

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 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: 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 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 '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 реєстр даних.

```

;-----II.МАКРОСИ-----
; Складний макрос для ініціалізації
MACRO M_Init ; Початок макросу
    mov ax, @data ; @data ідентифікатор, що створиться 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 прямокутник з датами 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 записувалися
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.

```
[J]=Dump
ds:0000 CD 20 FF 9F 00 EA FF FF AD DE E4 01 C9 15 AE 01 = f Ω i 20 78 3C 6D
ds:0010 C9 15 FF 02 24 10 92 01 01 01 01 02 FF FF FF 78 3C 6D
ds:0020 FF FF FF FF FF FF FF FF FF FF D3 48 94 F6 78 3C 6D
ds:0030 19 21 14 00 18 00 DD 4B FF FF FF FF 00 00 00 00 ↓? 1 ↑ | H
ds:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ♠
ds:0050 CD 21 CB 00 00 00 00 00 00 00 00 00 00 00 00 00 00 = ? 11
ds:0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0080 00 0D 00 FF 4E F7 61 22 A2 01 CA F7 01 80 FF FF J N=a "60 1=0C
ds:0090 CA F7 FF FF 00 00 64 F7 01 02 A5 32 CA F7 01 80 1= 2=0C 2=0C
ds:00A0 72 F7 28 06 18 00 14 00 6E F7 1C 21 A2 01 64 AD r=(+ i q n=-t0d d i
ds:00B0 14 00 00 00 64 AD 7E F7 2A 00 B3 08 64 AD 00 00 q d i ~ = 1 6 d i
ds:00C0 14 00 2E AB 48 AD 8C F7 8B 00 44 12 14 00 2E AB q 1/2 H i i ~ i D ? q 1/2
ds:00D0 48 AD 01 00 9C F7 69 01 9B 0B 02 00 01 00 48 AD H i 0 i ~ i 0 C 0 H i
ds:00E0 29 54 2E AB 2A F8 C0 02 5D 38 2E AB 48 AD 29 54 ) T 1/2 ~ ~ 1 0 18 1/2 H i ) T
ds:00F0 00 00 44 3A 5C 54 44 48 45 4C 50 2E 54 44 48 00 D : \TDHELP.TDH
ds:0100 B8 F7 48 8E D8 BE C0 E8 4C 00 33 F6 B8 00 00 BB 1=H 0 + 1 0 L 3 ? 1 i
ds:0110 D0 B4 09 CD 21 B4 01 CD 21 BE 00 00 83 C6 66 B9 11 0 = ? 1 0 = ? 1 1 â ? t 11
ds:0120 02 00 BF 07 02 51 E8 5B 00 83 C6 10 E8 55 00 83 0 1 . 0 Q 0 i â ? t 0 U â
ds:0130 C6 10 E8 4F 00 59 83 C6 10 E2 E7 BF 07 02 E8 43 t 0 U Yâ t ? t 1 . 0 0 C
ds:0140 00 83 C6 10 E8 3D 00 B4 09 CD 21 B4 01 CD 21 B4 â ? t 0 = - 0 = ? 1 0 = ? 1
```

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	

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

Після сортування масиву:

```

[ ]=CPU 80486                                     1=[ ]=
cs:0000 B8F748      mov     ax,48F7
cs:0003 8ED8        mov     ds,ax
cs:0005 8EC0        mov     es,ax
cs:0007 E84C00      call    0056
cs:000A 33F6        xor     si,si
cs:000C B80000      mov     ax,0000
cs:000F 8BD0        mov     dx,ax
cs:0011 B409        mov     ah,09
cs:0013 CD21        int     21
cs:0015 B401        mov     ah,01
cs:0017 CD21        int     21
cs:0019 BE0000      mov     si,0000
cs:001C 83C666      add     si,0066

ax 0039      c=0
bx 0000      z=1
cx 0000      s=0
dx 0039      o=0
si 0000      p=1
di 0000      a=0
bp 0000      i=1
sp 4000      d=0
ds 48F7
es 48F7
ss 491B
cs 48ED
ip 000A

48DD:0000 CD 20 FF 9F 00 EA FF FF = f 0
48DD:0008 AD DE E4 01 C9 15 AE 01  i 20 8<0
48DD:0010 C9 15 80 02 24 10 92 01  SCES 00
48DD:0018 01 01 01 00 02 FF FF FF 000 0

ss:4002 0000
ss:4000 0000

[ ]=Dump                                     2=[ ]=
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 38 00 38 00 38 00 38 00 38 00 8 8 8 8 8 8 8 8
ds:0070 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: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

```

Далі виконується виведення на екран відсортованого масиву:

[illegible]

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				

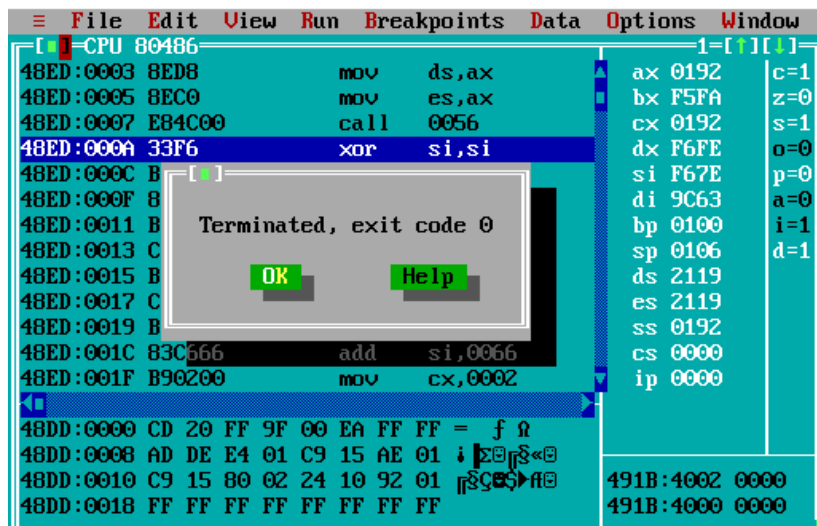
Запис в регістр di початок сегменту пам'яті, де зберігаються дати.

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.



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.