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 №7**

з дисципліни « Програмування на асемблері »

на тему: «ПРОЦЕДУРИ»

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:** ознайомитися з основами роботи з процедурами, видами адресацій процедур (NEAR, FAR); навчитися використовувати локальні змінні.

Структура звіту

1. написання коду програми з ім'ям *Lab7.asm.*
2. процес створення виконуваного файлу та процес відладки;
3. код програми з коментарями до кожного рядку програми з вмістом регістрів.
4. висновок.

Виконання роботи

Створюємо новий asm файл з кодом поданим нижче:

;Ця програма переводить вміст регістра AX у

;шістнадцяткове подання й записує результат у рядок,

;зміщення якого зберігається в регістрі BX

.model tiny

.stack 100h

.data

outStr db '0000$' ;Вихідний рядок

.code

;Вхідні дані для процедури translByte AL – байт, який

;потрібно перевести

;Вихідні дані BX – зміщення рядка, у перші два байти

;якого буде записано результат

translByte proc

push ax

push ax

shr al,4

cmp al,9

ja greater10

mov byte ptr [bx],'0'

add [bx],al

jmp next4Bit

greater10:

mov byte ptr [bx],'A'

sub al,10

add [bx],al

next4Bit:

pop ax

and al,0Fh

cmp al,9

ja \_greater10

mov byte ptr [bx+1],'0'

add [bx+1],al

jmp exitByteProc

\_greater10:

mov byte ptr [bx+1],'A'

sub al,10

add [bx+1],al

exitByteProc:

pop ax

ret

translByte endp

translWord proc

push ax

push ax

shr ax,8

call translByte

pop ax

and ax,00FFh

add bx,2

call translByte

sub bx,2

pop ax

ret

translWord endp

start:

mov ax,@data

mov ds,ax

mov bx,OFFSET outStr

mov ax,60000

call translWord

mov ah,9

mov dx,OFFSET outStr

int 21h

mov ax,4c00h

int 21h

end start

Для відладки було використано емулятор DOS через розширення MASM/TASM для Visual Studio Code, результат відладки показано на рисунку 1

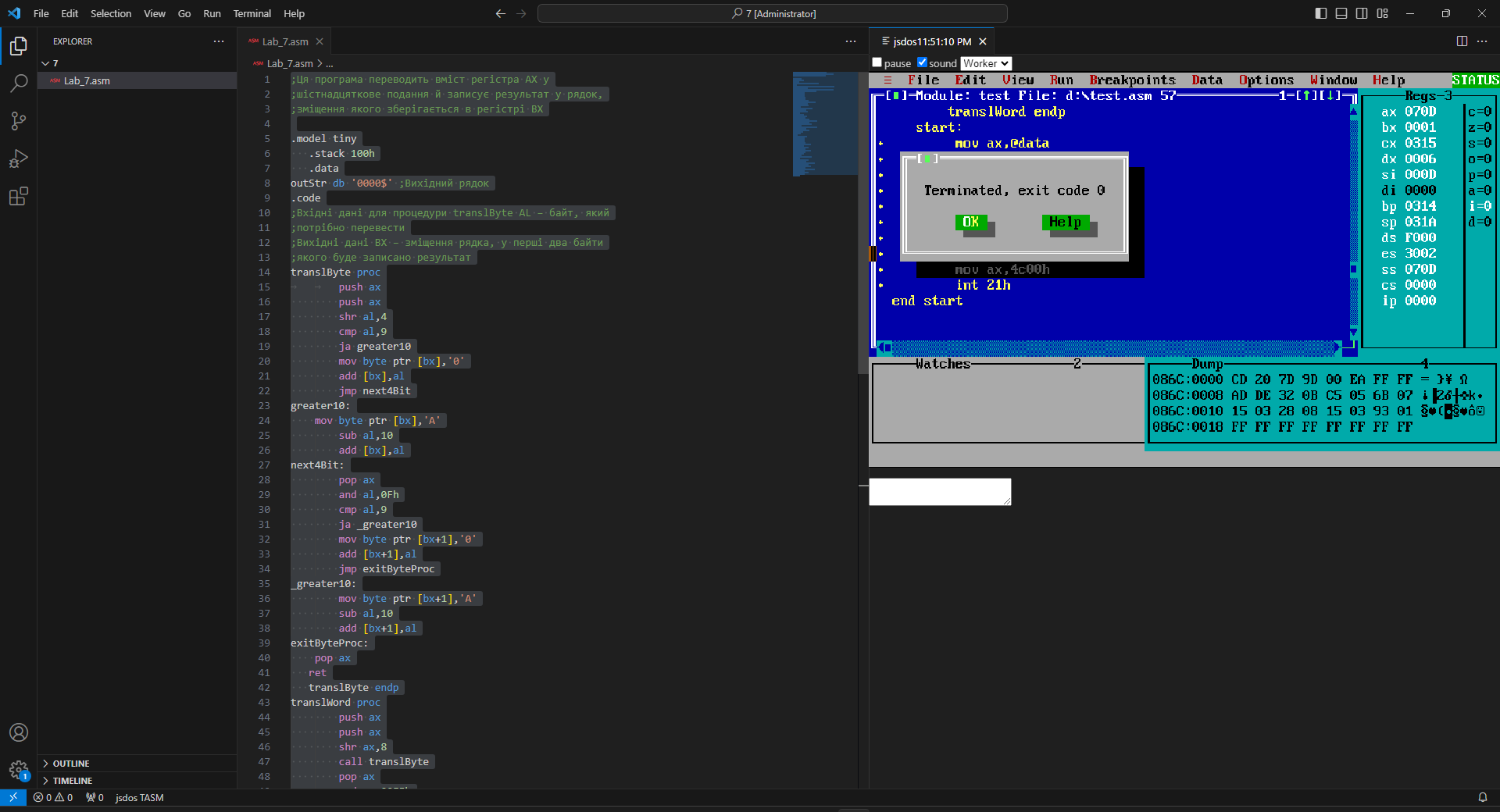


Рисунок 1 – відладка коду на асемблері

При відладці записано значення регістрів усіх рядків в сегменті коду.

;Ця програма переводить вміст регістра AX у

;шістнадцяткове подання й записує результат у рядок,

;зміщення якого зберігається в регістрі BX

.model tiny

.stack 100h

.data ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0100

outStr db '0000$' ;Вихідний рядок

.code

;Вхідні дані для процедури translByte AL – байт, який

;потрібно перевести

;Вихідні дані BX – зміщення рядка, у перші два байти

;якого буде записано результат

translByte proc

push ax ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0107

push ax ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0108

shr al,4 ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFA BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=010F

cmp al,9 ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFA BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0111

ja greater10 ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFA BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0113

mov byte ptr [bx],'0' ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFA BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0115

add [bx],al ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFA BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0118

jmp next4Bit ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFA BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=011A

greater10:

mov byte ptr [bx],'A'

sub al,10

add [bx],al

next4Bit:

pop ax ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFA BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0123

and al,0Fh ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0124

cmp al,9 ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0126

ja \_greater10 ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0128

mov byte ptr [bx+1],'0' ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=012A

add [bx+1],al ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=012E

jmp exitByteProc ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=0131

\_greater10:

mov byte ptr [bx+1],'A'

sub al,10

add [bx+1],al

exitByteProc:

pop ax ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFC BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=013C

ret ;AX=0000 BX=0000 CX=007B DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000 DS=0700 ES=0700 SS=0700 CS=0700 IP=013D

translByte endp

translWord proc

push ax

push ax

shr ax,8

call translByte

pop ax

and ax,00FFh

add bx,2

call translByte

sub bx,2

pop ax

ret

translWord endp

start:

mov ax,@data ;AX=0000 BX=0000 CX=007B DX=0000 SP=0100 BP=0000 SI=0000 DI=0000 DS=086C ES=086C SS=0884 CS=087C IP=005D

mov ds,ax ;AX=087C BX=0000 CX=007B DX=0000 SP=0100 BP=0000 SI=0000 DI=0000 DS=087C ES=086C SS=0884 CS=087C IP=0062

mov bx,OFFSET outStr ;AX=087C BX=0078 CX=007B DX=0000 SP=0100 BP=0000 SI=0000 DI=0000 DS=087C ES=086C SS=0884 CS=087C IP=0065

mov ax,60000 ;AX=EA60 BX=0078 CX=007B DX=0000 SP=0100 BP=0000 SI=0000 DI=0000 DS=087C ES=086C SS=0884 CS=087C IP=0068

call translWord ;AX=EA60 BX=0078 CX=007B DX=0000 SP=0100 BP=0000 SI=0000 DI=0000 DS=087C ES=086C SS=0884 CS=087C IP=006B

mov ah,9 ;AX=0960 BX=0078 CX=007B DX=0000 SP=0100 BP=0000 SI=0000 DI=0000 DS=087C ES=086C SS=0884 CS=087C IP=006D

mov dx,OFFSET outStr ;AX=0960 BX=0078 CX=007B DX=0078 SP=0100 BP=0000 SI=0000 DI=0000 DS=087C ES=086C SS=0884 CS=087C IP=0070

int 21h ;AX=0924 BX=0078 CX=007B DX=0078 SP=0100 BP=0000 SI=0000 DI=0000 DS=087C ES=086C SS=0884 CS=087C IP=0072

mov ax,4c00h ;AX=4C00 BX=0078 CX=007B DX=0078 SP=0100 BP=0000 SI=0000 DI=0000 DS=087C ES=086C SS=0884 CS=087C IP=0075

int 21h ;AX=070D BX=0000 CX=0315 DX=0006 SP=031A BP=0314 SI=0000 DI=0000 DS=0315 ES=3002 SS=070D CS=0000 IP=0000

end start

Висновок

Під час виконання лабораторної роботи ми ознайомились з основами роботи з процедурами у мові асемблера, вивчили різновиди адресації процедур (NEAR, FAR) та навчились використовувати локальні змінні для зберігання тимчасових даних.

Основними результатами роботи стало здобуття практичних навичок у написанні програм на асемблері з використанням процедурного підходу. Ми ретельно вивчили процес створення виконуваних файлів з допомогою асемблера та впровадили розгалуження та змінні для забезпечення більшої гнучкості та читабельності коду.

Отримані знання дозволили краще розуміти принципи адресації в асемблері, що є важливим елементом управління потоком виконання програм. Дослідження різних типів адресації процедур дало змогу зрозуміти їх відмінності та сфери застосування.

У цілому, лабораторна робота була корисною для поглиблення знань у мові асемблера та відкрила нові можливості для подальшого вивчення аспектів програмування на рівні машинного коду.