Solution to CS 3843 Midterm Exam Two Fall 2012

Name (Last)_______________, (First)_______________

You may use a calculator and one sheet of notes on this exam, but no other materials and no computer.

This test has a full score of 110 points. Answer question worth 100 points or more. The exam will be graded for a maximum score of 100 points. Show all the major steps in your work to receive partial credits.

Problem 1 (31 points)

Assume the following values are stored at the indicated memory addresses and registers

Address	Value	Register	Value
0x1000	0x1A	%eax	0x1000
0x1004	0x34	%ecx	0x2
0x1008	0xBF	%edx	0x5
0x100C	0x11		
0x1010	0xA2		
0x1014	0x10		

a) (16 points – 2 points each) Fill the following table:

Operand	Value	Operand	Value
%edx	0x5	(%eax, %ecx, 4)	0xBF
4(%eax)	0x34	0xFF8(, %edx, 4)	0x11
10(%eax, %ecx)	0x11	0x1000(%ecx, %edx, 2)	0x11
leal -0x10(%eax, %ecx, 8), %edx	0x1000	leal 0xFC(%ecx), %edx	0xFE

b) (15 points – 0.75 point each) Fill in the following table

Instruction	Destination	Value
decl %edx	%edx	0x4
andl %ecx, %edx	%edx	0x0
addl %edx, 4(%eax)	0x1004	0x39
imul $4, (%eax, %edx, 4)	0x1014	0x40
incl 0xC(%eax)	0x100C	0x12

Problem 2 (35 points – 1 point each) Fill in the following table. Assume that *x* and *y* are of type `short` which is 16 bits. Enter the value of (*y*-*x*) in decimal. This is the value that would be stored in *z* if `short z = y - x;`

For your convenience, the following numbers are provided for you:

The range of 16-bit signed number: -32768 ~ 32767
The range of 16-bit unsigned number: 0 ~ 615535

Sol: 2's complement representation:

$-9=N^*=2^{16}-N=2^{16}-9=65527$

$0x7ffe=2^{15}-2=32766$

$-0x7ffe=2^{16}-0x7ffe=65536-32766=32770$

Consider the instruction: `cmpw %eax, %ecx`

Fill in the value of the flags if `%eax` contains *x* and `%ecx` contains *y*.

x	*y*	*z* = *y* - *x*	ZF	SF	OF	CF
12	-9	-9-12 = -21 (signed) 65527-12=65515 (unsigned)	0	1	0	0
-9	12	12-(-9) = 21 (signed) 12-65527= -65515 (unsigned)	0	0	0	1
-9	-9	-9-(-9) = 0	1	0	0	0
9	0*x*7ffe	32766– 9 =32757	0	0	0	0
9	-0*x*7ffe	-32766-9=-32775 (signed) 32770-9=32761 (unsigned)	0	0	1	0
0x7ffe	9	9 – 32766 = -32757	0	1	0	1
0x7ffe	-9	-9-32766= -32775 (signed) 65527– 32766 = 32761 (unsigned)	0	0	1	0

Problem 3 (24 points) Please determine whether the following instruction is TRUE or FALSE, and if FALSE, and what's wrong with each line?

1) movl (%eax), 0x4(%esp) TURE ( ) FALSE (X)

If FALSE, explain why?

Ans: Cannot have both source and destination be memory address.

2) movb $0xFF, (%al) TURE ( ) FALSE (X)

If FALSE, explain why?

Ans: Cannot use %al as address register

3) movl %eax, 0xF(%edx) TURE (X) FALSE ( )

If FALSE, explain why?

4) movl %cx, (%edx) TURE ( ) FALSE (X)

If FALSE, explain why?

Ans: Mismatch between instruction suffix and register ID.

5) movl %ecx, %dx TURE ( ) FALSE (X)

If FALSE, explain why?

Ans: Destination operand incorrect size.

6) movl %eax, $0xFFD TURE ( ) FALSE (X)

If FALSE, explain why?

Ans: Cannot have immediate as destination

Problem 4 (20 points)

Based on the assembly code, (a) (5 points – 1 point per line) comment each assembly instruction and (b) (15 points) fill in the missing portion of the C code.

The portion of the generated assembly code implementing these expressions is as follows:

x at %ebp +8, *y* at %ebp+12, *z* at %ebp +16

```
1.  movl    12(%ebp),   %eax      // y into %eax
2.  xorl    8(%ebp), %eax         // %eax = y^x
3.  sall    $3, %eax              //  %eax << 3
4.  notl    %eax                  // %eax = ~%eax
5.  subl    16(%ebp), %eax        //  %eax = %eax - z
```

The expression of the C code:

1. int *arith* (int *x*, int *y*, int *z*) {
2. {
3. int *t1* = ___*x^y*_____________(4 points);
4. int *t2* = ___*t1<<3*__________ (4 points);
5. int *t3* = ___*~t2*_____________(4 points);
6. int *t4* = ____*t3*-z____________(3 points);
7. return *t4*;
8. }