16.317: Microprocessor Systems Design I

Fall 2013

Exam 1

October 2, 2013

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. **<u>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.</u>**

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 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. If `AX = 0FF0h`, which of the following instructions will set `CF = 1` and change `AX` to `0EF0h`?

A. `BTR AX, 8`
B. `BT AX, 8`
C. `BTC AX, 8`
D. `BTS AX, 8`

i. A and C

ii. A and D

iii. B and C

iv. B and D

v. None of the above

b. 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

1 (continued)

c. If `AX = 000Fh` and `CF = 0`, initially, what is the result of the instruction `ROR AX, 4`?

i. `AX = 00F0h, CF = 0`

ii. `AX = F000h, CF = 1`

iii. `AX = E000h, CF = 1`

iv. `AX = 0000h, CF = 1`

v. `AX = 00FFh, CF = 1`

d. If `AX = 0001h` and `CF = 1`, initially, what is the result of the instruction `RCL AX, 2`?

i. `AX = 0000h, CF = 0`

ii. `AX = 0003h, CF = 0`

iii. `AX = 0004h, CF = 0`

iv. `AX = 0006h, CF = 0`

v. `AX = 1000h, CF = 1`

2. (30 points) ***<u>Data transfers and memory addressing</u>***

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: 00000006h
ECX: 00000001h
EDX: 0000FF00h
ESI: 0000F000h
EDI: 00001000h
DS: 9300h
ES: 9200h

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:

`MOV   ES:[DI+10h], BL` Aligned? Yes No Not a memory access

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

`MOV   AX, [SI+1003h]` Aligned? Yes No Not a memory access

`MOVZX EDX, BYTE PTR ES:[BX+1000h]` Aligned? Yes No Not a memory access

`MOVSX EBX, WORD PTR [000Eh]` 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: 0000FFF7h
EBX: 000000A4h
ECX: 00000003h
EDX: 0000FFFEh
CF: 1
ESI: 00000004H
DS: 3170H

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
31714H	00	16	11	55

Instructions:

```
SBB   BX, [SI]
```

```
ADD   AX, BX
```

```
DEC   AX
```

```
IDIV  CL
```

```
NEG   DL
```

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: 000000E7h
EBX: 00003300h
ECX: 00000002h
EDX: 0000F63Ch
CF: 0
DS: 7230h

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:

```
XOR   AL, [0DH]
```

```
AND   AL, BH
```

```
ROR   AL, CL
```

```
SAR   AL, 4
```

```
RCL   AL, 3
```

5. (10 points) ***Extra credit***

Complete the program 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.

```
______________________________________ ; Use two instructions
                                       ;   to establish 63170h
                                       ;   as the starting
______________________________________ ;   address of the data
                                       ;   segment

______________________________________ ; Load the first two
                                       ;   bytes stored in
                                       ;   the current data
                                       ;   segment into SI,
                                       ;   and the next two
                                       ;   bytes into ES

______________________________________ ; Set DI = SI + 1000h
                                       ;   using a single
                                       ;   instruction

______________________________________ ; Load two bytes of data
                                       ;   into AX from the
                                       ;   extra segment (ES),
                                       ;   starting at offset
                                       ;   specified by SI

______________________________________ ; Load the next two
                                       ;   bytes of data from
                                       ;   the extra segment
                                       ;   into BX

______________________________________ ; Find the sum of the
                                       ;   previous two values

______________________________________ ; Divide the result of
                                       ;   the previous
                                       ;   instruction by 2
                                       ;   without using a
                                       ;   divide instruction
                                       ; Keep the sign intact

______________________________________ ; Store the previous
                                       ;   instruction’s result
                                       ;   into the extra
                                       ;   segment at offset
                                       ;   specified by DI
```