CSCI. 516 Program 2 - March 28, 2007

Write an Assembly language program to input, and compute the sum of three unsigned 64 bit integers, and display the result on the video screen.
Describe the capabilities of your program:
This program adds three unsigned 64 bits (decimal/hexadecimal - write the numerical bases of your input). The maximum size of each number in digits is (give this number).

Prompt the user with questions:
Enter the first number:
Enter the second number:
Enter the third number:
Display the result on a separate row:
The sum is:

The Program is due Monday April 16,2007, 2PM in the beginning of the regular lecture.
Please name your files using the same convention as you did for Program 1, just change #1 with #2. Turn a floppy with your source file together with all the files produced by the compiler.

Hint:

1) If you decide to work with decimal numbers and enter the numbers as strings you may use the procedure: ReadString **.**

ReadString - Reads a string from standard input, terminated by the Enter key.

To display your result you could use "WriteString".

WriteString - Writes a null-terminated string to standard output.

Ex:

```
.data
str1 BYTE "Consider this program seriously",0

.code
        mov edx,OFFSET str1
        call WriteString
        call Crlf
```

More info regarding these procedures is given on Pages 145, 147 – text book.
Now, you have two options:

a) convert the operands to binary, add them using a program similar to the program given below, then convert the sum to ASCII digit string and display it:

```
TITLE Extended Addition Example          (ExtAdd.asm)
; This program calculates the sum of two 64-bit integers.
INCLUDE Irvine16.inc

.data
op1 QWORD 0A2B2A40674981234h
op2 QWORD 08010870000234502h
sum DWORD 3 dup(?) ; = 0000000122C32B0674BB5736

.code
main PROC
        mov ax,@data
        mov ds,ax

        mov  esi,OFFSET op1             ; first operand
        mov  edi,OFFSET op2             ; second operand
        mov  ebx,OFFSET sum             ; sum operand
        mov  ecx,2                  ; number of doublewords
        call Extended_Add
        mov  esi,OFFSET sum             ; dump memory
        mov  ebx,4
        mov  ecx,3
        call DumpMem

        exit
main ENDP

;-----------------------------------------------------
Extended_Add PROC
;
; Calculates the sum of two extended integers that are
;   stored as an array of doublewords.
; Receives: ESI and EDI point to the two integers,
; EBX points to a variable that will hold the sum, and
; ECX indicates the number of doublewords to be added.
;-----------------------------------------------------
        pushad
        clc                     ; clear the Carry flag

L1:     mov eax,[esi]                   ; get the first integer
        adc eax,[edi]                   ; add the second integer
        pushfd                          ; save the Carry flag
        mov [ebx],eax                   ; store partial sum
        add esi,4               ; advance all 3 pointers
        add edi,4
```

```
        add ebx,4
        popfd                   ; restore the Carry flag
        loop L1             ; repeat the loop

        adc word ptr [ebx],0        ; add any leftover carry
        popad
        ret
Extended_Add ENDP
END main
```

b) Add the digit strings directly by successively adding each pair of ASCII. The sum is in an ASCII digit string, and can be directly displayed on the screen.
Ex.

```
mov ah,0
mov al,'8'          ; AL= 38h
add al,'2'          ; AL = 6Ah
aaa                 ; AL = 00h (adjust result)
or  al, 30h         ; AL = 30h = '0'
```

Note: for the summation of each pair, after the first one, you should use "adc" instead of "add". In order to display the carry which could come from the previous pair it is helpful to use a procedure similar to the one given below:

Ex.

```
mov ah,0
mov al,'8'          ; AX = 0038h
add al,'2'          ; AX = 006Ah
aaa                 ; AX = 0100h (adjust result)
or  ax,3030h        ; AX = 3130h = '10'
```

More info regarding instructions "aaa", "adc" and some example programs are given on Pages 249, 252 – text book.