Kaleb J. Himes
kaleb@gmail.com
kaleb@wolfssl.com
CSCI 361
Due Mon Oct 6th 2014

Problem 1 [20 pts]
Complete the following problems as instructed. Show the your work to receive credit.

(a) Convert the following 8-bit unsigned binary numbers to decimal. (these are 9 bit binary's)

Binary

|0 | 0 | 0| 0| 1| 0| 0| 1| 1| = 16+2+1 = **19**

|0 | 1 | 0| 1| 0| 1| 0| 1| 0| = 128+32+8+2 = **170**

|256| 128 | 64| 32| 16| 8| 4| 2| 1|

(b) Convert the following decimal numbers to both 8-bit Sign Magnitude and Two's Complement Notation.

	Sign Magnitude	Two's Complement
37	**0 0 1 0 0 1 0 1**	**0 0 1 0 0 1 0 1**
-37	**1 0 1 0 0 1 0 1**	**1 1 0 1 1 0 1 1**
-121	**1 1 1 1 1 0 0 1**	**1 0 0 0 0 1 1 1**
	(128)(64)(32)(16)(8)(4)(2)(1)	(flip the bits and add 1 if negative else stays same)

(c) Convert the following two hexidecimal numbers to decimal.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	3	4	5	6	7	8	9	a	b	c	d	e	f

16 powers:

8	7	6	5	4	3	2	1
268,435,456	16,777,216	1,048,576	65536	4096	256	16	1

Hex	
8ce	= 8*256 + 12*16 + 14*1 = 2048 + 192 + 14 = **2254**
a b c d e f 12	= **2,882,400,018** (see below)

(10*268,435,456) + (11*16,777,216) + (12*1,048,576) + (13*65536) + (14*4096) + (15*256) + (1*16) + (2*1) =

2,684,354,560 + 184,549,376 + 12,582,912 + 851,968 + 57,344 + 3,840 + 16 + 2 =
2,882,400,018

Kaleb J. Himes

Problem 2 [10 pts]

What are the largest and the smallest integers respresentable in:

2^15	2^14	2^13	2^12	2^11	2^10	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(a) 4-bit unsigned binary representation

LARGEST: 1 1 1 1 = 8+4+2+1 = **15**

SMALLEST: 0 0 0 0 = 0+0+0+0 = **0**

(b) 4-bit sign-magnitude representation

LARGEST: 0 1 1 1 = [+](4+2+1) = **7**

SMALLEST: 1 1 1 1 = [-](4+2+1) = **-7**

(c) 4-bit two's complement representation

LARGEST: 0 1 1 1 = 0+4+2+1 = **7**

SMALLEST: 1 0 0 0 = -8+0+0+0 = **-8**

(d) 8-bit unsigned binary representation

LARGEST: 1 1 1 1 1 1 1 1 = 128+64+32+16+8+4+2+1 = **255**

SMALLEST: 0 0 0 0 0 0 0 0 = 0 + 0 + …+ 0 = **0**

(e) 8-bit sign-magnitude representation

LARGEST: 0 1 1 1 1 1 1 1 1 = 64+32+16+8+4+2+1 = 255 – 128 = **127**

SMALLEST: 1 1 1 1 1 1 1 1 1 = 64+32+...+1 = **-127**

(f) 8-bit two's complement representation

LARGEST: 0 1 1 1 1 1 1 1 = **127**

SMALLEST: 1 0 0 0 0 0 0 0 = **-128**

(g) 16-bit unsigned binary representation

LARGEST: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 = **65,535**

SMALLEST: 0 0 0 0 0 0 0 0 = **0**

(h) 16-bit sign-magnitude representation

LARGEST: 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1=65,535-32,768 = **32,767**

SMALLEST: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 = **-32,767**

(i) 16-bit two's complement representation

LARGEST: 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 = **32,767**

SMALLEST: 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 = **-32,768**

(j) Explain why your answers differ between (h) and (i).

Because in "h" we have both -0 (1000000000000000) and 0 (0000000000000000) therefore we have lost one potential representation where in "i" we only have one zero and the negative zero before, is now the smallest possible number (-32,768).

Kaleb J. Himes

Problem 3 [10 pts]

2^15	2^14	2^13	2^12	2^11	2^10	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
									a	b	c	d	e	f

$t0	$t1	$t2	$t3	$t4	$t5	$t6	$t7	$s0	$s1	$s2	$s3	$s4	$s5	$s6	$s7
8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

X : 0000 00 01 001 1 0010 0110 1 000 00 10 0010 (base two)
Y : 1000 11 01 000 1 0010 0000 0 000 00 00 1100 (base two)

(a) Convert X into MIPS assembly.
First look at the first 6 bits to determine opcode
opcode 6: 000000 = 0x00 = add, addu, and, jr, nor, or... and more.
rs 5: 01001 = 1+8 = 9 = $9
rt 5: 10010 = 2 + 16 = 18 = $18
rd 5: 01101 = 1 + 4 + 8 = 13 = $13
shamt 5: 00000 = 0 = 0 = 0x0 = shift amount 0
funct 6: 100010 = 2 + 32 = 34 = 0x22 = sub
ALMOST THERE: sub $13, $9, $18
$13 = $t5, $9 = $t1, $18 = $s2

FINAL INSTRUCTION: **sub $t5, $t1, $s2**

(b) Which type (I-type, R-type, J-type) is instruction X?
R Type (see above)

(c) Convert Y into MIPS assembly.
Opcode 6: 100011 = 35 = 0x23 = lw (I_type)
rs 5: 01 000 = 8 = $t0
rt 5: 1 0010 = 18 = $s2
immediate 16: 0000 0 000 00 00 1100 = 12

FINAL INSTRUCTION: **lw $s2, 12($t0)**

(d) Which type (I-type, R-type, J-type) is instruction Y?
I Type (see above)

Kaleb J. Himes

Problem 4 [20 pts]

(a) Consider the following MIPS assembly instructions. What is a corresponding C statement?

add f, h, g

```
f = g + h;
```

sub f, i, f

```
f = i - f;
```

(b) If the variables f, g, h, and i have the values 1, 2, 3, and 5 respectively, what is the end value of f?

```
f = 2 + 3;
(f = 5)
(i = 5)
f = 5 - 5;
(f = 0)
```

(c) Now consider the following C statement. What is the corresponding MIPS assembly code?

B[8] = A[i-j];

Assume i and j are assigned to registers $s3 and $s4 respectively. Assume the base address of the arrays A and B are in registers $s6 and $s7 respectively.

```
sub $t0, $s3, $s4
 lw $t0, 9($s7) [0,1,2,3,4,5,6,7, 8]
                                  ^
                                  |
                (1,2,3,4,5,6,7,8,9)
```

Kaleb J. Himes

Problem 5 [10 pts]

```
addi $t0, $s6, 4
add $t1, $s6, $0
sw $t1, 0($t0)
lw $t0, 0($t0)
add $s0, $t1, $t0
```

For the MIPS assembly above, assume the registers $s0 and $s1 contain the values 0x0000 0014 and 0x0000 0028, respectively. Also assume register $s6 contains the value 0x0000 0200, and that memory contains the following values:

Address	Value		addr	value
0x0000 0200	0x0000 00c8	:	Array[0]	(12*16)+8 = 200
0x0000 0204	0x0000 012c	:	Array[1]	256+32+12 = 300
0x0000 0208	0x0000 0190	:	Array[2]	256+144 = 400

Find the value of $s0 at the end of the assembly code.

I'm going to make an assumption here that this is an array containing 4-byte values. Thus when we point to $s6, 4 we are pointing to the beginning of the second element that is 4-bytes beyond the first element. I will proceed with this assumption in place.

1. $t0 = $s6 + 4 (adds 4 to the memory address 0x00000200 and stores it in $t0, store Array[1] in $t0)
2. $t1 = $s6 (adds 0 to memory addr 0x00000200 and stores it in $t1, Array[0])
   $t0 = 0x...0204 (Array[1])
   $t1 = 0x...0200 (Array[0])
3. $t1 contains value 0x...00c8 = 200, store 200 in $t0 with zero offset.
4. load $t0 into itself for giggles? I guess …
   $t0 = 200
   $t1 = Array[0] = 0x...0200 = 0x...00c8 = 200

5. $s0 = 200 + Array[0] = 200 + 200 = **400**

NEXT PAGE

Kaleb J. Himes

Problem 6 [10 pts]

Assume the following register contents:

$t0 = 0x5ca1ab1e = 0101 1100 1010 0001 1010 1011 0001 1110

(a) What is the value of $t2 for the following sequence of instructions?

```
sll $t2, $t0, 4        = 1100 1010 0001 1010 1011 0001 1110 0000
andi $t2, $t2, -1        1111 1111 1111 1111 1111 1111 1111 1111
                         1100 1010 0001 1010 1011 0001 1110 0000
```

1. Shift left logical (shift the value $t0, by the amount of 4 and store the result in $t2)
2. andi $t2 with the 32 bit binary for -1 (1111 1111 1111 1111 1111 1111 1111 1111)

$t2 = 1100 1010 0001 1010 1011 0001 1110 0000 = ca1ab1e0

(assumed 2's complement since not specified)

(b) What is the value of $t2 for the following sequence of instructions?

```
srl $t2, $t0, 3          =  0000 1011 1001 0100 0011 0101 0110 0011
andi $t2, $t2, 0xf00d       0000 0000 0000 0000 1111 0000 0000 1101
                            1111 0100 0110 1011 0011 1010 1001 0001
```

1. Shift right logical on $t0, store in $t2, shift right by 3
2. andi $t2 with F00D (hahaha) and store the result in $t2

$t2 = 1111 0100 0110 1011 0011 1010 1001 0001 = f46b3a91

Problem 7 [10 pts]

Assume $t0 contains the following:

0x8000 8000 = 34,359,771,136

What is the value of $t1 after the following instructions?
Show a trace of $t1 throughout the iteration.

```
      slt $t1, $0, $t0        let $t1 hold the value for ($0 < $t0)  (either 0 or 1, T or F)
      bne $t1, $0, ELSE       if t1 does not equal 0, PC+4+address of ELSE
      j DONE                  if t1 was equal to 1 PC = address of DONE
ELSE: addi $t1, $t1, 4        if t1 was zero, add 4 to $t1 and store the result in $t1
DONE:                         in either case, halt here.
```

Kaleb J. Himes

The value of $t1 on the first step would have been 1 as 34 billion something is greater than zero, so we would bne $t1, $0, ELSE where we would add 4 to the 1 and store the result back in $t1.
At the end $t1 = 5.

Problem 8 [10 pts]

for (i = 0; i < x; i++)
    x += y;

Assume that the values of x, y and i are in registers $s5, $s6, $t1, respectively.

(a) Translate the above C code to MIPS assembly code. Use a minimum number of instructions.

```
lw $t0, $s5            # t0 is a constant holding our upper bound "x"
li $t1, 0              # t1 is our counter (i)
loop:
beq $t1, $t0, end      # if t1 == x we are done
loop body
add $s5, $s5, $s6
addi $t1, $t1, 1       # add 1 to t1
j loop                 # jump back to loop beginning
end:
```

(b) How many MIPS instructions does it take to implement the C code?
9 instructions

(c) If the variables x and y are both initialized to 1, what is the total number of MIPS instructions that is executed to complete the loop?

It's still going to be 9 instructions. Either way if we differentiate between x and y we still have to reference the register their value is contained in. We will still have to do the +=, we could change that to addi $s5, $s5, 1, but the instruction remains regardless.