EECE.3170: Microprocessor Systems Design I

Fall 2016

Exam 1

September 30, 2016

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., cell phones, laptops, tablets) are prohibited. If you have a cell 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 = 0x97, BL = 0xA6,` and `CF = 1`, what is the final result of the instruction `SBB AL, BL`?

   i. `AL = 0x91, CF = 0`

   ii. `AL = 0x91, CF = 1`

   iii. `AL = 0xF0, CF = 0`

   iv. `AL = 0xF0, CF = 1`

   v. `AL = 0xF1, CF = 1`

b. Given `AL = 0x7F` and `CF = 1`, what is the final result of the instruction `SAR AL, 3`?

   i. `AL = 0x0F, CF = 0`

   ii. `AL = 0x0F, CF = 1`

   iii. `AL = 0xFF, CF = 0`

   iv. `AL = 0xFF, CF = 1`

   v. `AL = 0x2F, CF = 1`

1 (continued)

c. If `DL = 0xA1`, `CL = 0x03`, and `CF = 0`, which instructions below will set `CF = 1`?

A. `ROL  DL, CL`
B. `RCL  DL, CL`
C. `ROR  DL, 2`
D. `RCR  DL, 2`

i. A and B

ii. B and C

iii. A and C

iv. B and D

v. None of the above

d. Given `AX = 0x0013` and `BL = 0xF9`, which of the following instructions would set `AL = 0x00` and `AH = 0x13`?

i. `IDIV BL`

ii. `DIV BL`

iii. `IMUL BL`

iv. `MUL BL`

v. None of the above

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: 0x00067340
EBX: 0x00000005
ECX: 0xFFFFFFFD
EDX: 0xDEADBEEF
ESI: 0x00091330
EDI: 0x00000003

Address	Lo			Hi
0x91330	40	03	09	00
0x91334	C9	82	F0	11
0x91338	D3	45	6A	6D
0x9133C	51	92	99	DD
0x91340	73	16	48	03

Instructions:

```
MOV   EDX, 0x00090330
```

Aligned? Yes No Not a memory access

```
MOVSX EAX, BYTE PTR [EBX+ESI]
```

Aligned? Yes No Not a memory access

```
XCHG  [EDX+0x1003], AX
```

Aligned? Yes No Not a memory access

```
LEA   EBX, [EAX+4*EDI]
```

Aligned? Yes No Not a memory access

```
MOVZX ECX, WORD PTR [ESI+0x0011]
```

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).

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

Initial state:
EAX: 0x0000A30F
EBX: 0x0000FA23
ECX: 0x0000AF07
EDX: 0x00000003
CF: 0
ESI: 0x000210A0

Address	Lo			Hi
0x210A0	32	0C	13	98
0x210A4	66	A0	FF	FF
0x210A8	E3	00	B5	08

Instructions:

```
SUB   BX, CX
```

```
INC   DWORD PTR [ESI+2*EDX]
```

```
ADD   AX, BX
```

```
IMUL  BYTE PTR [ESI+0x08]
```

```
NEG   BH
```

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: 0x0000B496
EBX: 0x000027A9
ECX: 0x00000003
EDX: 0x00002EA5
CF: 0

Address	Lo			Hi
0x31700	04	00	08	00
0x31704	83	00	01	01
0x31708	05	01	71	31
0x3170C	20	40	60	80
0x31710	02	00	AA	0F

Instructions:

```
XOR    AX, BX
```

```
SHR    AX, 6
```

```
AND    AH, BYTE PTR [0x31712]
```

```
ROL    AH, CL
```

```
NOT    EDX
```

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.

```
________________________________________ ; Copy the signed word
                                         ;  at address 0x1094
                                         ;  into EAX, keeping the
                                         ;  sign and value intact
________________________________________ ; Store the sum of EAX,
                                         ;  ESI, and the constant
                                         ;  value -2 in EDI,
                                         ;  using 1 instruction
________________________________________ ; If EDI now holds the
                                         ;  starting address of
                                         ;  an array of 32-bit
                                         ;  values, A, copy A[4]
                                         ;  into EBX
________________________________________ ; Swap the most and
                                         ;  least significant
                                         ;  words of EBX in 1
                                         ;  instruction
________________________________________ ; Invert the lower 8
                                         ;  bits of EBX without
                                         ;  changing any other
                                         ;  bits in 1 instruction
________________________________________ ; Divide ECX by 64 and
                                         ;  store the result in
                                         ;  ECX, in 1 instruction
________________________________________ ; Clear the middle 16
                                         ;  bits of EAX without
                                         ;  changing any other
                                         ;  bits in 1 instruction
________________________________________ ; Multiply the word at
                                         ;  the address stored in
                                         ;  EDI by 256, using a
                                         ;  single instruction
________________________________________ ; Move the carry flag
                                         ;  into the most
                                         ;  significant bit of
                                         ;  EAX (you may change
                                         ;  other bits of EAX)
________________________________________ ; Swap the lowest two
                                         ;  bytes of EAX with one
                                         ;  another
```