EECE.3170: Microprocessor Systems Design I

Spring 2016

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. Given `AL = C3h, BL = 1Bh,` and `CF = 1`, what is the final result of the instruction `ADC AL, BL`?

i. `AL = C3h, CF = 0`

ii. `AL = DEh, CF = 0`

iii. `AL = DEh, CF = 1`

iv. ***AL = DFh, CF = 0***

v. `AL = DFh, CF = 1`

b. Given `AL = 96h` and `CF = 0`, what is the final result of the instruction `RCR AL, 3`?

i. `AL = 12h, CF = 1`

ii. ***AL = 92h, CF = 1***

iii. `AL = B2h, CF = 0`

iv. `AL = D2h, CF = 1`

v. `AL = F2h, CF = 1`

1 (continued)

c. If `AX = 3A4Bh`, which of the following instructions will set `CF = 1`?

A. `BT   AX, 3`
B. `BTR  AX, 12`
C. `BTS  AX, 1`
D. `BTC  AX, 6`

i. Only A

ii. Only B

iii. A and D

iv. B and C

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

d. If `AX = 10F0H`, 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 = 1, DX = 0004h`***
`BSR DX, AX` → `ZF = 1, DX = 000Ch`

ii. `BSF DX, AX` → `ZF = 1, DX = 0004h`
`BSR DX, AX` → `ZF = 1, DX = 0012h`

iii. `BSF DX, AX` → `ZF = 0, DX = 0004h`
`BSR DX, AX` → `ZF = 0, DX = 000Ch`

iv. `BSF DX, AX` → `ZF = 1, DX = 0003h`
`BSR DX, AX` → `ZF = 1, DX = 0005h`

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

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

For each data transfer instruction in the sequence shown below, list all changed registers and/or memory locations and their final values. If memory is changed, be sure to explicitly list **all changed bytes**. Also, indicate if each instruction performs an aligned memory access, an unaligned memory access, or no memory access at all.

Constant values in address calculations will be zero-extended out to 32 bits if necessary.

Initial state:

EAX: 00067340h
EBX: 11016FEBh
ECX: FFFFFFFDh
EDX: DEADBEEFh
ESI: 00067330h
EDI: 00000003h

Address	Lo			Hi
67330h	01	08	AD	C0
67334h	02	62	38	73
67338h	CE	12	60	EB
6733Ch	B0	55	99	DD
67340h	F4	88	22	0A

Instructions:

`MOVZX DX, BYTE PTR [ESI+03h]` Aligned? [✓ Yes] No Not a memory access

```
Address = ESI + 03h = 67330h + 03h = 67333h
DX = zero-extended byte at 67333h = 00C0h
```

`MOV   EBX, 00067334h` Aligned? Yes No [✓ Not a memory access]

```
EBX = 00067334h (simply copy constant to register)
```

`XCHG  AX, [ESI+2*EDI]` Aligned? [✓ Yes] No Not a memory access

```
Address = ESI + 2 * EDI = 67330h + 2 * 03h = 67336h
AX = word at 67336h = 7338h
Word at 67336h = original contents of AX = 7340h
       (byte at 67336h = 40h, byte at 67337h = 73h)
```

`LEA   EDI, [EBX+ECX]` Aligned? Yes No [✓ Not a memory access]

```
EDI = EBX + ECX = 00067334 + FFFFFFFDh = 00067331h
       (FFFFFFFDh = -3)
```

`MOVSX ECX, WORD PTR [EAX]` Aligned? [✓ Yes] No Not a memory access

```
Address = EAX = 67338h (AX changed to 7338 in XCHG instruction)
ECX = sign-extended word at 67338h = 000012CEh
```

3. (30 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).

Constant values in address calculations will be zero-extended out to 32 bits if necessary.

Initial state:
EAX: 0000A30Fh
EBX: 00000700h
ECX: 000006FEh
EDX: 52DC7AFBh
CF: 0
ESI: 00011100h

Address	Lo			Hi
11800h	1B	79	02	10
11804h	00	44	15	5A
11808h	89	F6	A2	B1

Instructions:

```
ADD    [ESI+EBX], DX
```

Address = ESI + EBX = 00011100h + 00000700h = 11800h
Word at 11800h = Word at 11800h + DX = 791Bh + 7AFBh = F416h
(byte at 11800h = 16h, byte at 11801h = F4h)
CF = 0

```
SUB    EAX, ECX
```

EAX = EAX – ECX = 0000A30Fh – 000006FEh = 00009C11h, CF = 0

```
DEC    WORD PTR [ESI+EBX+4]
```

Address = ESI + EBX + 4 = 11100h + 700h + 4 = 11804h
Word at 11804h = Word at 11804h – 1 = 4400h – 1 = 43FFh
(byte at 11804h = FFh, byte at 11805h = 43h)

```
NEG    BYTE PTR [ESI+EBX+000Ah]
```

Address = ESI + EBX + 000Ah = 11100h + 700h + 000Ah = 1180Ah
Byte at 1180Ah = -(Byte at 1180Ah) = -A2h = -(1010 0010$_2$***)***
= 0101 1101$_2$*** + 1 = 0101 1110***$_2$*** = 5Eh***

`IDIV   CH` ***(NOTE: I unintentionally made this problem much harder than I meant to—in fact, the divide operation actually causes an exception because its result doesn't fit in one byte.)***

AL = AX / CH = 9C11h / 06h = -25583 / 6 = ~~10~~A7h (overflow)
AH = AX % CH (remainder) = -25583 % 6 = 05h

4. (20 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: 00000F6Dh
EBX: 0000FE9Ah
ECX: 00000006h
EDX: 000053DCh
CF: 0

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

Instructions:

```
SAR   AX, 3
```

AX = AX >> 3 (keep sign intact) = 0F6Dh >> 3
= 0000 1111 0110 1$\underline{1}$01$_2$ >> 3 = 0000 0001 1110 1101$_2$
= $\underline{\text{01EDh}}$, $\underline{\text{CF = 1}}$ (bit shifted into CF underlined above)

```
AND   AL, BL
```

AL = AL AND BL = EDh AND 9Ah = $\underline{\text{88h}}$

```
SHL   AL, CL
```

AL = AL << CL = 88h << 06h
= 1000 1$\underline{0}$00$_2$ << 6 = 0000 0000$_2$ = $\underline{\text{00h}}$, $\underline{\text{CF = 0}}$

```
NOT   DX
```

DX = ~DX = ~53DCh = ~(0101 0011 1101 1100$_2$)
= 1010 1100 0010 0011$_2$ = $\underline{\text{AC23h}}$

```
OR    AX, DX
```

AX = AX OR DX = 0100h OR AC23h = $\underline{\text{AD23h}}$

5. (10 points) ***Extra credit***

Complete the code snippet below by writing the appropriate x86 instruction into each of the blank spaces. The purpose of each instruction is described in a comment to the right of the blank.

```
MOV  AX, [4370h]                        ; Copy a word from
SHL  AX, 16                             ;  address 4370h into
(other instructions possible)           ;  the upper 16 bits of
                                        ;  EAX, using two
                                        ;  instructions. The
                                        ;  lower 16 bits of EAX
                                        ;  should be set to 0

SAR  BX, 7 -or- SHR BX, 7               ; Divide the value in BX
                                        ;  by 128 and place the
                                        ;  result in BX, using a
                                        ;  single instruction

BTS  ECX, 31                            ; Copy the sign bit of
                                        ;  the value in ECX
                                        ;  into the carry flag,
                                        ;  then ensure ECX holds
                                        ;  a negative value

RCR  DX, 1                              ; Move the value in the
                                        ;  carry flag into the
                                        ;  most significant bit
                                        ;  of DX (you may change
                                        ;  other bits of DX)

BSR  CL, BL                             ; Find the position of
                                        ;  the most significant
                                        ;  nonzero bit in BL,
                                        ;  and store that
                                        ;  position in CL

AND  EAX, FF0000FFh                     ; Clear the middle 16
                                        ;  bits of EAX without
                                        ;  changing any other
                                        ;  bits in 1 instruction

LEA  ESI, [EAX + EBX + 5]               ; Store the sum of EAX,
                                        ;  EBX, and the constant
                                        ;  value 5 in ESI, using
                                        ;  1 instruction

SHL  BYTE PTR [ESI], 4                  ; Multiply the byte at
                                        ;  the address stored in
                                        ;  ESI by 16, using 1
                                        ;  instruction

BT   EAX, 0                             ; Copy the rightmost bit
                                        ;  of EAX into CF
```