CS 3843 Final Exam Fall 2012

Name (Last)_______________, (First)________________

Please indicate your session: Morning_______ Afternoon__________

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

This test has a full score of 120 points. Answer question worth 110 points or more. The exam will be graded for a maximum score of 110 points. Show all the major steps in your work to receive partial credits.

Problem 1 (14 points)

(a) (1 point) Convert to decimal number: 0xA9C4

Sol: 0xA9C4 = $10\times16^3+9\times16^2+12\times16+4= 43460$

(b) (1 point) Convert to binary: 0xC7EF

Sol: 0xC7EF = $(1100,0111, 1110, 1111)_2$

(c) (2 point) Convert from decimal to hexadecimal and binary: 2617

Sol: $2617 = 10\times16^2+ 3\times16+9=$ 0xA39

Convert to binary= $(1010, 0011, 1001)_2$

(d) (4 points) Convert decimal number 382.45 to octal number (base 8) (hint: Division and Multiply method for converting integer and fraction part)

Sol: 1) integer part $382 = (576)_8$

$382= 5*8^2+7*8+6$

2) fraction part $.45= (.34631,4631,4631,...)_8$

```
 .45       .60      .80      .40      .20      .60
 × 8       × 8      × 8      × 8      × 8      × 8
--------  -------- ------   -----    -----    -----

(3).6     (4).8    (6).4    (3).2    (1).6    (4).8 ...
```

Therefore $382.45 =(576.34631,4631,4631,...)_8$

(e) (3 points) Use a 10-bit word, find the binary representation of -456 in

i) two's complement

ii) one's complement

iii) sign-magnitude

Sol:

$456=(01,1100,1000)_2= N$

i) $N^*=(10,0011,1000)_2$

ii) $\overline{N} = (10,0011,0111)_2$

iii) $N =(11,1100,1000)_2$

(f) (3 points) repeat problem (e) using a 12-bit word.

Sol:

$456=(0001,1100,1000)_2= N$

i) $N^*=(1110,0011,1000)_2$

ii) $\overline{N} = (1110,0011,0111)_2$

iii) $N =(1001,1100,1000)_2$

Problem 2 (9 points)

Assume x = 0xFDC4 and y=0x79A8, what is the following value:

Sol:

x = 0xFDC4 = $(1111, 1101, 1100, 0100)_2$
~x = ~0xFDC4 = $(0000, 0010, 0011, 1011)_2$
y = 0x79A8 = $(0111, 1001, 1010, 1000)_2$
~y = ~0x79A8 = $(1000, 0110, 0101, 0111)_2$

(a) (1.5 points) $\sim x \,\&\, y$

~x = ~0xFDC4 = $(0000, 0010, 0011, 1011)_2$
y = 0x79A8 = $(0111, 1001, 1010, 1000)_2$
~x & y = $(0000, 0000, 0010, 1000)_2$ = 0x28

(b) (1.5 points) $x \mid \sim y$

x = 0xFDC4 = $(1111, 1101, 1100, 0100)_2$
~y = ~0x79A8 = $(1000, 0110, 0101, 0111)_2$
$x \mid \sim y$ = $(1111,1111,1101,0111)_2$ = 0xFFD7

(c) (1.5 points) *~x & ~y*

~x = ~0xFDC4 = $(0000, 0010, 0011, 1011)_2$
~y = ~0x79A8 = $(1000, 0110, 0101, 0111)_2$
~x | ~y = $(0000, 0010, 0001, 0011)_2$ = 0x213

(d) (1.5 points) *x && y*

x && y = TURE && TRUE = TRUE =0x1

(e) (1.5 points) *x &! y*

x &! y = *x* & !TRUE = *x* & FALSE = *x* & 0 = 0

(f) (1.5 points) *x ||~ y*

x ||~ y = TRUE || TRUE = TRUE= 0x1

Problem 3 (12 points) Consider the following two 7-bit floating point representations based on the IEEE floating point format. Neither has a sign bit - they can only represent non-negative numbers.

i). Format A.

- There are *k=3* exponent bits. The exponent bias is 3.
- There are *n=4* fraction bits.

ii). Format B.

- There are *k=4* exponent bits. The exponent bias is 7.
- There are *n=3* fraction bits.

Below, you are given some bit patterns in Format A, and your task is to convert them to the closest value in Format B. If necessary, you should apply the round-to-even rounding rule. In addition, give the values of numbers given by the Format A and Format B bit patterns. Give these as whole numbers (e.g., 17) or as fractions (e.g., 17/64).

Format A		Format B	
Bits	Value	Bits	Value
011,0000	1	0111,000	1
101,1110	15/2	1001,111	15/2
010,1001	25/32	0110,100	¾
110,1111	31/2	1011,000	16
		1010,111	15

Problem 4 (12 points) Consider a hypothetical 10-bit IEEE floating point representation:

```
----------------------------------------------
| s (1 bit) | exp ( 5 bits) | frac (4 bits)    |
----------------------------------------------
```

a) (2 points) What is the bias?

b) (2 points) How many different values can be represented with 10 bits?

c) (2 points) What is the smallest positive normalized value?

d) (2 points) What is the largest positive normalized value?

e) (2 points) What is the largest positive denormalized value?

f) (2 points) What is the floating-point representation for 1.0?

Sol:

a) bias = 2^{k-1}-1=2^{5-1}-1=2^4-1=15

b) 2^{10}=1024

c) V = $(-1)^s \times M \times 2^E$,   s=0: positive value,

E = exp-bias = $(00001)_2$-15=1-15=14

M=1+frac$\times 2^{-n}$=1+(0000) $\times 2^{-4}$=1.0

smallest normalized positive value = 1.0*2^{-14}=1/(2^{14})

d) largest normalized positive value

s = 0, exp!=00000 and exp!=11111, exp=$(11110)_2$=16+8+4+2=30

E=exp-bias=30-15=15, frac = $(1111)_2$

M = 1 + frac $\times 2^{-4}$=1+0.1111=1.1111=1+1/2+1/4+1/8+1/16=29/16

therefore, V = $(-1)^0 \times 29/16 \times 2^E$=29/16$\times 2^{15}$

= (1.1111) $\times 2^{15}$=$(1111,1000,0000,0000)_2$

(e) largest denormalized value

s = 0, exp =00000, frac = 1111

M=frac $\times 2^4$=0.111,   E=1-bias=1-15=14;

V = M$\times 2^E$=0.1111$\times 2^{-14}$=$(00, 0000, 0000, 0000, 1111)_2$

f) V = 1.0= $(-1)^0 \times 1.0 \times 2^{-0}$

s = 0, M = 1.0, E=0

E = 0 = exp - bias => exp = bias = 15 = $(01111)_2$

M = 1 + frac$\times 2^{-4}$=1.0 => frac = $(0000)_2$

therefore, 1.0 =

s	k=5	n=4
0	01111	0000

Problem 5 (8 points) For each of the following values of *K*, find ways to express *x*K* using only the specified number of operations, where we consider both additions and subtractions to have comparable cost.

K	shifts	Add/Subs	Expression
28	2	1	*(x<<5)-(x<<2)*
63	1	1	*(x<<6)-x*
-12	2	1	*(x<<2)-(x<<4)*
97	2	2	*(x<<7)-(x<<5)+x*

Problem 6 (9 points) Assume the following values are stored at the indicated memory addresses and registers

Address	**Value**	**Register**	**Value**
0x200	0x3A	%eax	0x200
0x204	0x45	%ecx	0x08
0x208	0x3D	%edx	0x03
0x20C	0x11		
0x210	0x2F		
0x214	0x09		

a) (4 points) Fill the following table:

Operand	**Value**	**Operand**	**Value**
12(%eax)	0x11	4(%eax, %edx, 4)	0x2F
4(%eax, %ecx)	0x11	0x1F0(, %edx, 8)	0x3D
-3(%eax, %edx)	0x3A	0x200(%ecx, %edx, 4)	0x09
leal -0x10(%eax, %ecx, 8), %edx	0x230	leal 0xFC(%ecx), %edx	0x104

b) (5 points) Fill in the following table

Instruction	**Destination**	**Value**
xorl %edx, %ecx	%ecx	0x0B
subl %ecx, %edx	%edx	0x05
addl %edx, 4(%eax, %ecx, 2)	0x214	0x0C
imul $4, (%eax, %edx, 4)	0x20C	0x44
incl 0x8(%eax)	0x208	0x3E

Problem 7 (14 points) Fill in the following table. Assume that *x* and *y* are of a new type `short` integer which is 12 bits. Enter the value of (*y*-*x*) in decimal. This is the value that would be stored in *z* if `short z = y - x;`

The range of 12-bit signed number: -2048 ~ 2047
The range of 12-bit unsigned number: 0 ~ 2^{12}-1=4095

Sol: 2's complement representation:

-7=$N^* = 2^{12} - N = 2^{12} - 7 = 4089$

0x7fe =$2^{11} - 2 = 2046$

-0x7fe=$2^{12} - 0x7\text{fe}$=4096-2046=2050

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

(2 points each) Fill in the value of the flags if `%eax` contains *x* and `%ecx` contains *y*.

x	*y*	*z* = *y* - *x*	ZF	SF	OF	CF
15	7	7-15=-8(signed)	0	1	0	0
-7	15	15-(-7)=22 (signed) 15-4089=-4074 (unsigned)	0	0	0	1
-7	-7	(-7)-(-7)=0	1	0	0	0
7	0x7fe	2046-7=2039	0	0	0	0
7	-0x7fe	-2046-7=-2053(signed) 2050-7=2043(unsigned)	0	0	1	0
0x7fe	7	7-2046=-2039	0	1	0	1
0x7fe	-7	-7-2046=-2053(signed) 4089-2046=2043 (unsigned)	0	0	1	0

Problem 8 (12 points) Please check whether the following instruction is TRUE or FALSE, and if FALSE, and what's wrong with each line?

1) movl %edx, 0xFD(%eax) TURE (X) FALSE ( )

If FALSE, explain why?

2) movl (%ecx), 0xC(%esp) TURE ( ) FALSE (X)

If FALSE, explain why?

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

3) movb $0xFF, (%ah) TURE ( ) FALSE (X)

If FALSE, explain why?

Ans: Cannot use %al as address register

4) movw %ecx, (%edx) TURE ( ) FALSE (X)

If FALSE, explain why?

Ans: Mismatch between instruction suffix and register ID.

5) movb %cl, %dx TURE ( ) FALSE (X)

If FALSE, explain why?

Ans: Destination operand incorrect size.

6) movw %ax, $0xCD TURE ( ) FALSE (X)

If FALSE, explain why?

Ans: Cannot have immediate as destination

Problem 9 (20 points)

<table>
<tr>
<td>

C Code (caller.c)

```
int caller() {
     int arg1 = 38;
     int arg2 = 125;
     int diff = swap_subs(&arg1,
&arg2);
     int sum = arg1 + arg2;
     return sum+diff;
}
```

</td>
<td>

Here is the assembly code generated:

```
caller:
      pushl   %ebp
      movl    %esp, %ebp
      subl    $24, %esp                  // allocate 6 words on the
stack
      movl    $38, -4(%ebp)              // 38 on the stack
      movl    $125, -8(%ebp)             // 125 on the stack
      leal    -8(%ebp), %eax             // &125  into %eax
      movl    %eax, 4(%esp)              //  &125 on stack
      leal    -4(%ebp), %eax             // &38 into %eax
      movl    %eax, (%esp)               //  &38 on stack
      call    swap_subs
.R1   addl    -8(%ebp), %eax             //   %eax = %eax(diff) + 38;
      addl    -4(%ebp), %eax             //    %eax += 125
      leave                              // restore the stack pointer
      ret
```

</td>
</tr>
</table>

C Code(swap_subs.c)	Here is the assembly code generated:

int swap_subs(int *arg1, int *arg2)
{
    int tmp = *arg1;
    *arg1 = *arg2;
    *arg2 = tmp;
    return *arg1 - *arg2;
}
	swap_subs:
	    pushl %ebp
	    movl %esp, %ebp
	    pushl %ebx
	    movl 8(%ebp), %edx // arg1 into %edx
	    movl 12(%ebp), %ecx // arg2 into %ecx
	    movl (%edx), %ebx // %ebx = *arg1;
	    movl (%ecx), %eax // %eax = *arg2
	    movl %eax, (%edx) // *arg1 = *arg2;
	    movl %ebx, (%ecx) // *arg2 = *arg1;
	    movl (%edx), %eax // %eax = *arg2(old);
	    subl %ebx, %eax // %eax = *arg2(old) - *arg1(old)
	    popl %ebx
	    popl %ebp
	    ret

Keep track of register values: %eax, %ebx, %ecx, %edx, %ebp, %esp

address Stack Bottom (10 points)

address	
1004	xxxx
1000	old %ebp
FFC	~~38~~ 125
FF8	~~125~~ 38
FF4	
FF0	
FEC	FF8
FE8	FFC
FE4	.R1
FE0	1000
FDC	old %ebx

Stack Top

(10 points)

%eax	FF8	FFC	125	125	87	125	250				
%ebx	38										
%ecx	FF8										
%edx	FFC										
%ebp	1000	FE0	1000	old %ebp							
%esp	1004	1000	FE8	FE4	FE0	FDC	FE0	FE4	FE8	1000	1004

Problem 10 (10 points) For each byte sequence listed, determine the Y86 instruction sequences it encodes. If there is some invalid byte in the sequence, show the instruction sequence up to that point and indicate where the invalid value occurs. For each sequence, we show that the starting address, then a colon, and then the byte sequence.

0x100: 30F30F00000020314013FDFFFFFF60317008010000

Sol:

```
.pos   0x100    # start code at address 0x100
       irmovl  $15, %ebx           # load 15 into %ebx                     0x100:   30f30f000000
       rrmovl  %ebx, %ecx          # copy 15 to %ecx                       0x106:   2031
loop:                              # loop                                0x108:
       rmmovl %ecx, -3(%ebx)       # save %ecx at address 15-3=12          0x108:  4013fdffffff
       addl        %ebx, %ecx      # increment %ecx by 15                  0x10e:  6031
       jmp   loop                  # Goto loop                             0x110:  7008010000
```