16.317: Microprocessor Systems Design I

Spring 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 = 3Ch` and `CF = 1`, what is the final result of the instruction `RCR AL, 3`?

i. AL = 27h, CF = 1

ii. `AL = 87h, CF = 1`

iii. `AL = E4h, CF = 1`

iv. `AL = E1h, CF = 1`

v. `AL = 07h, CF = 1`

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

```
MOV     AX, A
ADD     AX, B
CMP     C, AX
SETLE   BL
MOV     AX, D
CMP     AX, A
SETG    BH
OR      BL, BH
```

i. `(C <= B) || (D > A)`

ii. `(C <= A) || (D > A)`

iii. (C <= A + B) || (D > A)

iv. `(C < A + B) || (D > A)`

v. `(C <= A + B) || (D + B > A)`

1 (continued)

c. If `AX = 0FF0h`, which of the following instructions will set `CF = 1`?

A. `BT  AX, 3`
B. `BTR  AX, 4`
C. `BTS  AX, 15`
D. `BTC  AX, 12`

i. Only A

ii. <u>Only B</u>

iii. A and D

iv. B and C

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

d. If `AX = 0808H`, 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 = 0008h
BSR DX, AX      → ZF = 1, DX = 0008h
```

ii.
```
BSF DX, AX      → ZF = 1, DX = 0003h
BSR DX, AX      → ZF = 1, DX = 0004h
```

iii.
```
BSF DX, AX      → ZF = 0, DX = 0003h
BSR DX, AX      → ZF = 0, DX = 000Bh
```

iv.
```
BSF DX, AX      → ZF = 1, DX = 0003h
BSR DX, AX      → ZF = 1, DX = 000Bh
```

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: FFFFFFFAh
ECX: 00000003h
EDX: 0000FE98h
ESI: 00010480h
EDI: 00010470h

Address	Lo			Hi
10470h	02	18	20	15
10474h	10	55	AA	12
10478h	47	FE	DC	11
1047Ch	93	59	31	70
10480h	56	DD	BA	EE
10484h	0F	23	41	19
10488h	49	64	7A	0F

Instructions:

```
MOV    EAX, [ESI+EBX]
```
Aligned? Yes No Not a memory access

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

```
MOVSX  EDX, WORD PTR [ESI+ECX]
```
Aligned? Yes No Not a memory access

```
LEA    SI, [DI+BX+0003h]
```
Aligned? Yes No Not a memory access

```
MOVZX  AX, BYTE PTR [ESI+0002h]
```
Aligned? Yes No Not a memory access

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: 00000010h
EBX: 00005195h
ECX: 00001006h
EDX: 0000A197h
CF: 1
ESI: 00021800h

Address	Lo			Hi
21820h	99	07	08	F0
21824h	83	00	01	61
21828h	05	C1	71	31
2182Ch	20	40	33	80
21830h	05	00	AB	0F
21834h	41	82	11	55

Instructions:

```
ADD    DX, BX
```

```
DEC    AL
```

```
DIV    CL
```

```
SUB    AX, [ESI+0034h]
```

```
NEG    CX
```

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: 0000009Bh
EBX: 0000445Ch
ECX: 00000005h
EDX: 0000F63Ch
CF: 0

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

Instructions:

```
OR     AX, BX



SHL    AX, 5



NOT    BL



SAR    AX, 3



ROL    DX, 5
```

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.

```
_____________________________________ ; Move an unsigned word
                                      ;  from address 1000h
                                      ;  and extend it to fill
                                      ;  EAX
_____________________________________ ; Set ESI equal to the
                                      ;  sum of EAX and EBX,
                                      ;  in one instruction
_____________________________________ ; Move the upper word of
                                      ;  ECX into the lower
                                      ;  word of ECX
                                      ;  without losing bits
_____________________________________ ; Use two instructions
                                      ;  transfer the lower
                                      ;  word of ECX to the
_____________________________________ ;  address stored in ESI
                                      ;  if that lower word
                                      ;  represents a negative
                                      ;  signed value

_____________________________________ ; Divide CX by 32 and
                                      ;  store the result in
                                      ;  CX using a single
                                      ;  instruction
_____________________________________ ; Clear the lower 12
                                      ;  bits of EAX, but
                                      ;  don't change any
                                      ;  other bits
_____________________________________ ; Determine the position
                                      ;  of the leftmost (most
                                      ;  significant) nonzero
                                      ;  bit in EAX, and store
                                      ;  that position in DL
_____________________________________ ; Use two instructions
                                      ;  and the previous
                                      ;  instruction's result
                                      ;  to set DH to 1 if the
_____________________________________ ;  leftmost nonzero bit
                                      ;  in EAX is in the left
                                      ;  half of that register
```