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 №4**

з дисципліни « Програмування на асемблері »

на тему: «КОМАНДИ ПЕРЕСИЛАННЯ ДАНИХ. СТЕК»

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:** вивчення режимів адресації, правил адресації при використанні регістрів, директив визначення даних, стеку.

Структура звіту

1. процес створення виконуваного файлу;
2. таблиця очікуваних значень операндів на основі роботи програми;
3. 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 |
| mov al, -3 | -3 | -3 |
| mov ax, 2A1Bh | 2A1Bh | 2A1Bh |
| mov bl, al | -3 | -3 |
| mov bh, al | -3 | -3 |
| sub ax, bx | 2A1Bh - (-3) | 2A1Eh |
| sub ax, ax | 2A1Eh - 2A1Eh | 0 |
| mov ax, W\_TAB | 1A2Bh | 1A2Bh |
| mov ax, W\_TAB + 3 | 1A2Bh + 3 | 1A2Eh |
| mov ax, W\_TAB + 5 | 1A2Bh + 5 | 1A33h |
| mov al, byte ptr W\_TAB + 6 | byte ptr (1A2Bh + 6) | 3Ch |
| mov al, B\_TAB | 1Ah | 1Ah |
| mov al, B\_TAB + 2 | 1Ah + 2 | 1Ch |
| mov ax, word ptr B\_TAB | word ptr (1Ah) | 1Ah |
| mov es: W\_TAB2 + 4, ax | W\_TAB2[4] = 1Ah | W\_TAB2[4] = 1Ah |
| mov bx, offset B\_TAB | bx = offset B\_TAB | bx = адреса B\_TAB в пам'яті |
| mov si, offset B\_TAB + 1 | si = offset B\_TAB + 1 | si = адреса B\_TAB + 1 в пам'яті |
| mov di, offset B\_TAB + 2 | di = offset B\_TAB + 2 | di = адреса B\_TAB + 2 в пам'яті |
| mov dl, [bx] | dl = B\_TAB[0] | dl = 01Ah |
| mov dl, [si] | dl = B\_TAB[1] | dl = 02Bh |
| mov dl, [di] | dl = B\_TAB[2] | dl = 03Ch |
| mov ax, [di] | ax = (WORD)B\_TAB[2] | ax = 03C4h |
| mov bp, bx | bp = bx | bp = адреса B\_TAB в пам'яті |
| mov al, [bp] | al = B\_TAB[0] | al = 01Ah |
| mov al, ds: [bp] | al = ds:[bp] | al = ds:B\_TAB[0] |
| mov al, es : [bx] | al = es:[bx] | al = es:B\_TAB[0] |
| mov ax, cs : [bx] | ax = cs:[bx] | ax = cs:B\_TAB[0] |
| mov ax, [bx] + 2 | ax = B\_TAB[2] | ax = 03Ch |
| mov ax, [bx] + 4 | ax = B\_TAB[4] | ax = 05Eh |
| mov ax, [bx + 2] | ax = B\_TAB[2] | ax = 03Ch |
| mov ax, [4 + bx] | ax = B\_TAB[4] | ax = 05Eh |
| mov ax, 2 + [bx] | ax = 2 + B\_TAB[0] | ax = 01Ch |
| mov ax, 4 + [bx] | ax = 4 + B\_TAB[0] | ax = 01Eh |
| mov al, [bx] + 2 | al = B\_TAB[2] | al = 03Ch |
| mov bp, bx | bp = bx | bp = адреса B\_TAB в пам'яті |
| mov ax, [bp + 2] | ax = B\_TAB[2] | ax = 03Ch |
| mov ax, ds: [bp] + 2 | ax = ds:[bp + 2] | ax = ds:B\_TAB[2] |
| mov ax, ss: [bx + 2] | ax = ss:[bx + 2] | ax = ss:B\_TAB[2] |
| mov si, 2 | si = 2 | si = 2 |
| mov ah, B\_TAB[si] | ah = B\_TAB[2] | ah = 03Ch |
| mov al, [B\_TAB + si] | al = B\_TAB[2] | al = 03Ch |
| mov bh, [si + B\_TAB] | bh = B\_TAB[2] | bh = 03Ch |
| mov bl, [si] + B\_TAB | bl = B\_TAB[2] | bl = 03Ch |
| mov bx, es:W\_TAB2[si] | bx = W\_TAB2[2] | bx = 13h |
| mov di, 4 | di = 4 | di = 4 |
| mov bl, byte ptr es : W\_TAB2[di] | bl = byte ptr W\_TAB2[4] | bl = byte ptr 14h |
| mov bl, B\_TAB[si] | bl = B\_TAB[2] | bl = 03Ch |

Таблиця 1 – таблиця значень регістрів розробленої програми для Emu8086;

|  |  |  |
| --- | --- | --- |
| **Крок** | **Зміна регістру** | **Пояснення** |
|  | AX(H) = 02 | AH (вищий байт регістру AX) отримує значення 02h. Це код служби INT 21h для виводу символу. |
| IP = 0102 | Інструкція INT 21h виконується, тому IP (вказівник інструкційного регістру) збільшується на 2. |
|  | DX(L) = 41 | DL (нижній байт регістру DX) отримує значення 41h, яке відповідає символу 'A'. |
| IP = 0104 | Інструкція INT 20h виконується, IP збільшується на 2. |
|  | CS = F400 | Сегментний регістр CS змінюється на F400h, що є адресою INT 20h у сегменті виходу (зазвичай DOS). |
| IP = 0200 | IP змінюється на адресу INT 20h в новому сегменті. |
| SS = FFF8 | Стековий сегментний регістр SS змінюється на FFF8h, щоб відведені дані могли бути збережені у стеку. |
|  | AX(L) = 41 | AL (нижній байт регістру AX) отримує значення 41h, яке відповідає символу 'A'. |
| IP = 0204 | IP збільшується на 2. |
|  | CS = 0700 | Повертаємось до вихідного сегменту CS. |
| IP = 0106 | IP змінюється на наступну інструкцію після INT 21h. |
| SP = FFFE | Вказівник стеку SP зменшується, що може означати видалення даних зі стеку. |
|  | CS = F400 | Сегментний регістр CS знову змінюється на F400h, повертаємось до INT 20h. |
| IP = 0150 | IP змінюється на адресу INT 20h у вихідному сегменті. |
| SP = FFF8 | Вказівник стеку SP змінюється, можливо, для відведення місця для збереження даних. |
|  | IP = 0154 | IP змінюється на наступну інструкцію після INT 20h. |

Результат виконання коду в емуляторі подано на рисунку 1

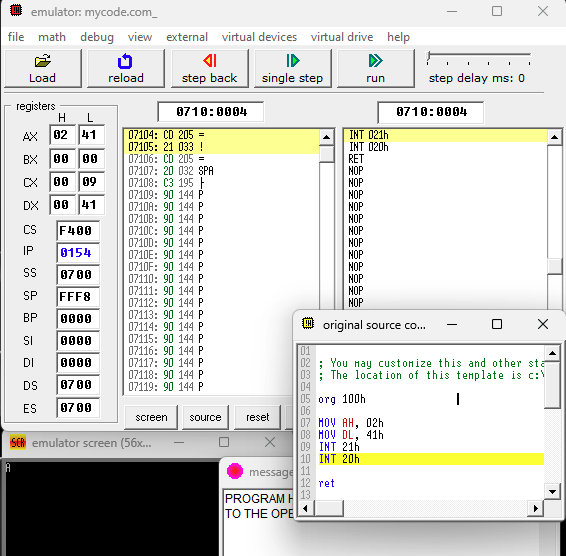


Рисунок 1 – екранні форми виконання створеної програми для Emu8086;

В програмі Debug вводимо спочатку флаг «-а». Та вводимо код програми. після пишемо команду «-g». Див. рисунок 2-3.

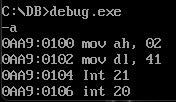


Рисунок 2 – екранна форма написання програми для Debug

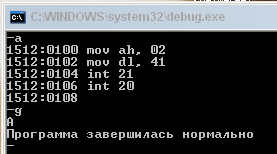


Рисунок 3 – екранна форма виконання створеної програми для Debug (звичайне виконання);

Для запуску коду в покроковому режимі повторюємо попередні кроки та вводимо команду «t».

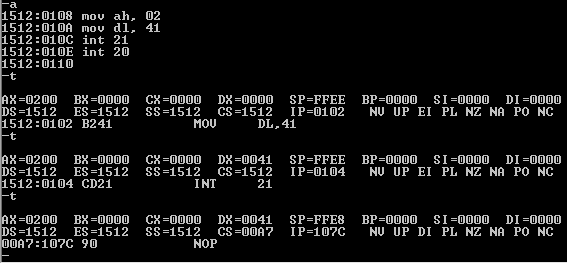


Рисунок 4 –кроки програми в Debug

Для завершення виконання коду необхідно ввести команду «g»

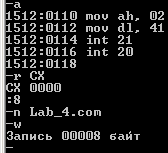


Рисунок 5 – створення \*.com файлу з коду програми

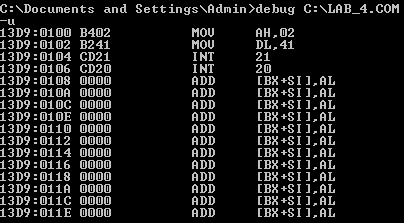


Рисунок 6 – результат дизасемблювання файлу \*.com з кодом програми

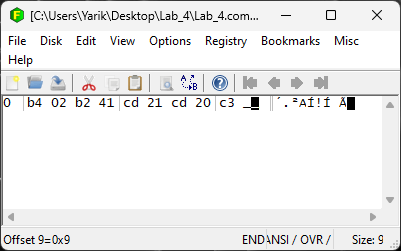


Рисунок 7 – файл \*.com в шістнадцятковому редакторі

Висновок

Під час виконання лабораторної роботи ознайомилися з роботою різних асемблерів, зокрема Emu8086 та Debug. Було написано програму мовою асемблер, яка виводила символ 'A' на екран, використовуючи системний виклик DOS INT 21h.

У роботі була розроблена таблиця значень регістрів для програми, виконаної в середовищі Emu8086. Також були створені екранні форми виконання програми для Emu8086 та Debug. У Debug було вивчено, як здійснюється послідовне виконання програми, а також як проводиться дизасемблювання.

Додатково був перевірений машинний код нашої програми в шістнадцятковому редакторі.

Ця лабораторна робота дозволила отримати практичні навички роботи з асемблером, а також вивчити процес виконання програм на низькому рівні, що було важливим для розуміння роботи комп'ютерних систем.