16.317: Microprocessor Systems Design I

Spring 2014

Exam 1

February 19, 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 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. Which of the following statements about x86 real mode memory accesses are true?

A. By default, if an instruction that accesses memory does not explicitly specify a segment, that instruction will access the data segment.

B. In an x86 processor, all memory accesses involving multiple bytes must be aligned.

C. To calculate a linear address in the data segment, add the 16-bit value in DS to the 16-bit effective address. (For example, if DS = 1000h and the effective address is 2000h, the linear address will be 3000h.)

i. None of the above

ii. Only A

iii. Only B

iv. A and B

v. B and C

b. If `EAX = 10203040h` and `EBX = AABBCCDDh`, which of the following instructions will change `EAX` to `102030DDh` and `EBX` to `AABBCC40h`?

i. `XCHG EAX, EBX`

ii. `XCHG AX, BX`

iii. `XCHG AL, BL`

iv. `XCHG AH, BH`

v. None of the above

1 (continued)

c. If `EAX = 00000003h` and `EBX = 00000005h`, what will the result of the instruction `IMUL BL` be?

i. `EAX = 00000005h`

ii. `EAX = 0000000Fh`

iii. `EAX = 00000015h`

iv. `EAX = 0000FFF8h`

v. `EAX = 00000005h, EDX = 00000003h`

d. Which of the following instructions will set `CF = 1` if `EAX = 0000F001h` and `EBX = 00001000h`?

A. `ADD AX, BX`

B. `SUB BX, AX`

C. `SHR AX, 1`

D. `SHL BX, 1`

i. Only A

ii. Only D

iii. A and C

iv. B and D

v. A, B, and C

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

For each data transfer instruction shown below, list <u>all</u> changed registers and/or memory locations and their final values. If memory is changed, be sure to explicitly list **<u>all changed bytes</u>**. Also, indicate if each instruction performs an aligned memory access, an unaligned memory access, or no memory access at all.

<u>Initial state:</u>

EAX: 00000000h
EBX: 00000002h
ECX: 00000001h
EDX: 00001FFEh
ESI: 0000F00Fh
EDI: 0000A000h
DS: 1245h
ES: 1046h

Address	Lo			Hi
12450h	02	17	20	14
12454h	16	31	70	AA
12458h	BE	CD	FA	00
1245Ch	49	64	7A	0F
12460h	FF	11	02	60
12464h	01	04	65	7F
12468h	99	30	88	78

<u>Instructions:</u>

`MOV EAX, [BX+4*CX]` <u>Aligned?</u> Yes No Not a memory access

`MOV ES:[DI+8005h], DL` <u>Aligned?</u> Yes No Not a memory access

`LEA BX, [SI+0FF3h]` <u>Aligned?</u> Yes No Not a memory access

`MOVSX EDX, BYTE PTR ES:[CX+2009h]` <u>Aligned?</u> Yes No Not a memory access

`MOVZX EBX, WORD PTR [0009h]` <u>Aligned?</u> 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: 00000014h
EBX: 0000FF08h
ECX: 00000003h
EDX: 00000004h
CF: 1
ESI: 00000008H
DS: 3170H

Address	Lo			Hi
31700H	04	07	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:

```
ADD   CX, [SI]
```

```
SBB   BX, AX
```

```
DEC   BH
```

```
IDIV  BYTE PTR [0001h]
```

```
NEG   DX
```

4. (25 points) ***<u>Logical instructions</u>***

For each instruction in the sequence shown below, list <u>all</u> changed registers and/or memory locations and their new values. If memory is changed, be sure to explicitly list **<u>all changed bytes</u>**. Where appropriate, you should also list the state of the carry flag (CF).

<u>Initial state:</u>

EAX: 000000FFh
EBX: 00001172h
ECX: 00000005h
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

<u>Instructions:</u>

```
AND   BL, [07H]
```

```
XOR   AL, BL
```

```
SAR   AL, CL
```

```
SHL   AL, 4
```

```
NOT   AL
```

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. You should assume the starting address of the data segment is appropriately set up for you.

```
________________________________________ ; Load the first two
                                         ;  bytes in the data
                                         ;  segment into BX

________________________________________ ; Use one instruction to
                                         ;  load the next two
                                         ;  bytes in the data
                                         ;  segment into CX,
                                         ;  and the two bytes
                                         ;  after that into GS

________________________________________ ; Find the difference
                                         ;  CX – BX, storing
                                         ;  the result in CX

________________________________________ ; Use two instructions
                                         ;  to take the new
                                         ;  value in CX and
________________________________________ ;  multiply it by BX
                                         ; Hint: the result may
                                         ;  not be in either
                                         ;  of those registers

________________________________________ ; Store all 32 bits of
                                         ;  that result in the
                                         ;  first four bytes of
________________________________________ ;  segment GS, using two
                                         ;  instructions (least
                                         ;  significant bits 1st)

________________________________________ ; Clear the upper 4
                                         ;  bits of BX, but don't
                                         ;  change any other bits
                                         ; (Clear = set to 0)

________________________________________ ; Flip the lowest 4 bits
                                         ;  of BX, but don't
                                         ;  change any other bits

________________________________________ ; Store BX into the data
                                         ;  segment at the offset
                                         ;  specified by the sum
                                         ;  of DI and CX
```