Comp 3350: Computer Organization & Assembly Language
HW # 5: Theme: Data Definitions, Addressing Modes, Arrays
All main questions carry equal weight.

(Credit awarded to only those answers for which work has been shown.)

1. [Memory Map] Fill in the following memory diagram with the data provided below. Please assume that the data segment begins at 0x0065A200.

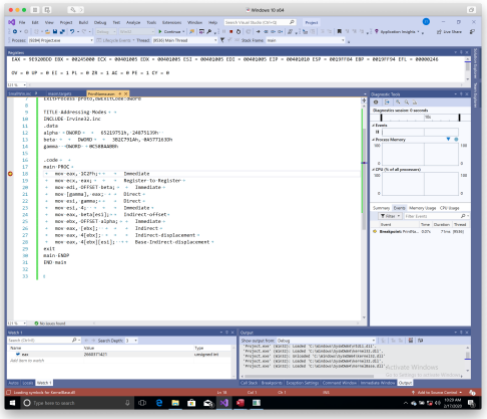
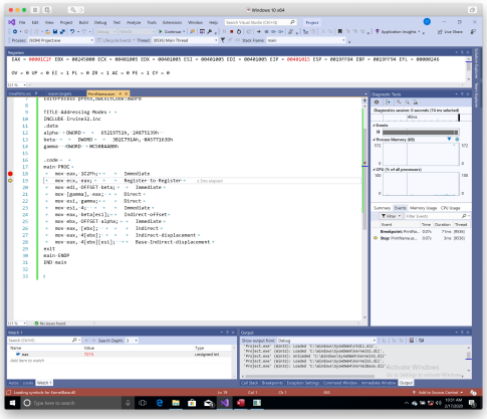
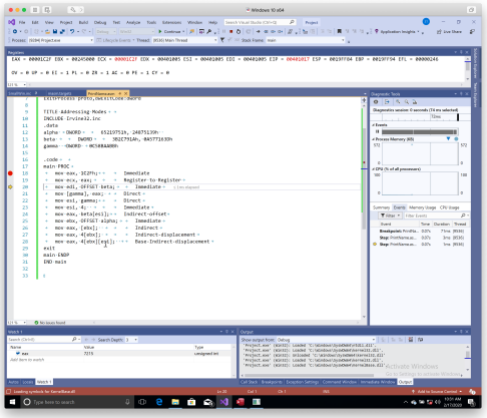
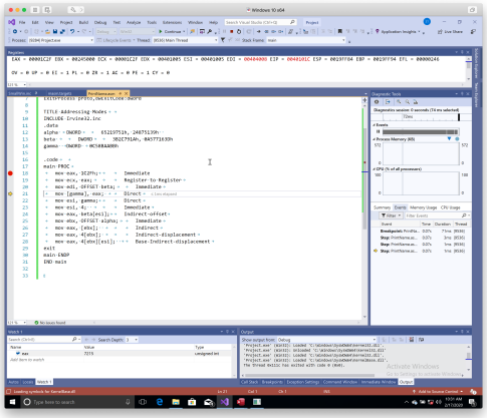
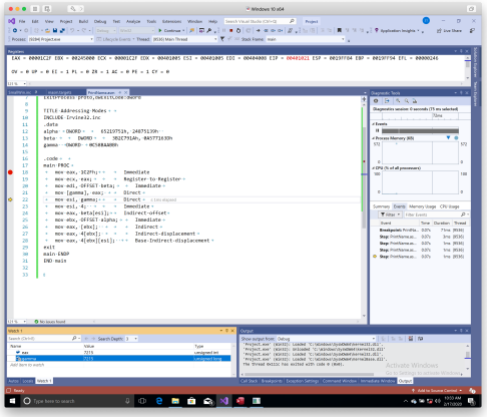
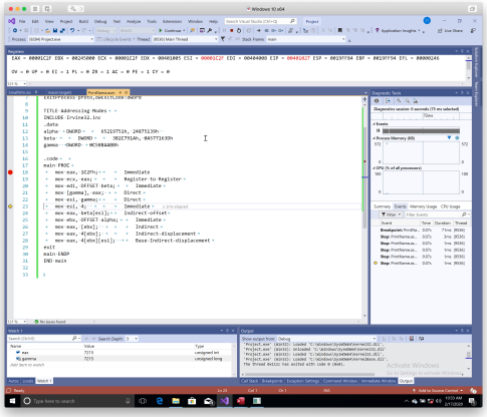
```
.data
Cat         BYTE        C7h
Dog         WORD        1234h
Horse       DWORD       0A1C1D1E1h, 22h
```

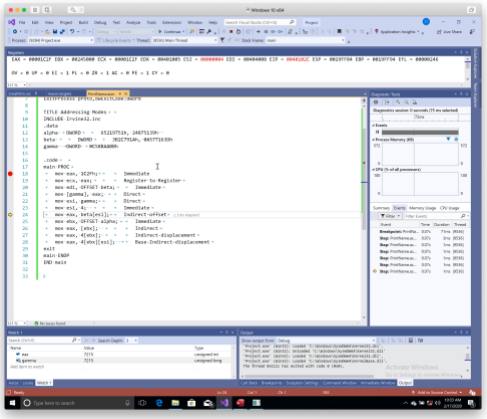
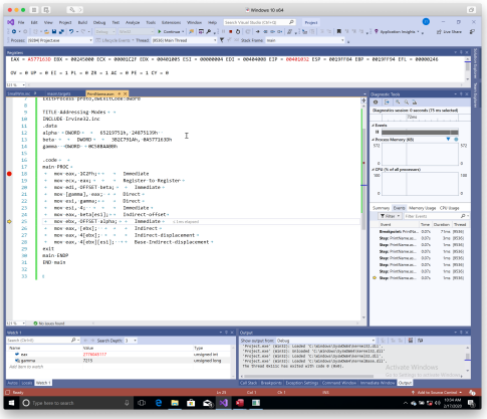
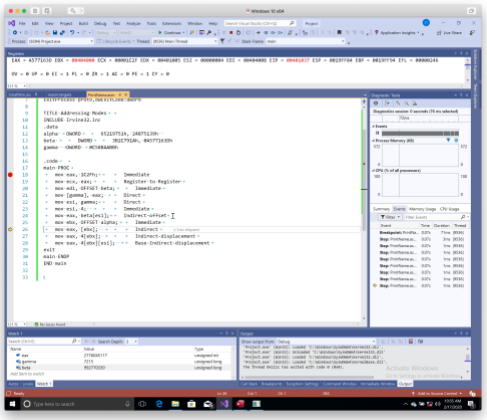
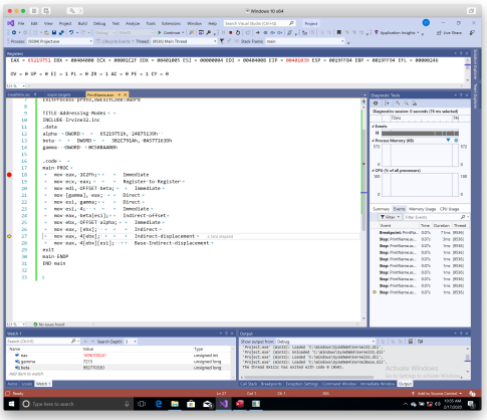
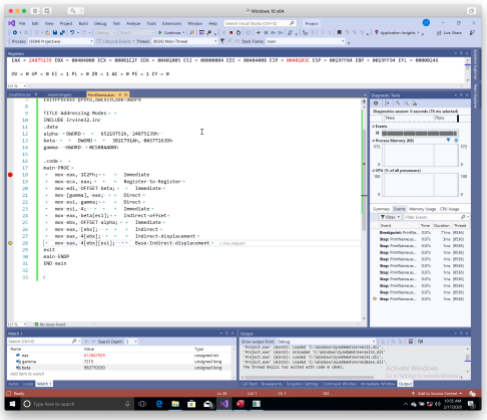
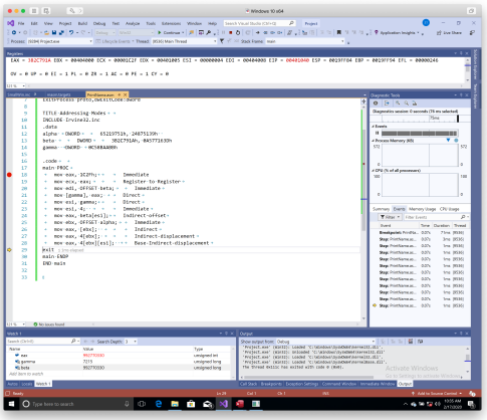
Variable	*Address*	*Data*
Cat	0x0065A200	***C7h***
Dog	0x0065A201	***34h***
	0x0065A202	***12h***
Horse	0x0065A203	***E1h***
	0x0065A204	***D1h***
	0x0065A205	***C1h***
	0x0065A206	***A1h***
	0x0065A207	***22h***
	0x0065A208	***00h***
	0x0065A209	***00h***
	0x0065A20A	***00h***

2. [Addressing Modes] Copy the following code into your assembly development environment and single-step through it. For each single step execution, submit the screenshot. For those instructions referencing memory, do the linear address computation by hand and typewrite it.

```
TITLE Addressing Modes                  (main.asm)
INCLUDE Irvine32.inc
.data
alpha       DWORD       65219751h, 24875139h
beta        DWORD       3B2C791Ah, 0A577163Dh
gamma       DWORD       0C58BAABBh

.code
main PROC
mov eax, 1C2Fh;                 Immediate

      mov eax, 1C2Fh;               Immediate
      mov ecx, eax;                 Register to Register
      mov edi, OFFSET beta;         Immediate
      mov [gamma], eax;             Direct
      mov esi, gamma;               Direct
      mov esi, 4;                   Immediate
      mov eax, beta[esi];           Indirect-offset
```

```
        mov ebx, OFFSET alpha;          Immediate
        mov eax, [ebx];                 Indirect
        mov eax, 4[ebx];                Indirect-displacement
        mov eax, 4[ebx][esi];           Base-Indirect-displacement
exit
main ENDP
END main
```

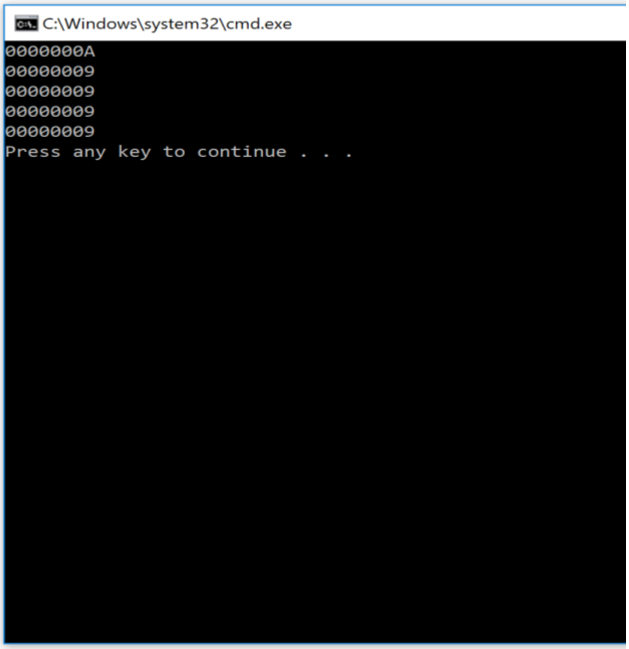







```
The linear address computation
        mov eax, beta[esi]  ;  Indirect-offset
                            ; eax = [beta+esi] = [0x00404008+4] = [0x0040400C]
                            ;       => 0xA577163d
        mov eax, [ebx];          Indirect
                            ; eax = [ebx] = [00404000]  => 0x65219751
        mov eax, 4[ebx];         Indirect-displacement
                            ; eax = [ebx+4] = [00404000+4] = [00404004]
                            ;       => 0x24875139
        mov eax, 4[ebx][esi];  Base-Indirect-displacement
                            ; eax = [ebx+esi+4] = [00404000+4+4] =
                                    [0x00404008] => 0x3b2c791a
```

3. [Indirect addressing] Write a program that adds the corresponding odd indexed elements of Array2 from Array1 and stores the results in Array3; e.g. for the 8th element, Array3 [7] ← Array1 [7] + Array2 [7]. Note that Array3 will have about half the number of elements of the other two arrays. Include commands to display the elements of all the arrays. Submit screenshot of the displays of the elements of all the arrays. You can use WriteInt or WriteHex to display the elements of the arrays. Fill in Array1 and Array2 each by your own ten numbers.

```
.data
Array1      WORD 7h, …
Array2      WORD 4h, …
Array3      WORD 10 DUP (?)

For example, if we have:

      Array1 WORD 9h, 8h, 7h, 6h, 5h, 4h, 3h, 2h, 1h
      Array2 WORD 1h, 2h, 2h, 4h, 4h, 6h, 6h, 8h, 8h

Then the expected result stored in Array3 should be:

      Ah,9h
```

```
INCLUDE Irvine32.inc

.data
Array1 word 9h, 8h, 7h, 6h, 5h, 4h, 3h, 2h, 1h
Array2 word 1h, 2h, 2h, 4h, 4h, 6h, 6h, 8h, 8h
Array3 word 10 DUP (?)

.code
main proc
    mov ecx, 5
    mov eax, OFFSET Array1
    mov ebx, OFFSET Array2
    mov esi, OFFSET Array3

; addition
L1: mov edx, [eax]
    add edx, [ebx]
    mov [esi], edx
    add eax, 4
    add ebx, 4
    add esi, 2
    loop L1

    mov ecx, 5
    mov esi, OFFSET Array3
    mov eax, 0
; print
L2: mov ax, [esi]
    call WriteHex
    call Crlf
    add esi, 2
    loop L2

    ;exit
main ENDP

END main
```



4. [Loops] Write a program to compute the sum of first *n* even integers of the series: *Sum = 2 + 4 + 6 + 8 ...* 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 bxe 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 = 7*.

```
.data
str1 BYTE "Please enter n: ", 0
sum DWORD 0

.code
main proc
    ; a.    Prompt user for integer n
    mov edx, OFFSET str1
    call WriteString

    ; b.    Read the value of n from user input
    call ReadInt
    mov ecx, eax
    mov eax, 2


    ; c.    Calculate Sum
L1: add sum, eax
    add eax, 2
    loop L1

    ; d.    Print Sum on screen.
    mov eax, sum
    call WriteInt

exit
main ENDP
```