16.317: Microprocessor Systems Design I

Fall 2015

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 = A1h` and `CF = 1`, what is the final result of the instruction `ROR AL, 2`?

i. `AL = 43h, CF = 0`

ii. AL = 68h, CF = 0

iii. `AL = 86h, CF = 0`

iv. `AL = E8h, CF = 0`

v. `AL = 28h, CF = 0`

b. Given `AL = D9h` and `CF = 0`, what is the final result of the instruction `RCL AL, 4`?

i. `AL = 0D, CF = 1`

ii. `AL = 2D, CF = 1`

iii. `AL = 90, CF = 1`

iv. AL = 96, CF = 1

v. `AL = 9D, CF = 1`

1 (continued)

c. If `AX = 080Ch`, which of the following instructions will set `CF = 0`?

A. `BT  AX, 0`
B. `BTR  AX, 3`
C. `BTS  AX, 11`
D. `BTC  AX, 4`

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 = 03C0H`, 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 = 0006h
BSR DX, AX    → ZF = 1, DX = 0009h
```

ii.
```
BSF DX, AX    → ZF = 1, DX = 0009h
BSR DX, AX    → ZF = 1, DX = 0006h
```

iii.
```
BSF DX, AX    → ZF = 0, DX = 0006h
BSR DX, AX    → ZF = 0, DX = 0006h
```

iv.
```
BSF DX, AX    → ZF = 1, DX = 0009h
BSR DX, AX    → ZF = 1, DX = 0009h
```

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.

Initial state:

EAX: 00000000h
EBX: 00000005h
ECX: FFFFFFFCh
EDX: E7A83170h
ESI: 00093000h
EDI: 00093010h

Address	Lo			Hi
93000h	B0	21	AA	36
93004h	15	99	FE	0C
93008h	CE	12	60	EB
9300Ch	89	0A	0B	FF
93010h	00	11	03	20
93014h	08	17	A1	B8
93018h	99	30	CB	ED

Instructions:

`MOVZX EAX, WORD PTR [EDI+ECX]` Aligned? ***Yes*** No Not a memory access

Address = EDI + ECX = 00093010h + FFFFFFFCh = 0009300Ch
(FFFFFFFCh = -4)
EAX = zero-extended word at 0009300Ch = 00000A89h

`MOV [EDI+EBX], DX` Aligned? Yes ***No*** Not a memory access

Address = EDI + EBX = 00093010h + 00000005 = 00093015h
mem(93015h) = DX = 3170h (byte at 93015h = 70h, 93016h = 31h)

`XCHG AL, [ESI+EBX+4]` Aligned? ***Yes*** No Not a memory access

Address = ESI + EBX + 04h = 00093000h + 5 + 4 = 00093009h
mem(93009h) = AL = 89h
AL = mem(93009h) = 12h

`MOVSX CX, BYTE PTR [ESI+3]` Aligned? ***Yes*** No Not a memory access

Address = ESI + 3 = 00093000h + 00000003h = 00093003h
CX = sign-extended byte at 00093003h = 0036h

`LEA SI, [DI+4*BX]` Aligned? Yes No ***Not a memory access***

SI = DI + 4 * BX = 3010h + 4 * 0005h = 3024h

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).

Initial state:
EAX: 00000C16h
EBX: 0000A037h
ECX: 0000FFF9h
EDX: 0000941Fh
CF: 1
ESI: 00072300h

Address	Lo			Hi
72300h	C0	00	02	10
72304h	10	10	15	5A
72308h	89	01	05	B1
7230Ch	20	40	AC	DC
72310h	04	08	05	83
72314h	FF	99	B0	11

Instructions:

```
SUB    DX, BX
```

DX = DX – BX = 941Fh – A037h = F3E8h, CF = 1

```
ADD    AX, DX
```

AX = AX + DX = 0C16h + F3E8h = FFFEh, CF = 1

```
INC    BYTE PTR [ESI+00000014h]
```

Address = ESI + 00000014h = 00072300h + 00000014h = 00072314h
(72314h) = (72314h) + 1 = FFh + 1 = 00h, CF = 1

```
NEG    CX
```

$$CX = -CX = -FFF9h = -(1111\ 1111\ 1111\ 1001_2)$$
$$= 0000\ 0000\ 0000\ 0110_2 + 1$$
$$= 0000\ 0000\ 0000\ 0111_2 = 0007h$$

```
IMUL   CL
```

AX = AL * CL = FEh * 07h = -2 * 7 = -14 = FFF2h

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: 00008980h
EBX: 0000FE92h
ECX: 00000003h
EDX: 000096B7h
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:

```
SHR   BX, CL
```

BX = BX >> CL (shift in zeroes) = FE92h >> 3 = 1FD2h, CF = 0

```
SAR   AX, 5
```

AX = AX >> 5 (keep sign intact) = 8980h >> 5 = FC4Ch, CF = 0

```
XOR   BX, AX
```

BX = BX XOR AX = FC4Ch XOR 1FD2h = E39Eh

```
NOT   AX
```

AX = ~AX = ~FC4Ch = 03B3h

```
AND   AX, DX
```

AX = AX AND DX = 03B3h AND 96B7h = 02B3h

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.

```
MOVSX   EBX, BYTE PTR [2150h]        ; Move a signed byte
                                     ;  from address 2150h &
                                     ;  extend it to fill EBX

SHL     EBX, 6                       ; Multiply EBX by 64 and
                                     ;  store the result in
                                     ;  EBX, in 1 instruction

LEA     EDI, [EBX + 4*ECX]           ; Set EDI equal to the
                                     ;  sum of EBX and (ECX
                                     ;  multiplied by 4) in a
                                     ;  single instruction

BT      BYTE PTR [EDI], 7            ; Access a byte at the
                                     ;  address stored in EDI
                                     ;  and copy its most
                                     ;  significant bit into
                                     ;  the carry flag

RCL     AL, 1                        ; Move the value in the
                                     ;  carry flag into the
                                     ;  least significant bit
                                     ;  of AL (you may change
                                     ;  other bits of AL)

BSF     CL, AL                       ; Find the position of
                                     ;  the rightmost (least
                                     ;  significant) nonzero
                                     ;  bit in AL, and store
                                     ;  that position in CL

XOR     EDX, FF000000h               ; Invert the upper 8
                                     ;  bits of EDX without
                                     ;  changing any other
                                     ;  bits in 1 instruction

ROL     AH, 4  -or-  ROR AH, 4       ; Swap the 4 upper bits
                                     ;  and 4 lower bits of
                                     ;  AH in 1 instruction

SAR     EDX, CL                      ; Use the value in CL
                                     ;  to shift EDX to the
                                     ;  right while keeping
                                     ;  the sign intact

BTC     EDX, 0                       ; Copy the rightmost bit
                                     ;  of EDX into the carry
                                     ;  flag, then invert
                                     ;  that bit
```