University of Pittsburgh

CS/COE 447 Spring 2010 Exam 1

There are a total of 100 points. You are allowed to use the Green Card (or a copy of it) that comes with the text.

Do not use any pseudo instructions. Use only instructions listed in the core instruction set on the front of the green card, as they are specified there. That is, if you were to assemble your program, the assembler should not replace any of your instructions by two or more instructions. **Exception: you may use the la and li pseudo instructions.**

We can't answer questions like *What do you want for this question?* or *I don't understand this question.* It would be too disruptive. It would also be unfair, since some people would get extra information.

Please just use your best judgment.

Show your work for partial credit.

Each question is on its own page, to give you plenty of room. Don't feel you need to fill up each page; just write what you need to.

Good luck!!

1. (8 points)

   (a) Translate B397 hex into binary
   1011 0011 1001 0111

   (b) Translate 1 0111 1011 1001 1111 binary into hex
   17b9f

   (c) Translate 38 decimal into binary
   100110 - 32 + 6 + 2

   (d) Translate 2^8 decimal into binary
   100000000

2. (12 points) Give the machine code for the following instructions, first in binary and then in hex:

```
subu $t1,$a1,$s1
```

Binary:
000000 00101 10001 01001 00000 100011

0000 0000 1011 0001 0100 1000 0010 0011

Hex: 0x00b14823

```
sw  $a1, 16($t1)
```

Binary:

101011 01001 00101 0000 0000 0001 0000
1010 1101 0010 0101 0000 0000 0001 0000

Hex: 0xad250010

```
ori $t1, $t1, 0x7070
```

Binary:

001101 01001 01001 0111 0000 0111 0000
0011 0101 0010 1001 0111 0000 0111 0000
35297070
Hex: 35297070

3. (5 points) Give the machine code for the **beq** instruction below, first in binary and then in hex. Note that in the **beq** instruction, **$t1 is rs and $a3 is rt.**

```
Address           Source

0x0040098 len_loop: lbu  $t1,0($t0)
0x004009c           beq  $t1,$a3,len_exit
0x00400a0           addi $t0,$t0,1
0x00400a4           sub  $t1,$t1,$a0
0x00400a8           lbu  $t1,0($t0)
0x00400ac           j    len_loop
0x00400b0 len_exit: sub  $v0, $t0, $a0
0x00400b4           jr   $ra
```

```
Binary:
000100 01001 00111 0000 0000 0000 0100
0001 0001 0010 0111 0000 0000 0000 0100

Hex:
0x11270004
```

4. (26 points) Suppose memory contains the following values.

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	0xFEDCBA98	0x3338FABC	0x81234011	0x00401111	0x00000000

What value (in hex) is placed into which register or memory location by each of the following instructions?

	register or memory location	hex value	
`la $t0,label4`	`$t0 0x10010000`	1	
`lw $t1,0($t0)`	`$t1 0xfedcba98`		
`lw $t2,12($t0)`	`$t2 0x00401111`		
`addi $t2,$t2,0x934`	`$t2 0x00401A45`	1	
`sw $t2,4($t0)`	`0x10010004 0x00401A45`	1 - ok if same as previous value	
`lbu $t3,1($t0)`	`$t3 0x000000BA`		
`sb $t3,2($t0)`	`0x10010002 0xBA`	1	
`lw $t4,0($t0)`	`$t4 0xFEBABA98`	1	
`lui $t5,0x1056`	`$t5 0x10560000`		
`ori $t5,$t5,9`	`$t5 0x10560009`		
`addi $t6,$zero,0x7bF0`	`$t6 0x00007BF0`	1	
`li $s0,0xAAAAAAAA`	`$s0 0xaaaaaaaa`	1	
`and $t7,$t6,$s0`	`$t7 0x00002aa0`		
`or $t8,$t6,$s0`	`$t8 0xaaaafbfa`		
`srl $t9,$t6,3`	`$t9 0x00000f7e`		

5. (14 points) Below is a solution to Program 1, Part 2.

```
Please answer the questions marked by #Q:  There are 7 of them.
    .data
string: .space 9
    .text
        li   $a0, 0xCAB123EA
la   $a1, string
jal  translate
        j    endprogram

translate:
addi $t9, $zero, 8
addi $t8, $zero, 0
lui $t7, 0xF000
        #Q: What is $t7 now?


translate_loop:
and  $t1, $a0, $t7
        #Q: The first time through the loop, what is $t1 now?


srl  $t1, $t1, 28
        #Q: The first time through the loop, what is $t1 now?


sll  $a0, $a0, 4
        #Q: The first time through the loop, what is $a0 now?


slti $t2, $t1, 0xA
        #Q: The first time through the loop, what is $t2 now?


beq  $t2, $zero, label
        #Q: The first time through the loop, is the next instruction executed?


addi $t0, $t1, 0x30
j    translate_store
label:
addi $t0, $t1, 0x37
```

```
translate_store:

sb   $t0, 0($a1)
        #Q: The first time through the loop, what value is stored in memory?


addi $a1, $a1, 1
addi $t8, $t8, 1
bne  $t8, $t9, translate_loop
sb   $zero, 0($a1)
jr   $ra
endprogram:
       #Q: What is $t7 now?
A: 0xf0000000
        #Q: The first time through the loop, what is $t1 now?
A:0xc0000000
        #Q: The first time through the loop, what is $t1 now?
A:0x0000000c
        #Q: The first time through the loop, what is $a0 now?
A: 0xab123ea0
        #Q: The first time through the loop, what is $t2 now?
A:0
        #Q: The first time through the loop, is the next instruction executed?
A: no
sb   $t0, 0($a1)
        #Q: The first time through the loop, what value is stored in memory?
A: 0x43 (one byte)
```

6. (15 points) Assume $t0 has been assigned some value. Write MIPS code that sets bits 1, 4, and 8 (makes them 1) in $t0. The code should preserve the contents of all the other bits in $t0.

ori $t0, $t0, 0x112

It is 0x112 because the 1's below are bits 1, 4, and 8 1 0001 0010

7. (20 points) Assume the following:

```
.data
k: .word 2
array: .word 87,22,88,777,23,89,845
```

Write MIPS assembly-language instuctions to accomplish the following pseudo-code segment. k and array in the pseudo code refer to the above variables stored in memory. You must use those labels in your code. And, your code should work even if we were to change the values stored in k and array.

```
if (array[k] >= 0):
   array[0] = 10;
else:
   array[1] = 50;
```

```
.data
k: .word 2
array: .word 87,22,88,777,23,89,845

.text
# get array[k]
la $t0,k
lw $t0,0($t0)
sll $t0,$t0,2
la $t1,array
add $t0,$t0,$t1
lw $t0,0($t0)

la $t5,array

# is array[k] >= 0?
slt $t0,$t0,$zero
# $t0 is 0 if it is
bne $t0,$zero,else
#Stay here to execute the ‘‘then’’ clause
li $t6,10
sw $t6,0($t5)
j end
else:
li $t6, 50
sw $t6,4($t5)
```

```
end:
```