16.317: Microprocessor-Based Systems I

Summer 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 is not one of the operation types implemented by most microprocessors?

  i. Logical operations

  ii. Matrix operations

  iii. Program control operations

  iv. Data transfer operations

  v. Arithmetic operations

b. Which of the following statements about data storage are true?

  A. Registers provide long-term data storage because of their capacity, while memory provides storage locations that can always be quickly accessed.
  B. Memory locations are typically referenced by address; registers are typically referenced by name.
  C. In a segmented memory architecture, only a subset of the total address space is accessible at any given time.
  D. Data stored in registers is accessed using "immediate addressing," a term that got its name from the fact that registers are so fast they can be accessed "immediately."
  E. When a couple is engaged to be married, they often go to stores to set up wedding registers, a process that most grooms-to-be love more than life itself.

  i. A and B

  ii. B and C

  iii. C and D

  iv. A, B, and C

  v. B, C, and D

1 (cont.)

c. Given AX = 0013H, CX = 0003H, and DX = 0000H, what is the result of the instruction: DIV CX?

i. AX = 0013H, DX = 0000H

ii. AX = 0004H, DX = 0001H

iii. AX = 0001H, DX = 0004H

iv. ***<u>AX = 0006H, DX = 0001H</u>***

v. AX = 0001H, DX = 0006H

d. Given AX = 00FFH, BX = 0008H, and DX = 0000H, what is the result of the instruction IMUL BL?
<u>NOTE:</u> This problem originally contained a typo, with the instruction written as "IMUL BX". In that case, choice (ii) would be the correct answer—that instruction would normally change DX and AX, but DX remains 0000H

i. AX = 00F8H

ii. AX = 07F8H

iii. ***<u>AX = FFF8H</u>***

iv. AX = 00F8H, DX = 00FFH

v. Pigs fly out of every window on the fourth floor of Ball Hall.

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

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

- (CS) = 1100H
- (DS) = 5020H
- (GS) = 4097H
- (IP) = FEDCH
- (ESI) = 70838E97H
- (EDI) = 102030AAH
- (EBX) = FFEE2245H
- (EBP) = 12345EACH

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 = 11000H + FEDCH = **20EDCH**	If you access a double word at this address, is the access aligned? **Yes—the address is divisible by 4.**
GS:DI+45H	GS (shifted) + DI + 45H = 40970H + 30AAH + 45H = **43A5FH**	For what data sizes (byte, word, double word) will an access to this address be aligned? **Only a byte—the address is divisible by neither 2 nor 4, so word and double word accesses are not aligned.**
DS:BX+DI	DS (shifted) + BX + DI = 50200H + 2245H + 30AAH = **554EFH**	What is the smallest value you could add to this address to make it an aligned address for double word accesses? **Adding 1 to this address would make it divisible by 4 (554F0H), which would make it aligned for double word accesses.**
GS:BX+SI+ 102H	GS (shifted) + BX + SI + 102H = 40970H + 2245H + 8E97H + 102H = **4BB4EH**	What is the largest amount of data that can be accessed in an aligned access to this address? **The address is divisible by 2, but not 4, so a word is the largest amount of data that can be accessed in an aligned access to this address.**

3. (40 points, 20 points per part) ***<u>Assembly language</u>***

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

a. <u>Initial state:</u>

EAX: 00000000H
EBX: 0000000AH
ECX: 0000021EH
EDX: 0000FF00H
CF: 0
ESI: 00000008H
EDI: FFFF0000H
DS: 3170H
ES: 3171H

Address				
31700H	04	00	08	00
31704H	10	10	20	20
31708H	89	01	19	91
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

<u>Instructions:</u>

```
MOVSX  EAX, BYTE PTR [SI+03H]
```

EAX = sign-extended byte at DS:SI+03H
= sign-extended byte at 31700H + 0008H + 03H
= sign-extended byte at 3170BH = 91H, sign-extended to 32b
= <u>FFFFFF91H</u>

```
LES    SI, [10H]
```

SI = word at DS:10H = word at 31710H = <u>0002H</u>
ES = word at DS:12H = word at 31712H = <u>0FABH</u>

```
SUB    AX, DX
```

AX = AX – DX = FF91H – FF00H = <u>0091H</u>

```
NEG    WORD PTR [BX+12H]
```

Negate word at DS:BX+12H = word at 31700 + 000A + 12 =
word at 3171CH = 00FFH
-(00FFH) = FF01H → <u>Byte at 3171CH = 01H; byte at 3171DH = FFH</u>

```
ADD    AL, CL
```

AL = AL + CL = 91H + 1EH = <u>AFH</u>

3 (cont.)

b. Initial state:

EAX: 00000000H
EBX: 00001000H
ECX: 00000002H
EDX: 000000F0H
CF: 0
ESI: 00002000H
EDI: FFFF1000H
DS: AC00H

Address				
AE000H	04	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, [SI+0BH]
```

AL ***= byte at DS:SI+0BH = byte at AC000 + 2000 + 0B***
= byte at AE00B = B1H

```
SAR   AL, CL
```

AL ***= AL >> CL (keep sign intact) = B1H >> 2***
= $1011\ 0001_2$ >> 2 = $1110\ 1100_2$ = ECH
CF ***= last bit shifted out = 0***

```
ROL   AL, 5
```

AL ***= AL rotated left by 5***
= $1110\ 1100_2$ rotated left by 5 = $1001\ 1101_2$ = 9DH
CF ***= last bit shifted out = 1***

```
AND   AL, DL
```

AL = AL & DL = 9DH & F0H = 90H

```
MOV   [DI+BX+1EH], AL
```

Byte at DS:DI+BX+1EH = AL = 90H
Address = AC000 + 1000 + 1000 + 1E = AE01EH
Byte at AE01EH = 90H