16.317: Microprocessor-Based Systems I

Summer 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. Assume SS = 3000H and SP = F018H before the 80386 executes the following instructions:

```
PUSH    AX
PUSH    CX
PUSH    EDX
PUSH    ESI
```

What is the physical address of the top of the stack after executing the instructions above?

i. 30000H

ii. 3F00CH

iii. 3F010H

iv. 3F018H

v. 3F024H

b. You are given the incomplete loop below:

```
        MOV  CX, 000AH
        MOV  SI, FFFFH
L:      INC  SI
        MOV  AX, [SI]
        CMP  AX, 00H
        _____________
```

Choose one of the instructions below to fill in the blank so that the loop above will exit if (a) 10 iterations have been completed, or (b) the `MOV` instruction loads a non-zero byte from memory:

i. JMP L

ii. LOOP L

iii. LOOPE L

iv. LOOPNE L

v. IMUL AX

1 (cont.)

c. Assuming A, B, C, and D are all signed integers, what compound condition does the following instruction sequence test?

```
MOV     AX, A
ADD     AX, B
CMP     AX, C
SETLE   BL
MOV     AX, C
CMP     AX, D
SETG    BH
OR      BL, BH
```

i. `(A <= C) || (C > D)`

ii. `(B <= C) || (C > D)`

iii. `(A + B <= C) || (C >= D)`

iv. `(A <= B + C) || (C > D)`

v. ***`(A + B <= C) || (C > D)`***

d. Which of the following statements accurately reflect your opinion(s)? Circle all that apply (but please don't waste too much time on this question)!

i. "I still don't know what the difference between a selector and a descriptor is."

ii. "I'm not sure Dr. Geiger knows what the difference between a selector and a descriptor is."

iii. "Would someone please explain why we're not just programming in C?"

iv. "Is the semester over yet?"

Any of the above are "correct."

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.

GDTR = 001631A00038
LDTR = 0010
LDTR cache: base = 00163180
LDTR cache: limit = 001F

DS = 000E
ES = 001B
EDI = 0000444A
EBX = 0000F000

Memory	Address
Base = 030010F0 Limit = 020F	00163170
Base = 12300020 Limit = 0007	00163178
Base = A0331010 Limit = 0027	00163180
Base = FE002200 **Limit = FFFF**	**00163188** ***DS desc.***
Base = 12340000 Limit = 00FF	00163190

Memory	Address
Base = AC000000 Limit = 0317	00163198
Base = 01610200 Limit = 03F7	001631A0
Base = 00163170 Limit = 0027	001631A8
Base = 00163180 Limit = 001F	001631B0
Base = 05000120 **Limit = C00F**	**001631B8** ***ES desc.***

What address does each of the following instructions access?

a. `MOV DX, [40H]`

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

DS = 000EH = ***0000 0000 0000 1110***$_2$ → *index = 1, TI = 1 (local), RPL = 2*

Since the LDT starts at address 00163180, the descriptor at 00163188 (which has index 1 within the LDT) describes the data segment. So, the physical address being accessed is:

Seg. base + EA = FE002200H + 40H = ***FE002240H***

b. `XOR ES:[DI], CX`

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

ES = 001BH = ***0000 0000 0001 1011***$_2$ → *index = 3, TI = 0 (global), RPL = 3*

Since the GDT starts at address 001631A0, the descriptor at 001631B8 (which has index 3 within the GDT) describes this segment. So, the physical address being accessed is:

Seg. base + EA = 05000120H + DI = 05000120 H+ 444AH = ***0500456AH***

c. `BSF AX, WORD PTR [BX+100H]`

Solution: *Like part (a), this problem accesses the data segment, and therefore uses the same segment base address. The physical address being accessed is therefore:*

Seg. base + EA = FE002200H + BX + 100H
= FE002200H + F000H + 100H = ***FE011300H***

3. (40 points) ***Assembly language***

For each instruction sequence shown below, list all changed registers, memory locations, and/or flags, as well as their new values.

a. Initial state:

EAX: 0000ABC0H
EBX: 000012ACH
ECX: 00000020H
EDX: 00000000H
ESI: 00000012H
EDI: 00000200H
DS: 4130H
FLAGS: 00H

Address				
41300H	00	F0	08	00
41304H	10	10	00	FF
41308H	30	00	19	91
4130CH	20	40	60	80
41310H	AA	AA	AB	0F
41314H	00	16	55	55
41318H	17	03	7C	EE
4131CH	AA	55	42	D2
41320H	86	75	30	90

Instructions:

```
BTC       BX, 6
```

BX = 12ACH = 0001 0010 1010 1100 → bit 6 = 0 (will be changed to 1)
Instruction result: ***CF = bit 6 = 0; BX = 12ECH***

```
SETNC DL
```

DL = FFH if CF == 0; 00H otherwise
Instruction result: ***DL = FFH****, since BTC set CF = 0*

```
BSR       AX, [SI]
```

AX = position of first non-zero bit in (DS:SI)
(DS:SI) = (41312H) = 0FABH = 0000 1111 1010 1011
→ first non-zero bit (scanning MSB to LSB) is bit 11
Instruction result: ***AX = 000BH, ZF = 1 (value isn't 0)***

```
AND       AH, DL
```

Instruction result: AH = AH & DL = 00H & FFH = ***00H***

```
SAHF
```

Instruction result: FLAGS = AH = ***00H***

3 (cont.)
b. Initial state:

EAX: 00003170H
EBX: 0000315CH
ECX: 000031C5H
EDX: 00000000H
ESI: 00000012H
EDI: 0000001CH
DS: 4130H
FLAGS: 00H

Address				
41300H	00	F0	08	00
41304H	10	10	00	FF
41308H	30	00	19	91
4130CH	20	40	60	80
41310H	AA	AA	AB	0F
41314H	00	16	55	55
41318H	17	03	7C	EE
4131CH	AA	55	42	D2
41320H	86	75	30	90

Notes: For CMP instructions, note the relationship between compared values (e.g., "AX < BX"). For jumps, indicate if the jump is taken and why (e.g., "JG not taken because AX < BX"). Only evaluate instructions that are actually executed—don't evaluate skipped instructions.

Instructions:

```
        CMP     AX, BX
            Compares 3170H to 315CH → AX > BX

        JL      L1
            Jump not taken, since AX > BX

        CMP     AX, CX
            Compares 3170H to 31C5H → AX < BX

        JL      L2
            Jump taken, since AX < BX

        INC     AX
        JMP     END
L1:     DEC     AX
        JMP     END
            Previous 4 instructions are skipped

L2:     MOV     AX, 0123H
            AX = 0123H

END:    MOV     [DI], AX
            (DS:DI) = AX = 0123H
            DS:DI = 4130:001C → address == 4131C
            Byte at 4131C = 23H
            Byte at 4131D = 01H
```