COAL Fall 2018 Midterm # 1 Solution

Time Allowed: 60 minutes. **Maximum Points**: 30 points

==========

Q No. 1 Briefly answer each of the following: [6 x 2 = 12 points]

(i) Explain why memory access takes more machine cycles than register access?

Register are located near the CPU (access delay ~ 1nsec) and memory (RAM) is output the processor needing ~ 70-150nsecs.

(ii) What are the basic steps in the instruction execution cycle?

Fetch, decode, read operands from memory (if needed), execute, save result

(iii) After a program has been loaded into memory, how does it begin execution?

The OS start executing instructions in a specified address in the code segment.

(iv) How do you differentiate a data label from a code label?

Data label points to an offset in the data segment. Code labels do the same in code segment. Both facilitate referring memory location by names and save tedious work while coding in assembly.

(v) How address, data, and control busses are used during the instruction execution?

They are used to access memory (data operands or instructions). Address bus generates a 20 bit address and control bus facilitate reading or writing control. Data is transferred to/from memory on data bus.

(vi) Give one example instruction for each of the following addressing modes:

a. Register indirect

MOV EAX, [EBX]; EBX contains a valid non-zero memory address

b. Base indexed

MOV EAX, array[ESI] ; array db 100 dup (?) and ESI provides the index(offset)

Q No. 2 (i) Give the contents of the status flags C, S and Z and the content of destination register after the execution of each of the following sequence of instructions:

```
x1  sword 8F7AH, 7AF8H                                   [02 points]
x2  word 0000H

a.    MOV   AX, x1
      ADD   AX, [x1+2]   AX=0A72h    C = _0__  S = _0___ Z = _0___
b.    MOV   ESI, OFFSET x2
      Mov   BX, [ESI]
      SUB   BX, 1        BX=FFFFh    C = __1_  S = _1___ Z = _0___
```

(ii) Consider the following data definition directives [2 + 4 = 6 points]

```
.DATA

      . . .  (some declarations not shown)

var1  BYTE  2 DUP(41h,42h,43h),3 DUP(?)
      WORD  2 DUP(12h,13h,14h)
var2  DWORD 1234h,5678h
var3  QWORD 0A987654321h
```

a) Assign proper physical addresses to each byte of the word array stored in the above data segment (Assume DS= 2CAEh and the first element of var1 at offset 2366h).

*20 bit address = 2CAE**0** + 2366h = 2EE46h.*
Word array start @ 2EE4F (skipping byte array's 9 elements). 12 00 13 00 14 00 12 00 13 00 14 00

b) Consider the above data segment. Give the content of the destination register after the execution of the following instructions:

```
MOV   EAX, DWORD PTR [var3+2]      EAX= 00A98765h
ADD   AH, TYPE var1                AH = 87 + 1 = 88h
XCHG  AH, AL                       ; EAX: __00A96588h_____

MOV ESI, OFFSET var1 + 0Ch         ESI = 0Ch (decimal 12)
MOV EAX, LENGTHOF var2             EAX = 2
ADD EAX, [ESI+4]                   ; EAX: __14001302h_______
```

Q. No. 3 (i) Write assembly language code that directly exchanges respective elements of two word sized arrays X1 and X2 having 20 elements each. Your code should not use a third array. [05 points]

```
.data
x1 word 20 dup (10)
x2 word 20 dup (20)
.code
MOV ECX, 100
MOV ESI, 0; x1[100] is invalid x1[0]..x1[99] are 100 elements. Cannot use ECX.
L1:
  MOV AX, x1[ESI*Type x1]
  XCHG AX, x2[ESI*Type x1]
  MOV x2[ESI*Type x1], AX
  INC ESI ; is Type directive not used need to replace this with ADD ESI, 2.
LOOP L1
```

(ii) Write an assembly language program that declares three integer arrays containing 100 elements each. Consider *bArray* as a byte array, *wArray* as a word array and *dArray* as a double word array. Also, define another array xArray, where each element holds the sum of the respective elements of *bArray*, *wArray*, and *dArray*, as follows:

xArray[i] = bArray[i] + wArray[i] + dArray[i] [05 points]

```
.data
bArray byte 100 dup (10h)
wArray word 100 dup (FF20h)
dArray dword 100 dup (FFFFFF30h)
xArray  dword 100 dup (?)
.code
Count = 100
MOV ECX, count
MOV ESI, 0;
L1:
   MOVZX AX, bArray[ESI*Type bArray] ; make byte element word
   ADD AX, wArray[ESI*Type wArray]
   MOVZX EDX, AX                     ; make word element double word
   ADD EDX, dArray[ESI*Type dArray]
   MOV xArray[ESI*Type xArray], EDX  ; sum of each element of different length
   INC ESI
LOOP L1
```

STAY BRIGHT