MIPS Control Flow

1) What are the instructions to branch on each of the following conditions?

`$s0 < $s1`	`$s0 <= $s1`	`$s0 > 1`	`$s0 >= 1`

2) Complete the MIPS so that it flows like the C.

```
// Strcpy:
// $s1 -> char s1[] = "Hello!";
// $s2 -> char *s2 =
//        malloc(sizeof(char)*7);
int i=0;
do {
    s2[i] = s1[i];
    i++;
} while(s1[i] != '\0');
s2[i] = '\0';
```

```
      addiu $t0, $0,  0
Loop: addu  $t1, $s1, $t0  # s1[i]
      addu  $t2, $s2, $t0  # s2[i]
      lb    $t3, 0($t1)    # char is
      sb    $t3, 0($t2)    # 1 byte!
      addiu $t0, $t0, 1
      addiu $t1, $t1, 1

      __________________________

      __________________________

      __________________________
Done: addiu $t2, $t2, 1
      sb    $t4, 0($t2)
```

```
// Nth_Fibonacci(n):
// $s0 -> n, $s1 -> fib
// $t0 -> i, $t1 -> j
// assume the following values
// are in registers already
int fib = 1, i = 1, j = 1;

if(n==0)      return 0;
else if(n==1) return 1;
n-=2;
while(n != 0) {
    fib = i + j;
    j = i;
    i = fib;
    n--;
}
return fib;
```

```
      ...
      __________________________

      __________________________

      __________________________
      addiu $s0, $s0, -2

Loop: __________________________
      addu $s1, $t0, $t1
      addiu $t0, $t1, 0
      addiu $t1, $s1, 0
      addiu $s0, $s0, -1
      j    Loop
Ret0: addiu $v0, $0,  0
      j    Done
Ret1: addiu $v0, $0,  1
      j    Done
RetF: addu  $v0, $0,  $s1
      ...
Done: jr   $ra
```

Instruction Formats

R-Instruction format (register-to-register) *Examples: addu, and, sll, jr*

op code	rs	rt	Rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

See green sheet to see what registers are read from and what is written to

I-Instruction Format (register immediate) *Examples: addiu, andi, bne*

op code	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

Note: Immediate is 0 or sign-extended depending on instruction (see green sheet)

J-Instruction Format (jump format) *For j and jal*

op code	Address
6 bits	26 bits

KEY: An instruction is R-Format if the opcode is 0. If the opcode is 2 or 3, it is J-format. Otherwise, it is I-format. Different R-format instructions are determined by the "funct".

1. How many total possible instructions can we represent with this format?

2. What could we do to increase the number of possible instructions?

MIPS Addressing Modes

3. You need to jump to an instruction that 2^28 + 4 bytes higher than the current PC. How do you do it? (HINT: you need multiple instructions)

4. You now need to branch to an instruction 2^17 + 4 bytes higher than the current PC, when $t0 equals 0. Assume that we're not jumping to a new 2^28 byte block. Write MIPS to do this.

5. Given the following MIPS code (and instruction addresses), fill in the blank fields for the following instructions (you'll need your green sheet!):

```
0x002cff00: loop: addu $t0, $t0, $t0      |  0 |    |    |    | 0 |   |
0x002cff04:       jal  foo                |  3 |                      |
0x002cff08:       bne  $t0, $zero, loop   |  5 |  8  |                |
...
0x00300004: foo:  jr $ra                 $ra=______________________
```

6. What instruction is `0x00008A03`?