**Andrew Tran**

**Computer Science 2253 Lab Experience Seven**

Objectives:

1. Writing loops in Assembler
2. Utilizing predefined macros from the irvine32.inc library.

The purpose of this lab is for you to create your own assembly language instructions. I recommend you load a project solution into Visual C++ express/studio so you do not have to reset the environment variables each time you are to create an assembler program.

NOTE: If you have changed the location of the Irvine libraries, you will have to correct the project properties settings of the project so Visual Studio can locate these files. The authors website contains detail instructions on this is accomplished. The website is <http://kipirvine.com/asm/gettingStarted/index.htm>

Pay close attention to the status flags since we will be using this to control the branching instructions, i.e. looping and decision.

**What you are to do:**

1. Write a program that uses addition and subtraction to set and clear the following flags:
   1. carry
   2. zero
   3. sign
   4. overflow

Insert a call to dumpregs to display the registers and flags after each instruction that modifies the flags. Use comments within the code explaining how and why the flags were affected by your assembler instructions.

1. Do problems 2, 4, and 6 on page 187.

For each of the above exercises copy the program’s into a word document and paste the output window of the program below the source code in a word document.

**What to hand in:**

1. Compress the .asm files and the word document into a single file called {yourName}Lab7.zip. For example TimWrennLab7.zip.
2. Place the compressed file into the D2L DropBox labeled Lab Seven.
3. Hand-in hard copies of the programs created in this lab.
4. Hand-in a hard copy of the word document.

Due Date: As specified by the drop box date.

Problem 1:

INCLUDE Irvine32.inc

.data

.code

main PROC

; carry flag

mov al, 3

sub al, 4; the most significiant bit went out of range

call dumpregs

; zero flag

mov eax, 3

sub eax, 3; makes a zero value in eax

call dumpregs

; sign flag

mov al, 10

sub al, 11; makes a negative eax value

call dumpregs

; overflow flag

mov al, 10000000b

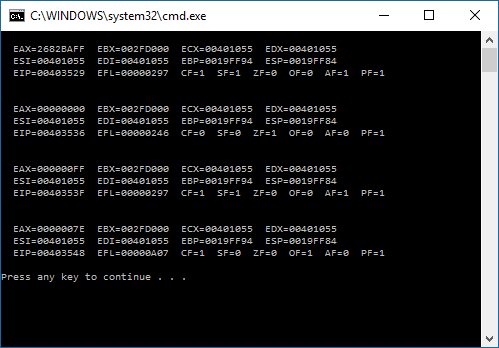
add al, 11111110b; two negative values make a positive

call dumpregs

exit

main ENDP

END main



Problem 2:

INCLUDE Irvine32.inc

.data

index DWORD 1

chars BYTE 'H','A','C','E','B','D','F','G'

links DWORD 0, 4, 5, 6, 2, 3, 7, 0

array BYTE 0, 0, 0, 0, 0, 0, 0, 0

.code

main PROC

mov ecx, 8

mov edi, 1

mov al, chars[1]

mov array, al

mov esi, index

mov ebx, offset links

L1:

mov eax, links[esi \* 4]

mov dl, chars[eax]

mov array[edi], dl

mov esi,eax

inc edi

loop L1

mov ecx,8

mov esi,0

L2:

mov al, array[esi]

inc esi

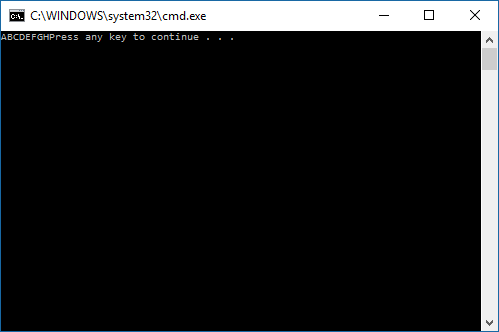
call WriteChar

loop L2

exit

main ENDP

END main



Problem 4:

INCLUDE Irvine32.inc

.data

string BYTE "Enter an integer: ",0

string1 BYTE "The sum is: ",0

sum DWORD 0

row BYTE 9

col BYTE 25

.code

main PROC

mov ecx, 2

mov ebx, 3

mov sum, 0

L2:

call Clrscr

L1:

dec ebx

mov dh, row

mov dl, col

call GoToXY

mov edx, OFFSET string

call WriteString

call ReadInt

add sum, eax

add row,2

mov dh, row

mov dl, col

call GoToXY

mov edx, OFFSET string1

call WriteString

mov eax, sum

call WriteInt

loop L1

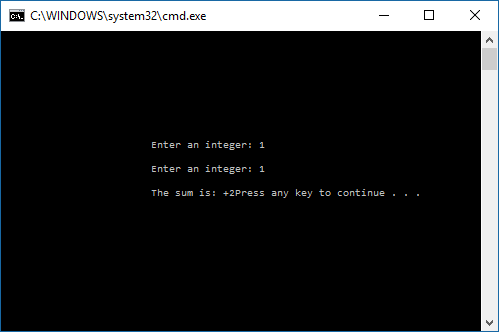
mov ecx, ebx

loop L2

exit

main ENDP

END main



Problem 6:

INCLUDE Irvine32.inc

.data

String BYTE 10 DUP(0), 0

.code

main PROC

mov ecx, 20

L1:

push ecx

mov ecx, 10

mov esi, OFFSET String

L2:

mov eax, 26

call RandomRange

add eax, 'A'

mov[esi], al

inc esi

loop L2

mov edx, OFFSET String

call WriteString

call Crlf

pop ecx

loop L1

exit

main ENDP

END main

