16.317: Microprocessor-Based Systems I

Fall 2012

Exam 2 Solution

1. (20 points, 5 points per part) ***Multiple choice***

For each of the multiple choice questions below, clearly indicate your response by circling or underlining the single choice you think best answers the question.

a. Which of the following statements most accurately describes the role of the LDTR in 80386 protected mode?

i. The LDTR directly provides the base address and limit for the global descriptor table.

ii. The LDTR directly provides the base address and limit for the current local descriptor table.

iii. The LDTR is a selector that points to a descriptor in the global descriptor table. This descriptor directly provides the base address and limit for the current local descriptor table.

iv. The LDTR is a selector that points to a descriptor in the global descriptor table. This descriptor directly provides the base address and limit for the current data segment.

v. The LDTR is a selector that points to a descriptor in the current local descriptor table. This descriptor directly provides the base address and limit for the current data segment.

b. Say you have a high-level program containing an array of `short int` values, `X[30]`, and an integer variable, `i`. If the base address of this array is 31700000H and the value of `i` is 2, what is the address of the array element `X[i]`? Assume `short int` values hold 16 bits.

i. 31700000H

ii. 31700002H ***2/5 points—didn't multiply i by size of each element***

iii. 31700004H

iv. 31700008H ***2/5 points—people who chose this did the right operation but seemingly ignored that short ints hold 2 bytes, not 4 bytes***

1 (cont.)

c. Given the conditional statement below:

```
if (AX == BX)
   CX++;
else
   CX--;
```

Which of the following assembly sequences correctly implements this conditional statement?

i.

```
     CMP  AX, BX
     JE   L1
     DEC  CX
     JMP  L2
L1:  INC  CX
L2:                ; End of statement
```

ii.

```
     CMP  AX, BX
     JNE  L1
     DEC  CX
     JMP  L2
L1:  INC  CX
L2:                ; End of statement
```

iii.

```
     CMP  AX, BX
     JE   L1
     DEC  CX
L1:  INC  CX
L2:                ; End of statement
```

iv.

```
     CMP  AX, BX
     JNE  L1
     DEC  CX
L1:  INC  CX
L2:                ; End of statement
```

1 (cont.)

d. Which of the following pieces of information related to function calls are stored on the stack?

A. The function's return address

B. The starting address of the function

C. Function arguments (i.e., values passed to the function)

D. Local variables (i.e., variables declared inside the function)

E. The function's name

i. A and C ***2/5 points***

ii. A, B, and C ***2/5 points***

iii. ***<u>A, C, and D</u>***

iv. B, C, and D ***2/5 points***

v. All of the above (A, B, C, D, and E)

2. (40 points) ***Protected mode memory accesses***

Assume the 80386 is running in protected mode with the state given below. Note that each memory location shown contains a descriptor for a particular segment. Also, please note that you cannot assume the memory range shown contains the entire GDT and LDT.

GDTR = 117201500050
LDTR = 0010 ***(typo in original exam)***
LDTR cache: base = 11720128
LDTR cache: limit = 0027

DS = 0007
ES = 001F
SS = 0003
ESI = 00001202
EBP = 00000FC4

Memory	Address
Base = 030010F0 Limit = 020F	11720120
Base = 0FEE0110 **Limit = 0FEC**	**11720128** ***DS desc.***
Base = A0331010 Limit = 0027	11720130
Base = FE002200 Limit = FFFF	11720138
Base = 011B1002 **Limit = 0AFF**	**11720140** ***ES desc.***

Memory	Address
Base = AC000000 Limit = 0317	11720148
Base = 01610200 **Limit = 03F7**	**11720150** ***SS desc.***
Base = 11620128 Limit = FFFF	11720158
Base = 11720128 Limit = 0027	11720160
Base = 11720120 Limit = 0007	11720168

What physical address does each of the following instructions access?

a. `MOVSX  DX, BYTE PTR SS:[BP-4]`

Solution: *To find a segment base address, look first at its selector—in this case, SS:*

*SS = 0003H = **0000 0000 0000 0011**$_2$ → index = 0, TI = 0 (global), RPL = 3*

Since the GDT starts at address 11720150, the descriptor at 11720150 (which has index 0 within the GDT) describes the stack segment. So, the physical address being accessed is:

*Seg. base + EA = 01610200H + (0FC4-4) = **016111C0H***

b. `ADD  AX, [SI]`

Solution: *In this problem, the segment we need is DS, so we break down that selector:*

*DS = 0007H = **0000 0000 0000 0111**$_2$ → index = 0, TI = 1 (local), RPL = 3*

Since the LDT starts at address 11720128, the descriptor at 11720128 (which has index 0 within the LDT) describes this segment. So, the physical address being accessed is:

*Seg. base + EA = 0FEE0110H + SI = 0FEE0110H + 1202H = **0FEE1312H***

2 (continued)

c. `ROR  WORD PTR ES:[SI+10H], 7`

Solution: *In this problem, the segment we need is ES, so we break down that selector:*

ES = 001FH = ***0000 0000 0001 1111***$_2$ → *index = 3, TI = 1 (local), RPL = 3*

Since the LDT starts at address 11720128, the descriptor at 11720140 (which has index 3 within the LDT) describes this segment. So, the physical address being accessed is:

Seg. base + EA = 011B1002H + (SI + 10) = 011B1002H + 1212H = ***011B2214H***

3. (40 points, 20 points per part) ***Reading PIC assembly language***

Show the result of each PIC 16F684 instruction in the sequences below. Be sure to show not only the state of updated registers, but also the carry (C) and zero (Z) bits.

a.
```
cblock 0x20
    x
endc
```

clrf	x	**x = 0x00**
movlw	0x72	**W = 0x72**
addlw	0xF0	**W = W + 0xF0 = 0x72 + 0xF0 = 0x162** **Therefore, W = 0x62, C = 1, Z = 0 (result not zero)**
movwf	x	**x = W = 0x62**
comf	x, W	**W = ~x = ~0x62 = ~(0110 0010_2) = 1001 1101_2 = 0x9D** **Z = 0**
bsf	x, 4	**Set bit 4 of x** **x originally is 0x62 = 0110 0010_2 (bit 4 is underlined)** **x changes to 0111 0010_2 = 0x72**
incf	x, F	**x = x + 1 = 0x72 + 1 = 0x73**
sublw	0x02	**W = 0x02 – W = 0x02 – 0x9D = 0x65** **C = 1, Z = 0**
swapf	x, F	**Swap nibbles of x** **x originally is 0x73 → changes to 0x37**
addwf	x, W	**W = x + W = 0x37 + 0x65 = 0x9C** **C = Z = 0**

3 (cont.)

b.
```
    cblock 0x20
        q
    endc
```

	movlw	0x0B	**W = 0x0B**
	movwf	q	**q = W = 0x0B**
	xorlw	0x33	**W = W XOR 0x33 = 0x0B XOR 0x33 = 0x38**
	iorwf	q, W	**W = q OR W = 0x0B OR 0x38 = 0x3B**
	andlw	0x87	**W = W AND 0x87 = 0x3B AND 0x87 = 0x03**
	bsf	STATUS, C	**Set C = 1**
	rlf	q, F	**Rotate q through carry one bit to the left** **(C, q) = 1 0000 1011_2 originally** **After rotate, (C, q) = 0 0001 0111_2** **→ C = 0, q = 0x17**
	btfss	q, 7	**Check bit 7 of q and skip next instruction if set** **q = 0x17 = 0001 0111_2 → bit isn't set, so don't skip**
	goto	L1	**Unconditionally jumps to L1**
	rrf	q, W	
L1:	andwf	q, F	**q = q AND W = 0x17 AND 0x03 = 0x03**