# EECE.3170: Microprocessor Systems Design I

Summer 2016

Exam 1 Solution

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]

***EDX = 0x00090330 (simply assigns constant to register)***

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

***Address = EBX + ESI = 00000005 + 0x00091330 = 0x00091335***
***EAX = sign-extended byte at 0x91335 = 0xFFFFFF82***

`XCHG   [EDX+0x1003], AX` Aligned? Yes [✓ No] Not a memory access

***Address = EDX + 0x1003 = 0x90330 + 0x1003 = 0x91333***
***mem(0x91333) = AX = 0xFF82 (byte at 0x91333 = 0x82, 0x91334 = 0xFF)***
***AX = word at 0x91333 = 0xC900h***

`LEA    EBX, [EAX+4*EDI]` Aligned? Yes No [✓ Not a memory access]

***EBX = EAX + 4 * EDI = 0xFFFFC900 + 4 * 3 = 0xFFFFC90C***

`MOVZX  ECX, WORD PTR [ESI+0x0011]` Aligned? Yes [✓ No] Not a memory access

***Address = ESI + 0x0011 = 0x00091330 + 0x0011 = 0x91341***
***ECX = zero-extended word at 0x91341 = 0x00004816***

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
```

***BX = BX – CX = 0xFA23 – 0xAF07 = 0x4B1C, CF = 0***

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

***Address = ESI + 2 * EDX = 0x000210A0 + 2 * 3 = 0x210A6***
***mem(0x210A6) = mem(0x210A6) + 1 = 0x00E3FFFF + 1 = 0x00E40000***
***(byte at 0x210A6 = 0x00, 0x210A7 = 0x00, 0x210A8 = 0xE4, 0x210A9 = 0x00)***

```
ADD   AX, BX
```

***AX = AX + BX = 0xA30F + 0x4B1C = 0xEE2B, CF = 0***

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

***Address = ESI + 0x08 = 0x210A8***
***AX = AL * byte @ 0x210A8 = 0x2B * 0xE4 = 43 * -28***
***= -1204 = FB4Ch***

```
NEG   BH
```

***BH = -BH = -0x4B = -(0100 1011$_2$) = 1011 0100$_2$ + 1***

***= 1011 0101 = 0xB5***

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
```

***AX = AX XOR BX = 0xB496 XOR 0x27A9 = 0x933F***

```
SHR   AX, 6
```

***AX = AX >> 6 (shift in zeroes) = 0x933F >> 6 = 0x024C, CF = 1***

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

***AH = AH AND mem(0x31712) = 0x02 AND 0xAA = 0x02***

```
ROL   AH, CL
```

***AH = AH rotated left by CL = 0x02 rotated left by 3***
***= 0x10, CF = 0***

```
NOT   EDX
```

***EDX = ~EDX = ~0x00002EA5 = 0xFFFFD15A***

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.

```
MOVSX   EAX, WORD PTR [0x1094]             ; Copy the signed word
                                           ;  at address 0x1094
                                           ;  into EAX, keeping the
                                           ;  sign and value intact

LEA     EDI, [EAX+ESI-2]                   ; Store the sum of EAX,
                                           ;  ESI, and the constant
                                           ;  value -2 in EDI,
                                           ;  using 1 instruction

MOV     EBX, [EDI+16]                      ; If EDI now holds the
                                           ;  starting address of
                                           ;  an array of 32-bit
                                           ;  values, A, copy A[4]
                                           ;  into EBX

ROR     EBX, 16 -or- ROL  EBX, 16          ; Swap the most and
                                           ;  least significant
                                           ;  words of EBX in 1
                                           ;  instruction

XOR     EBX, 0x000000FF -or- NOT BL        ; Invert the lower 8
                                           ;  bits of EBX without
                                           ;  changing any other
                                           ;  bits in 1 instruction

SHR     ECX, 6 -or- SAR  ECX, 6            ; Divide ECX by 64 and
                                           ;  store the result in
                                           ;  ECX, in 1 instruction

AND     EAX, 0xFF0000FF                    ; Clear the middle 16
                                           ;  bits of EAX without
                                           ;  changing any other
                                           ;  bits in 1 instruction

SHL     WORD PTR [EDI], 8                  ; Multiply the word at
                                           ;  the address stored in
                                           ;  EDI by 256, using a
                                           ;  single instruction

RCR     EAX, 1                             ; Move the carry flag
                                           ;  into the most
                                           ;  significant bit of
                                           ;  EAX (you may change
                                           ;  other bits of EAX)

XCHG  AL, AH  -or-  ROL  AX, 8             ; Swap the lowest two
  -or- ROR  AX, 8                          ;  bytes of EAX with one
                                           ;  another
```