Comp 3350: Computer Organization & Assembly Language
HW # 5: Theme: Data Definitions, Addressing Modes, Arrays

1. [Memory Map] Fill in the following memory diagram with the data provided below. Please assume that the data segment begins at 0x00404000.

```
        .data
        Alpha       WORD        33h, 24h
        Beta        BYTE        67h
        Gamma       DWORD       5677h
        Delta       BYTE        33h
```

Address	*Variable*	*Data*
00404000	**Alpha**	**33**
00404001		**00**
00404002		**24**
00404003		**00**
00404004	**Beta**	**67**
00404005	**Gamma**	**77**
00404006		**56**
00404007		**00**
00404008		**00**
00404009	**Delta**	**33**

2. [Addressing Modes] Copy the following code into your assembly development environment and single-step through it. For those instructions referencing memory, write the linear address.

```
TITLE Addressing Modes                  (main.asm)
INCLUDE Irvine32.inc

.data
alpha    DWORD       44h, 23h
beta           DWORD       6788h, 66h
gamma    DWORD       1234h

.code
main PROC
   mov eax, 12h;                  Immediate
   mov ecx, eax;                  Register to Register
   mov edi, OFFSET beta;          Immediate   00404008
   mov [gamma], eax;              Indirect    00404010
   mov esi, [gamma];              Direct      00404010
   mov esi, 4;                    Immediate
   mov eax, beta[esi];            Indirect-offset    0040400c
   mov ebx, OFFSET alpha;         Immediate   00404000
   mov eax, [ebx];                Indirect    00404000
   mov eax, 4[ebx];               Indirect-displacement 00404004
   mov eax, 4[ebx][esi];          Base-Indirect-displacement 00404008
exit
```

```
main ENDP
END main
```

3. [Indirect addressing] Write a program that adds a constant value to each element of an array and places the value in the ModArray. Use:

```
.data
Array       WORD 23h, 45h, 45h, 56h, 25h, 44h, 22h, 54h, 12h
ConstVal    WORD 20h
ModArray    WORD 9 DUP (?)
```

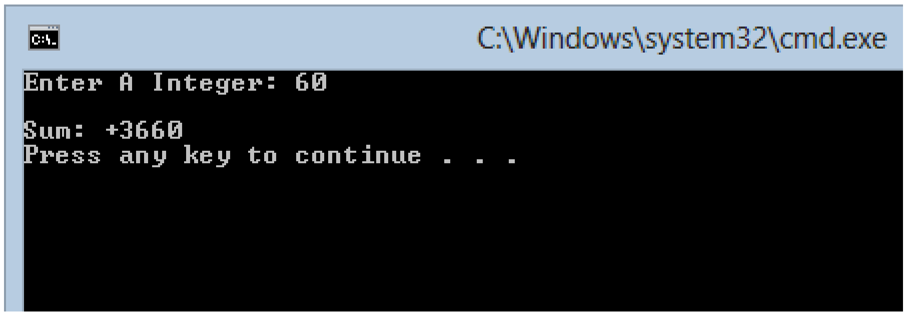
```
TITLE INDIRECT ADDRESSING
INCLUDE IRVINE32.INC
.DATA
      ARRAY       WORD  23H, 45H, 45H, 56H, 25H, 44H, 22H, 54H,
12H
      CONSTVAL    WORD  20H
      MODARRAY    WORD  9     DUP(?)

.CODE
MAIN PROC

      MOV ECX, LENGTHOF ARRAY; INITALIZE LOOP
      MOV ESI, 0
      L1:
            MOV AX, ARRAY[ESI]
            ADD AX, CONSTVAL
            MOV [MODARRAY+ESI], AX
            ADD ESI, TYPE ARRAY
      LOOP L1

      MOV EBX, OFFSET MODARRAY
      MOV ECX, LENGTHOF MODARRAY; INITALIZE LOOP
      MOV ESI, 0

      L2:
            MOV EAX, [EBX+ESI]
            CALL WRITEHEX
            CALL CRLF
            ADD ESI, TYPE MODARRAY
      LOOP L2
      EXIT
MAIN ENDP
END MAIN
```

4. [Loops] Write a program to compute the sum of first *n* even integers. *Sum = 2 + 4 + 6 … + n*. Your program must:

   a. Prompt user for integer *n*,
   b. Read the value of *n* from user input
   c. Calculate *Sum*, and;
   d. Print *Sum* on screen.

   Please use the "WriteInt" procedure, not "DumpRegs". Other relevant procedures: "ReadInt" and "WriteString." The calculation can be done in many ways, and all submissions that evidence proper programming practice are acceptable. In your homework submission, please embed both the code and one screen shot for *n = 60*. You can assume that the user is considerate and careful and thus only inputs even values for *n*.

SOURCE CODE IS ON NEXT PAGE…



```
Enter A Integer: 60

Sum: +3660
Press any key to continue . . .
```

```
TITLE COMPUTING THE SUM OF N EVEN INTEGERS
INCLUDE IRVINE32.INC
.code
main PROC
        call clrscr
        call PromptForN
        mov ecx, eax     ;MOVING THE INPUT VALUE TO ECX.
        call EvenInteger ; INPUT SHOULD BE IN ECX, OUTPUT SHOULD BE IN EAX
        call DisplaySum
        exit
main ENDP
;=====================================================================
COMMENT !
PromptForN subroutine recieves an integer as input from the user
This integer will be how many even integers to add together.!
;=====================================================================
PromptForN PROC
.data
        str1 BYTE "Enter A Integer: ",0
.code
        mov edx, OFFSET str1
        call WriteString ;Displays the string offset at edx
        call ReadInt     ;THE INTEGER STORED INTO EAX
        call Crlf
        ret
PromptForN ENDP

;=====================================================================
COMMENT !
EvenInteger SUBROUTINE USES THE ECX REG FOR AS THE NUMBER OF EVEN INTEGERS TO SUM TOGETHER
AND EAX REG IS THE RESULT.
EX. ECX = 5
THEN EAX = 30 !
;=====================================================================
EvenInteger PROC USES ecx esi edx
        mov eax, 2h ;This is our constant multiplier
        mov esi, 1h ;This is our increment multiplier
        mov ebx, 0h ;zero out the ebx reg for calculations
        LP1:
                push eax;PUSH THE CONSTANT MULTIPLIER ONTO THE STACK TO SAVE FOR LATER,
                        ;SINCE MUL STORES HE RESULTS IN EAX AND ALSO USES EAX AS THE MULTIPLIER
                mul esi ;MULTIPLIES ESI BY EAX WHICH IS TWO.
                add ebx, eax ;ADDS RESULT OF THE MULTIPLICATION TO THE PREVIOUS MULTIPLICATION TO EBX
                pop eax ;RESTORES THE CONSTANT MULTIPLIER, WHICH IS TWO
                inc esi ;INCREMENTS THE INCREMENT MULTIPLIER TO PREPARE FOR NEXT EVEN INTEGER.
        loop LP1
        mov eax, ebx
        ret
EvenInteger ENDP
;=====================================================================
COMMENT !
DisplaySum subroutine displays what's in the eax register assuming that
it is a summation of even integers.!
;=====================================================================
DisplaySum PROC USES edx
.data
        str2 BYTE "Sum: ",0
.code
        mov edx, OFFSET str2
        call WriteString
        call WriteInt
        call Crlf
        ret
DisplaySum ENDP
;=====================================================================
END main
```