МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота №6**

з дисципліни « Програмування на асемблері »

на тему: «РЯДКОВІ ІНСТРУКЦІЇ І МАСИВИ»

Виконав: студент 2 курсу групи № 622п

121 «Інженерія програмного забезпечення»

(код спеціальності)

Зайченко Я. І.

(ПІБ студента)

Прийняв: ст. викладач каф. 603

\_ Дем’яненко В. А.\_\_\_\_\_\_

(ПІБ викладача)

Національна шкала:

Кількість балів:

Харків – 2024

**Мета роботи:** вивчити команди переміщення даних, команди повторення строкових інструкцій; вивчити організацію масивів мовою Асемблер.

Структура звіту

1. написання коду програми з ім'ям *Lab6.asm.*
2. процес створення виконуваного файлу та процес відладки;
3. код програми з коментарями про зміст регістрів усіх рядків у сегменті коду.
4. написання іншої програми й стеження за її роботою. Записаний результат роботи програми.
5. висновок.

Виконання роботи

Створюємо новий asm файл з кодом поданим нижче:

SSEG segment para stack 'stack'

Db 1, 2, 3, 4, 5, 6, 7, 8, 9, 128dup(0Ah)

SSEG ends

DSEG segment para public 'data'

B\_TAB db 01Ah, 02Bh, 03Ch, 04Dh, 05Eh, 06Fh, 07Ah, 08Bh

W\_TAB dw 01A2Bh, 03C4Dh, 05E6Fh, 07A8Bh

B\_TAB1 db 0Ah, 8 dup(1)

W\_TAB1 dw 8 dup(1)

DSEG ends

ESEG segment

W\_TAB2 dw 11h, 12h, 13h, 14h, 15h, 16h, 17h, 18h

ESEG ends

CSEG segment para public 'code'

PROG proc far

assume ds : DSEG, cs : CSEG, ss : SSEG, es : ESEG

push ds

mov ax, 0

push ax

mov ax, dseg

mov ds, ax

mov ax, eseg

mov es, ax

mov al, -3

mov ax, 3

mov B\_TAB, -3

mov W\_TAB, -3

mov ax, 2A1Bh

mov bl, al

mov bh, al

sub ax, bx

sub ax, ax

mov ax, W\_TAB

mov ax, W\_TAB + 3

mov ax, W\_TAB + 5

mov al, byte ptr W\_TAB + 6

mov al, B\_TAB

mov al, B\_TAB + 2

mov ax, word ptr B\_TAB

mov es : W\_TAB2 + 4, ax

mov bx, offset B\_TAB

mov si, offset B\_TAB + 1

mov di, offset B\_TAB + 2

mov dl, [bx]

mov dl, [si]

mov dl, [di]

mov ax, [di]

mov bp, bx

mov al, [bp]

mov al, ds: [bp]

mov al, es : [bx]

mov ax, cs : [bx]

mov ax, [bx] + 2

mov ax, [bx] + 4

mov ax, [bx + 2]

mov ax, [4 + bx]

mov ax, 2 + [bx]

mov ax, 4 + [bx]

mov al, [bx] + 2

mov bp, bx

mov ax, [bp + 2]

mov ax, ds: [bp] + 2

mov ax, ss: [bx + 2]

mov si, 2

mov ah, B\_TAB[si]

mov al, [B\_TAB + si]

mov bh, [si + B\_TAB]

mov bl, [si] + B\_TAB

mov bx, es:W\_TAB2[si]

mov di, 4

mov bl, byte ptr es : W\_TAB2[di]

mov bl, B\_TAB[si]

mov bx, offset B\_TAB

mov al, 3[bx][si]

mov ah, [bx + 3][si]

mov al, [bx][si + 2]

mov ah, [bx + si + 2]

mov bp, bx

mov ah, 3[bp][si]

mov ax, ds:3[bp][si]

mov ax, word ptr ds : 2[bp][si]

ret

PROG endp

CSEG ends

end PROG

Програмний застосунок emu8086 має в собі шаблони виконуваних файлів \*.com, \*.exe. в нашому випадку будемо створювати \*.exe файл.

SSEG segment para stack 'stack'.

Даний фрагмент коду робить неможливим створення виконуваного \*.com файлу. Оскільки формат .com не підтримує сегменти стеку та передбачений для простих програм, які не вимагають великої кількості пам'яті або складних обробок. Сегмент стеку, в якому зазвичай зберігаються локальні змінні та адреси повернення, не є допустимим у файлі формату \*.com.

На основі роботи програми в таблиці записані очікувані значення операндів

Таблиця 1 – значення операндів

|  |  |  |
| --- | --- | --- |
| Оператор | Операнд-приймач | |
| До виконання | Після виконання |
| push ds | - | Стек: [DSEG] |
| mov ax, 0 | - | ax = 0 |
| push ax | - | Стек: [DSEG, 0] |
| mov ax, dseg | - | ax = DSEG |
| mov ds, ax | - | ds = DSEG |
| mov ax, eseg | - | ax = ESEG |
| mov es, ax | - | es = ESEG |
| mov al, -3 | - | al = -3 |
| mov ax, 3 | - | ax = 3 |
| mov B\_TAB, -3 | - | B\_TAB[0] = -3 |
| mov W\_TAB, -3 | - | W\_TAB[0] = -3 |
| mov ax, 2A1Bh | - | ax = 2A1Bh |
| mov bl, al | 2A1Bh | bl = al = 2Bh (lower byte) |
| mov bh, al | 2A1Bh | bh = al = 2Ah (higher byte) |
| sub ax, bx | 2A1Bh - 2Bh | ax = 1Eh |
| sub ax, ax | 1Eh - 1Eh | ax = 0 |
| mov ax, W\_TAB | 0 | ax = W\_TAB[0] = 1A2Bh |
| mov ax, W\_TAB + 3 | 1A2Bh | ax = W\_TAB[3] = 7A8Bh |
| mov ax, W\_TAB + 5 | 7A8Bh | ax = W\_TAB[5] = 131Ch |
| mov al, byte ptr W\_TAB + 6 | 131Ch | al = W\_TAB[6] = 18h |
| mov al, B\_TAB | 18h | al = B\_TAB[0] = 1Ah |
| mov al, B\_TAB + 2 | 1Ah | al = B\_TAB[2] = 3Ch |
| mov ax, word ptr B\_TAB | 3Ch | ax = B\_TAB[0] + B\_TAB[1]\*256 = 2B1Ah |
| mov es: W\_TAB2 + 4, ax | 2B1Ah | W\_TAB2[4] = ax = 2B1Ah |
| mov bx, offset B\_TAB | 2B1Ah | bx = адреса B\_TAB |
| mov si, offset B\_TAB + 1 | адреса B\_TAB | si = адреса B\_TAB + 1 |
| mov di, offset B\_TAB + 2 | адреса B\_TAB + 1 | di = адреса B\_TAB + 2 |
| mov dl, [bx] | адреса B\_TAB + 2 | dl = B\_TAB[0] = 1Ah |
| mov dl, [si] | 1Ah | dl = B\_TAB[1] = 2Bh |
| mov dl, [di] | 2Bh | dl = B\_TAB[2] = 3Ch |
| mov ax, [di] | 3Ch | ax = (WORD)B\_TAB[2] \* 256 + B\_TAB[3] = 4D3Ch |
| mov bp, bx | 4D3Ch | bp = bx |
| mov al, [bp] | 4D3Ch | al = B\_TAB[0] = 1Ah |
| mov al, ds: [bp] | 1Ah | al = ds:[bp] = DSEG:[адреса B\_TAB] |
| mov al, es : [bx] | DSEG:[адреса B\_TAB] | al = es:[bx] = ESEG:[адреса B\_TAB] |
| mov ax, cs : [bx] | ESEG:[адреса B\_TAB] | ax = cs:[bx] = CSEG:[адреса B\_TAB] |
| mov ax, [bx] + 2 | CSEG:[адреса B\_TAB] | ax = B\_TAB[2] + B\_TAB[3]\*256 = 3C4Dh |
| mov ax, [bx] + 4 | 3C4Dh | ax = B\_TAB[4] + B\_TAB[5]\*256 = 5E6Fh |
| mov ax, 2 + [bx] | 5E6Fh | ax = 2 + B\_TAB[0] = 1Ch |
| mov ax, 4 + [bx] | 1Ch | ax = 4 + B\_TAB[0] = 20h |
| mov al, [bx] + 2 | 20h | al = B\_TAB[2] = 3Ch |
| mov bp, bx | 3Ch | bp = bx |
| mov ax, [bp + 2] | 3Ch | ax = B\_TAB[2] + B\_TAB[3]\*256 = 3C4Dh |
| mov ax, ds: [bp] + 2 | 3C4Dh | ax = ds:[bp] + B\_TAB[2] + B\_TAB[3]\*256 |
| mov ax, ss: [bx + 2] | 3C4Dh | ax = ss:[bx] + B\_TAB [2] + B\_TAB[3]\*256 |
| mov si, 2 | 3C4Dh | si = 2 |
| mov ah, B\_TAB[si] | 3C4Dh | ah = B\_TAB[2] = 3Ch |
| mov al, [B\_TAB + si] | 3Ch | al = B\_TAB[2] = 3Ch |
| mov bh, [si + B\_TAB] | 3Ch | bh = B\_TAB[2] = 3Ch |
| mov bl, [si] + B\_TAB | 3Ch | bl = B\_TAB[0] + B\_TAB[si] = 1Ah + B\_TAB[2] = 5Eh |
| mov bx, es:W\_TAB2[si] | 5Eh | bx = W\_TAB2[2] = 13h |
| mov di, 4 | 13h | di = 4 |
| mov bl, byte ptr es : W\_TAB2[di] | 4 | bl = W\_TAB2[4] = 15h |
| mov bl, B\_TAB[si] | 15h | bl = B\_TAB[2] = 3Ch |
| mov bx, offset B\_TAB | 3Ch | bx = адреса B\_TAB |
| mov al, 3[bx][si] | 3Ch | al = B\_TAB[3] = 4Dh |
| mov ah, [bx + 3][si] | 4Dh | ah = B\_TAB[3 + si] = 5Eh |
| mov al, [bx][si + 2] | 5Eh | al = B\_TAB[si + 2] = 3Ch |
| mov ah, [bx + si + 2] | 3Ch | ah = B\_TAB[si + 2 + si] = B\_TAB[2 + 2] = 5Eh |
| mov bp, bx | 5Eh | bp = bx |
| mov ah, 3[bp][si] | 5Eh | ah = 3 + B\_TAB[si] = 3 + B\_TAB[2] = 3 + 3Ch = 3Fh |
| mov ax, ds:3[bp][si] | 3F5Eh | ax = ds:[bx + 3 + si] = DSEG:[адреса B\_TAB + 3 + 2] |
| mov ax, word ptr ds : 2[bp][si] | 3F5Eh | ax = ds:[bx + 2 + si] = DSEG:[адреса B\_TAB + 2 + 2] |

Висновок

В ході виконання лабораторної роботи було проведено дослідження режимів адресації та визначено правила їх використання, а також проаналізовано процес створення виконуваного файлу. Одним з результатів роботи стала розробка таблиці очікуваних значень операндів, що базується на функціонуванні програми. Мета цієї роботи полягала у вивченні режимів адресації, визначення правил їх використання, а також у поглибленні розуміння директив визначення даних та стеку.