs:002C E85500	call 0084	di 0207	a=0
cs:002F 83C610	add si,0010	bp 0000	i=1
cs:0032 E84F00	call 0084	sp 3FFE	d=0
cs:0035 59	pop cx	ds 48F7	
cs:0036 83C610	add si,0010	es 48F7	
cs:0039 E2E7	loop 0022	ss 491B	
cs:003B BF0702	mov di,0207	cs 48ED	
cs:003E E84300	call 0084	ip 0026	
48DD:0000 CD 20 FF 9F 00 EA FF FF = f 0		ss:4000 0000	
48DD:0008 AD DE E4 01 C9 15 AE 01 i 20 00 00		ss:3FFE 0002	
48DD:0010 C9 15 80 02 24 10 92 01 00 00 00 00			
48DD:0018 01 01 01 00 02 FF FF FF 00 00 00			

Виклик процедури запису в масив.

CPU 80486

cs:001F B90200 mov cx,0002  
cs:0022 BF0702 mov di,0207  
cs:0025 51 push cx  
cs:0026 E85B00 call 0084  
cs:0029 B3C610 add si,0010  
cs:002C E85500 call 0084  
cs:002F B3C610 add si,0010  
cs:0032 E84F00 call 0084  
cs:0035 59 pop cx  
cs:0036 B3C610 add si,0010  
cs:0039 E2E7 loop 0022  
cs:003B BF0702 mov di,0207  
cs:003E E84300 call 0084

ax 0033  
bx 0000  
cx 0000  
dx 0000  
si 0076  
di 0217  
bp 0000  
sp 3FFE  
ds 48F7  
es 48F7  
ss 491B  
cs 48ED  
ip 0029

c=0  
z=0  
s=0  
o=0  
p=1  
a=0  
i=1  
d=0

48DD:0000 CD 20 FF 9F 00 EA FF FF = f 0  
48DD:0008 AD DE E4 01 C9 15 AE 01 i 2 0 0 0  
48DD:0010 C9 15 80 02 24 10 92 01 0 0 0 0  
48DD:0018 01 01 01 00 02 FF FF FF 0 0 0 0

ss:4000 0000  
ss:3FFE 0002

Dump

ds:0000 39 00 39 00 39 00 39 00 39 00 39 00 39 00 39 00 9 9 9 9 9 9 9 9  
ds:0010 39 00 39 00 39 00 39 00 39 00 39 00 39 00 39 00 9 9 9 9 9 9 9 9  
ds:0020 39 00 39 00 39 00 39 00 39 00 39 00 39 00 39 00 9 9 9 9 9 9 9 9  
ds:0030 39 00 39 00 39 00 39 00 39 00 39 00 39 00 39 00 9 9 9 9 9 9 9 9  
ds:0040 38 00 38 00 38 00 38 00 38 00 38 00 38 00 38 00 8 8 8 8 8 8 8 8  
ds:0050 38 00 38 00 38 00 38 00 38 00 38 00 38 00 38 00 8 8 8 8 8 8 8 8  
ds:0060 38 00 38 00 38 00 31 00 34 00 30 00 32 00 32 00 8 8 8 1 4 0 2 2  
ds:0070 30 00 30 00 33 00 38 00 38 00 38 00 38 00 38 00 0 0 3 8 8 8 8 8  
ds:0080 37 00 37 00 37 00 37 00 37 00 37 00 37 00 37 00 7 7 7 7 7 7 7 7  
ds:0090 37 00 37 00 37 00 37 00 37 00 37 00 37 00 37 00 7 7 7 7 7 7 7 7  
ds:00A0 37 00 37 00 37 00 37 00 37 00 37 00 37 00 37 00 7 7 7 7 7 7 7 7  
ds:00B0 37 00 37 00 37 00 37 00 37 00 37 00 37 00 37 00 7 7 7 7 7 7 7 7  
ds:00C0 36 00 36 00 36 00 36 00 36 00 36 00 36 00 36 00 6 6 6 6 6 6 6 6  
ds:00D0 36 00 36 00 36 00 36 00 36 00 36 00 36 00 36 00 6 6 6 6 6 6 6 6  
ds:00E0 36 00 36 00 36 00 36 00 36 00 36 00 36 00 36 00 6 6 6 6 6 6 6 6  
ds:00F0 36 00 36 00 36 00 36 00 36 00 36 00 36 00 36 00 6 6 6 6 6 6 6 6  
ds:0100 34 00 34 00 34 00 34 00 34 00 34 00 34 00 34 00 4 4 4 4 4 4 4 4  
ds:0110 34 00 34 00 34 00 34 00 34 00 34 00 34 00 34 00 4 4 4 4 4 4 4 4  
ds:0120 33 00 33 00 33 00 33 00 33 00 33 00 33 00 33 00 3 3 3 3 3 3 3 3  
ds:0130 33 00 33 00 33 00 33 00 33 00 33 00 33 00 33 00 3 3 3 3 3 3 3 3  
ds:0140 33 00 33 00 33 00 33 00 33 00 33 00 33 00 33 00 3 3 3 3 3 3 3 3

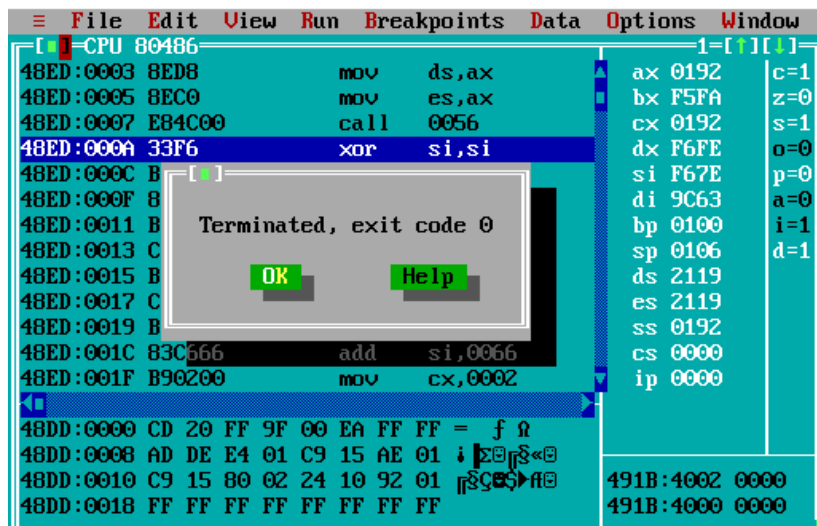
Зміщуємось на рядок і знову викликаємо функцію запису.

Dump

ds:0000 39 00 39 00 39 00 39 00 39 00 39 00 39 00 39 00 9 9 9 9 9 9 9 9  
ds:0010 39 00 39 00 39 00 39 00 39 00 39 00 39 00 39 00 9 9 9 9 9 9 9 9  
ds:0020 39 00 39 00 39 00 39 00 39 00 39 00 39 00 39 00 9 9 9 9 9 9 9 9  
ds:0030 39 00 39 00 39 00 39 00 39 00 39 00 39 00 39 00 9 9 9 9 9 9 9 9  
ds:0040 38 00 38 00 38 00 38 00 38 00 38 00 38 00 38 00 8 8 8 8 8 8 8 8  
ds:0050 38 00 38 00 38 00 38 00 38 00 38 00 38 00 38 00 8 8 8 8 8 8 8 8  
ds:0060 38 00 38 00 38 00 31 00 34 00 30 00 32 00 32 00 8 8 8 1 4 0 2 2  
ds:0070 30 00 30 00 33 00 38 00 38 00 38 00 38 00 38 00 0 0 3 8 8 8 8 8  
ds:0080 37 00 37 00 37 00 31 00 38 00 31 00 31 00 32 00 7 7 7 1 8 1 1 2  
ds:0090 30 00 30 00 32 00 37 00 37 00 37 00 37 00 37 00 0 0 2 7 7 7 7 7  
ds:00A0 37 00 37 00 37 00 37 00 37 00 37 00 37 00 37 00 7 7 7 7 7 7 7 7  
ds:00B0 37 00 37 00 37 00 37 00 37 00 37 00 37 00 37 00 7 7 7 7 7 7 7 7  
ds:00C0 36 00 36 00 36 00 36 00 36 00 36 00 36 00 36 00 6 6 6 6 6 6 6 6  
ds:00D0 36 00 36 00 36 00 36 00 36 00 36 00 36 00 36 00 6 6 6 6 6 6 6 6  
ds:00E0 36 00 36 00 36 00 36 00 36 00 36 00 36 00 36 00 6 6 6 6 6 6 6 6  
ds:00F0 36 00 36 00 36 00 36 00 36 00 36 00 36 00 36 00 6 6 6 6 6 6 6 6  
ds:0100 34 00 34 00 34 00 34 00 34 00 34 00 34 00 34 00 4 4 4 4 4 4 4 4  
ds:0110 34 00 34 00 34 00 34 00 34 00 34 00 34 00 34 00 4 4 4 4 4 4 4 4  
ds:0120 33 00 33 00 33 00 33 00 33 00 33 00 33 00 33 00 3 3 3 3 3 3 3 3  
ds:0130 33 00 33 00 33 00 33 00 33 00 33 00 33 00 33 00 3 3 3 3 3 3 3 3  
ds:0140 33 00 33 00 33 00 33 00 33 00 33 00 33 00 33 00 3 3 3 3 3 3 3 3







## 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.