16.317: Microprocessor Systems Design I

Fall 2015

Exam 1
September 30, 2015

Name: _____

For this exam, you may use a calculator and one 8.5" x 11" double-sided page of notes. All other electronic devices (e.g., cellular phones, laptops, PDAs) are prohibited. If you have a cellular phone, please turn it off prior to the start of the exam to avoid distracting other students.

The exam contains 5 questions. The first four questions will give you a total of 100 points; the fifth question is an extra credit problem worth 10 points. **In order to receive any extra credit for Question 5, you must clearly demonstrate that you have made a significant effort to solve each of the first four questions.**

Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

You will be provided with two pages (1 double-sided sheet) of reference material for the exam: a list of the x86 instructions we have covered thus far. You do not have to submit this sheet when you turn in your exam.

You will have 50 minutes to complete this exam.

Q1: Multiple choice	/ 20
Q2: Data transfers and memory addressing	/ 30
Q3: Arithmetic instructions	/ 30
Q4: Logical instructions	/ 20
TOTAL SCORE	/ 100
Q5: EXTRA CREDIT	/ 10

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. ***(rotate thru carry)*** 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. ***(bit test)*** 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. Only B

iii. A and D

iv. B and C

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

d. ***(bit scan)*** 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: 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

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

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

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

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

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
```

```
ADD    AX, DX
```

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

```
NEG    CX
```

```
IMUL   CL
```

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
```

```
SAR    AX, 5
```

```
XOR    BX, AX
```

```
NOT    AX
```

```
AND    AX, DX
```

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 a signed byte
                                              ;  from address 2150h &
                                              ;  extend it to fill EBX
_____________________________________________ ; Multiply EBX by 64 and
                                              ;  store the result in
                                              ;  EBX, in 1 instruction
_____________________________________________ ; Set EDI equal to the
                                              ;  sum of EBX and (ECX
                                              ;  multiplied by 4) in a
                                              ;  single instruction
_____________________________________________ ; Access a byte at the
                                              ;  address stored in EDI
                                              ;  and copy its most
                                              ;  significant bit into
                                              ;  the carry flag
_____________________________________________ ; Move the value in the
                                              ;  carry flag into the
                                              ;  least significant bit
                                              ;  of AL (you may change
                                              ;  other bits of AL)
_____________________________________________ ; Find the position of
                                              ;  the rightmost (least
                                              ;  significant) nonzero
                                              ;  bit in AL, and store
                                              ;  that position in CL
_____________________________________________ ; Invert the upper 8
                                              ;  bits of EDX without
                                              ;  changing any other
                                              ;  bits in 1 instruction
_____________________________________________ ; Swap the 4 upper bits
                                              ;  and 4 lower bits of
                                              ;  AH in 1 instruction
_____________________________________________ ; Use the value in CL
                                              ;  to shift EDX to the
                                              ;  right while keeping
                                              ;  the sign intact
_____________________________________________ ; Copy the rightmost bit
                                              ;  of EDX into the carry
                                              ;  flag, then invert
                                              ;  that bit
```