16.317: Microprocessor-Based Systems I

Fall 2012

Exam 1 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 about data storage are **false**?

A. Registers provide fast data access because most processors have a relatively small number of registers, and those registers are located close to the part of the processor that actually performs the computation.
B. Memory locations are typically referenced by name; registers are typically referenced by address.
C. In a segmented memory architecture, the total address space is divided into fixed address ranges that cannot overlap with one another.
D. Effective address computations typically involve a constant value, one or more registers, or some combination of constants and registers.
E. The older your computer gets, the more likely it is to suffer "memory lapses"—losing its product keys, forgetting where it put data, and even failing to recognize programs it's known for years.

i. A and E

ii. B and E

iii. A, D, and E

iv. ***B, C, and E***

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

1 (continued)

b. Given `AX = 0020H` and `CX = 0005H`, what is the result of the instruction: `MUL CL`?

i. `AX = 0025H`

ii. ***<u>`AX = 00A0H`</u>***

iii. `AX = 0100H`

iv. `AX = 0020H`

v. `AX = 0005H`

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

```
MOV     AX, A
CMP     AX, B
SETNE   BL
MOV     AX, C
SUB     AX, A
CMP     AX, D
SETGE   BH
AND     BL, BH
```

i. `(A == B) && (C >= D)`

ii. `(A != B) && (C >= D)`

iii. `(A == B) && (C – A >= D)`

iv. ***<u>`(A != B) && (C – A >= D)`</u>***

v. `(A != B) && (A – C >= D)`

1 (continued)

d. Given the following short code sequence:

```
        MOV  CX, 10
ST:     MOV  AX, [SI]
        ADD  SI, 2
        CMP  AX, 00EFH
        LOOPNE ST
```

Under what conditions will the loop **end** (in other words, the `LOOPNE` instruction will not return to the label `ST`)?

i. `CX > 0`

ii. `(CX > 0)` and `(AX != 00EFH)`

iii. `(CX == 0)` or `(AX != 00EFH)`

iv. ***`(CX == 0)` or `(AX == 00EFH)`***

v. `(CX == 00EFH)`

2. (40 points) ***Memory addressing***

Assume the state of the 80386DX registers are as follows:

- (CS) = 1345H
- (DS) = 3170H
- (SS) = AE01H
- (IP) = 202EH
- (ESI) = 7083F002H
- (EDI) = 102021CBH
- (EBX) = FFEE2202H
- (EBP) = 1234BEACH

Complete the table below by:

- Calculating the physical address that corresponds to each given logical address.
- Answering the alignment-related question given in the third column for each address.
  - Be sure to justify your answer in each case.

Logical address	**Physical address**	**Alignment-related question**
CS:IP	CS (shifted) + IP = 13450 + 202E = **1547EH**	If you access a word at this address, is the access aligned? Why or why not? **Yes, the access is aligned, because a word holds two bytes, and the address is divisible by 2.**
SS:BP+10H	SS (shifted) + (BP+10H) = AE010 + (BEAC+10) = AE010 + BEBC = **B9ECCH**	For what data sizes (byte, word, double word) will an access to this address be aligned? Why? **Accesses to bytes, words, and double words will all be aligned, as this address is divisible by 1, 2, and 4.**
DS:SI+ 1002H	DS (shifted) + (SI+1002H) = 31700 + (F002+1002) = 31700 + <u>0004</u> = **31704H** *<u>Remember, EA is only 16 bits</u>*	Say you could access 8 bytes of data (a quad word) at this address—would the access be aligned? Why or why not? **Access to a quad word would not be aligned, as this address is not divisible by 8.**
DS:BX+DI+ 101H	DS (shifted) + (BX+DI+101H) = 31700 + (2202+21CB+101)= 31700 + 44CE = **35BCEH**	What is the largest amount of data that can be accessed in an aligned access to this address? **A word, since the address is divisible by 2 but not by 4.**

3. (40 points, 20 points per part) ***Assembly language***

For each instruction sequence shown below, list all changed registers and/or memory locations and their new values. If memory is changed, be sure to explicitly list **all changed bytes**. Where appropriate, you should also list the state of the carry flag (CF).

a. Initial state:

EAX: 00000000H
EBX: 00000008H
ECX: 0000021EH
EDX: 0000FF00H
CF: 0
ESI: 00000004H
EDI: FFFF0000H
DS: 3170H

Address	Lo			Hi
31700H	04	00	08	00
31704H	10	10	20	20
31708H	89	01	71	31
3170CH	20	40	60	80
31710H	02	00	AB	0F
31714H	00	16	11	55
31718H	17	03	7C	EE
3171CH	FF	00	42	D2
31720H	86	75	30	90

Instructions:

```
MOVSX  EAX, WORD PTR [SI+0AH]
```

EAX = sign-extended word at DS:SI+0AH

= sign-extended word at 3170EH

= 8060H, sign-extended = FFFF8060H

```
LDS    SI, [BX]
```

SI = word at DS:BX

= word at 31708H = 0189H

DS = word at DS:BX+2

= word at 3170A = 3171H

```
BT     BYTE PTR [00H], 2
```

CF = bit 2 of byte at DS:00

= bit 2 of byte at 31710H

= bit 2 of 02H = bit 2 of 0000 0010$_2$ ***→ CF = 0***

```
ADC    AX, DX
```

AX = AX + DX + CF

= 8060H + FF00H + 0 = 7F60H

CF = carry out of last bit = 1

```
NEG    AX
```

AX = -AX = -7F60H = -(0111 1111 0110 0000$_2$***)***

= 1000 0000 1001 1111$_2$ ***+ 1 = 1000 0000 1010 0000***$_2$ ***= 80A0H***

3 (cont.)
b. Initial state:

EAX: 00000038H
EBX: 00001000H
ECX: 00000004H
EDX: 0000003CH
CF: 0
ESI: 00002000H
EDI: FFFF1000H
DS: AE00H

Address	Lo			Hi
AE000H	C0	00	02	10
AE004H	10	10	15	AA
AE008H	89	01	05	B1
AE00CH	20	40	AC	DC
AE010H	04	08	05	83
AE014H	00	16	02	06
AE018H	17	03	19	78
AE01CH	FF	00	12	24
AE020H	1E	00	20	07

Instructions:

```
MOV   AL, [00H]
```

AL = (DS:00H) = C0H

```
SAR   AL, CL
```

AL = AL >> CL (keep sign intact)
= C0 >> 4 = (1100 0000$_2$) >> 4 = 1111 1100$_2$ = FCH
CF = last bit shifted out = 0

```
RCL   AL, 3
```

AL = AL rotated left through carry by 3 bits
(CF,AL) = 0 1111 1100$_2$ originally
= 1 1110 0011$_2$ after rotate → CF = 1; AL = E3H

```
AND   AL, DL
```

AL = AL & DL = E3H & 3CH = 20H

```
BSF   BL, AL
```

Scan AL from right to left for first non-zero bit; store position in BL and set ZF = 1 if AL is non-zero, 0 otherwise

AL = 20H = 0010 0000$_2$ → first non-zero bit = bit 5

→ ZF = 1; BL = 05H