16.317: Microprocessor Systems Design I

Spring 2013

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. If the carry flag (`CF`) is set to 0, which of the following instructions will always set `CF = 1`?

A. `STC`
B. `CLC`
C. `CMC`
D. `LAHF`
E. `SAHF`

i. A and C

ii. B and C

iii. A, C, and D

iv. A, C, and E

v. B, C, and E

b. If `AH = 0FH`, what is the result of the instruction `BTC AH, 7`?

i. `CF = 0, AH` is unchanged

ii. `CF = 1, AH` is unchanged

iii. `CF = 0, AH = 4FH`

iv. CF = 0, AH = 8FH

v. `CF = 1, AH = 8FH`

1 (cont.)

c. Which of the following statements the compare instruction `CMP AX, BX` are true?

A. The instruction subtracts `AX – BX` and stores the result in `AX`.

B. The instruction subtracts `AX – BX` but does not store the result anywhere.

C. If `AX` and `BX` are equal, the zero flag (`ZF`) is set to 1.

D. If `AX` is less than `BX`, then the sign flag (`SF`) will always be 1.

E. If `AX` is less than `BX`, then the sign flag (`SF`) will always be 0.

i. Only A

ii. Only B

iii. <u>B and C</u>

iv. B, C, and D

v. A, C, and E

d. If `AX = 0FF0H`, which of the following choices correctly shows the results of performing the two bit scan instructions (`BSF` and `BSR`) on this register?

i.
```
BSF DX, AX      → ZF = 0, DX unchanged
BSR DX, AX      → ZF = 0, DX unchanged
```

ii.
```
BSF DX, AX      → ZF = 1, DX unchanged
BSR DX, AX      → ZF = 1, DX unchanged
```

iii.
```
BSF DX, AX      → ZF = 0, DX = 0004H
BSR DX, AX      → ZF = 0, DX = 000BH
```

iv.
```
BSF DX, AX      → ZF = 1, DX = 0004H
BSR DX, AX      → ZF = 1, DX = 000BH
```

v.
```
BSF DX, AX      → ZF = 1, DX = 000BH
BSR DX, AX      → ZF = 1, DX = 0004H
```

2. (30 points) ***Data transfers and memory addressing***

For each data transfer instruction shown below, list all changed registers and their final values. Also, indicate if each instruction performs an aligned memory access, an unaligned memory access, or no memory access at all.

Initial state:

EAX: 00000000H
EBX: 00000008H
ECX: 0000021EH
EDX: 0000FF00H
ESI: 0000F000H
EDI: 00000101H
DS: 2201H
ES: 2000H

Address	Lo			Hi
22000H	20	13	80	40
22004H	FF	AF	BC	13
22008H	99	88	77	66
2200CH	A8	B1	F0	43
22010H	78	D6	32	33
22014H	34	35	12	16
22018H	93	03	7C	EF

Instructions:

`MOV AX, [BX+01H]` Aligned? Yes ***No*** Not a memory access

EA = BX + 01H = 0009H; SBA = 22010H (access DS)
PA = SBA + EA = 22010H + 0009H = 22019H
AX = word at 22019H = 7C03H

`MOVSX EBX, BYTE PTR [0001H]` Aligned? ***Yes*** No Not a memory access

EA = 0001H; SBA = 22010H (access DS)
PA = SBA + EA = 22010H + 0001H = 22011H
EBX = sign-extended byte at 22011H = FFFFFFD6H

`MOVZX ECX, WORD PTR ES:[SI+3004H]` Aligned? ***Yes*** No Not a memory access

EA = SI + 3004H = F000H + 3004H = ~~1~~2004H (EA is only 16 bits)
SBA = 20000H (access ES)
PA = SBA + EA = 20000H + 2004H = 22004H
ECX = zero-extended word at 22004H = 0000AFFFH

`LEA DI, [SI+1A2BH]` Aligned? Yes No ***Not a memory access***

EA = SI + 1A2BH = F000H + 1A2BH = ~~1~~0A2BH (EA is only 16 bits)
DI = EA = 0A2BH

`LDS EDX, ES:[2006H]` Aligned? ***Yes No*** Not a memory access

EA = 2006H; SBA = 20000H (access ES)
PA = SBA + EA = 20000H + 2006H = 22006H
EDX = double-word at 22006H = 889913BCH (not aligned)
DS = word at 2200AH = 6677H (aligned)

3. (25 points) ***Arithmetic instructions***

For each instruction in the 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).

Initial state:

EAX: 00000047H
EBX: 000000C5H
ECX: 0000021EH
EDX: 0000FFFEH
CF: 1
ESI: 00000004H
DS: 3170H

Address	Lo			Hi
31700H	04	00	08	00
31704H	83	00	01	01
31708H	05	01	71	31
3170CH	20	40	60	80
31710H	02	00	AB	0F
31714H	00	16	11	55

Instructions:

```
ADC    AX, [SI]
```

EA = SI = 0004H; SBA = 31700 (access DS) → PA = 31704H
AX = AX + word at 31704H + CF
= 0047H + 0083H + 1 = 00CBH
CF = 0

```
SUB    AX, BX
```

AX = AX – BX = 00CBH – 00C5H = 0006H
CF = 0

```
NEG    BX
```

BX = -BX = -00C5H = -(0000 0000 1100 0101_2)
= 1111 1111 0011 1010_2 + 1 = 1111 1111 0011 1011_2 = FF3BH

```
IMUL   DL
```

AX = AL * DL = 06H * FEH = 6 * -2 = -12
= -(0000 0000 0000 1100_2) = 1111 1111 1111 0011_2 + 1
= 1111 1111 1111 0100_2 = FFF4H

```
INC    AH
```

AH = AH + 1 = FF + 1 = 00H

4. (25 points) ***Logical instructions***

For each instruction in the 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).

Initial state:
EAX: 000000F0H
EBX: 00001000H
ECX: 00000003H
EDX: 0000F63CH
CF: 0
DS: 1000H

Address	Lo			Hi
10000H	C0	00	02	10
10004H	10	10	15	5A
10008H	89	01	05	B1
1000CH	20	40	AC	DC
10010H	04	08	05	83

Instructions:

```
AND   AL, [07H]
```

PA = SBA + EA = 10000H + 07H = 10007H
AL = AL AND (byte at 10007) = F0H AND 5AH = 50H

```
XOR   AL, DH
```

AL = AL XOR DH = 50H XOR F6H = A6H

```
SAR   AL, 3
```

AL = AL >> 3 (arithmetic shift)
= A6H >> 3 = 1010 0111$_2$ >> 3 = 1111 0100$_2$ = F4H
CF = last bit shifted out = 1

```
ROL   AL, 4
```

AL = AL rotated left 4 bits
= F4H rotated left 4 bits = 1111 0100$_2$ rotated left 4 bits
= 0100 1111$_2$ = 4FH
CF = copy of last bit rotated out = 1

```
RCR   AL, 2
```

AL = AL rotated right through carry 2 bits
(AL,CF) = 0100 1111 1$_2$ rotated right 2 bits
= 1101 0011 1$_2$
AL = 1101 0011$_2$ = D3H, CF = 1