16.317: Microprocessor Systems Design I

Fall 2014

Exam 1
October 1, 2014

Name: _____ **ID #:** _____

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 four pages (2 double-sided sheets) of reference material for the exam: a list of the x86 instructions and condition codes we have covered thus far. You do not have to submit these pages 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	/ 25
Q4: Logical instructions	/ 25
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. How many iterations does the following loop execute?

```
          MOV  CX, 0008H
          MOV  AX, 0000H
START:    ADD  AX, 0003H
          CMP  AX, CX
          LOOPNE START
```

i. 2

ii. 3

iii. 4

iv. 6

v. 8

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

```
MOV     AX, D
CMP     A, AX
SETL    BL
SUB     AX, B
CMP     AX, C
SETGE   BH
AND     BL, BH
```

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

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

iii. `(A <= D) && (D – B >= C)`

iv. `(A < D) && (D – B >= C)`

v. `(A <= D) && (B – D >= C)`

1 (continued)

c. Given the conditional statement below:

```
if (AX == BX)
   CX++;
else
   CX--;
```

Which of the following assembly sequences correctly implements this conditional statement?

i.
```
     CMP  AX, BX
     JE   L1
     DEC  CX
     JMP  L2
L1:  INC  CX
L2:                ; End of statement
```

ii.
```
     CMP  AX, BX
     JNE  L1
     DEC  CX
     JMP  L2
L1:  INC  CX
L2:                ; End of statement
```

iii.
```
     CMP  AX, BX
     JE   L1
     DEC  CX
L1:  INC  CX
L2:                ; End of statement
```

iv.
```
     CMP  AX, BX
     JNE  L1
     DEC  CX
L1:  INC  CX
L2:                ; End of statement
```

d. If AX = 1001H, 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 = 0000h
BSR DX, AX      → ZF = 1, DX = 000Ch
```

ii.
```
BSF DX, AX      → ZF = 1, DX = 0000h
BSR DX, AX      → ZF = 1, DX = 0003h
```

iii.
```
BSF DX, AX      → ZF = 0, DX = 0000h
BSR DX, AX      → ZF = 0, DX = 000Ch
```

iv.
```
BSF DX, AX      → ZF = 1, DX = 000Ch
BSR DX, AX      → ZF = 1, DX = 0000h
```

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 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: 00000003h
EDX: 0000FE98h
ESI: 00092900h
EDI: 00092910h

Address	Lo			Hi
92900h	05	99	33	82
92904h	A0	11	FE	20
92908h	CC	08	19	27
9290Ch	02	17	20	14
92910h	16	31	70	AA
92914h	BE	CD	FA	00
92918h	49	64	7A	0F

Instructions:

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

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

```
LEA   CX, [SI+07FEh]
```
Aligned? Yes No Not a memory access

```
XCHG  [ESI+EBX+02h], DL
```
Aligned? Yes No Not a memory access

```
MOVZX EBX, BYTE PTR [ESI+001Ah]
```
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: 00000114h
EBX: 0000FFE8h
ECX: 00001013h
EDX: 00000004h
CF: 1
ESI: 000316F0h

Address	Lo			Hi
31700H	04	07	08	00
31704H	83	00	01	01
31708H	05	01	71	31
3170CH	20	40	60	80
31710H	05	00	AB	0F
31714H	00	16	11	55

Instructions:

```
SUB    CX, [ESI+20h]      (original exam had typo: SI, not ESI)

ADC    BX, AX

INC    BL

MUL    BYTE PTR [ESI+10h] (original exam had typo: SI, not ESI)

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: 000000FFh
EBX: 00001172h
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:

```
RCL    DL, 3



XOR    DX, AX



SAR    DH, 4



SHL    DL, 4



NOT    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.

```
_____________________________________ ; Set EDX to the sum
                                      ;  ECX + ESI + 00002014h
_____________________________________ ; Load a word into BX
                                      ;  from the address
                                      ;  stored in EDX
_____________________________________ ; Divide the signed
                                      ;  double word in
                                      ;  registers (DX,AX) by
                                      ;  the value in BX
_____________________________________ ; Use two instructions
                                      ;  to store the quotient
                                      ;  of the division at
                                      ;  address 93014h, and
_____________________________________ ;  store the remainder
                                      ;  of the division at
                                      ;  the next aligned
                                      ;  address after 93014h
_____________________________________ ; Multiply BX by 4 and
                                      ;  store the result in
                                      ;  BX using a single
                                      ;  instruction
_____________________________________ ; Complement the upper 4
                                      ;  and lower 4 bits of
                                      ;  BX, but don't change
                                      ;  any other bits
_____________________________________ ; Modify BX so that the
                                      ;  8 bits you just
                                      ;  changed are in BH,
                                      ;  and the remaining
                                      ;  bits are in BL
_____________________________________ ; Check the value of the
                                      ;  lowest bit of BL
_____________________________________ ; If that lowest bit is
                                      ;  0, store the value of
                                      ;  BH at the address
                                      ;  stored in ESI
```