Name:____

Problem 1 (10 points) MIPS Function Conventions

Consider the following MIPS code:

```
main: ...          #Save some registers here
addi $t0,$0,45
addi $s0,$0,28
...                #Save some registers here
addi $a0,$0,12
jal compute
...                #Restore some registers here
add $s0,$t0,$s0
li $v0,4
...                #Restore some registers here
jr $ra
```

Fill in the table describing whether or not you must save/restore each register before/after the function call. Provide a concise reason why or why not.

Register	Must Save/Restore (Yes/No)	Reason
$t0		
$s0		
$a0		
$v0		
$ra		

Name:____

Problem 2 (10 points) Pseudoinstructions

Convert the following MIPS pseudoisntructions into the corresponding sequence of real instructions. Use register $at to store any temporary values.

a) Branch if greater than: `bgt $t0, $t1, label`

b) Branch if less than: `blt $t0, $t1, label`

c) Branch if greater than or equal: `bge $t0, $t1, label`

d) Branch if less than or equal: `ble $t0, $t1, label`

Name:____

Problem 3 (20 points) Pointers

Convert the following C code to MIPS. You must follow all MIPS function conventions.

```
void swap(int **a)
{
     int temp = *(*a+1);
     *(*a+1) = **a;
     **a = temp;
}
```

Name:____

Problem 4 (25 points): Towers of Hanoi

a) (15 points) Convert the following C code into MIPS. Do not use pseudoinstructions and follow all MIPS function calling conventions.

```
/*
      Towers of Hanoi
      DESCRIPTION:
            Given three pegs, one with a set of N disks of
            increasing size, determine the minimum (optimal)
            number of steps it takes to move all the disks from
            their initial position to another peg without placing
            a larger disk on top of a smaller one.
      INPUT:
            Unsigned integer N
      OUTPUT:
            Unsigned integer denoting the number of steps

 */
unsigned int hanoi(unsigned int N)
{
      if(N == 1)
            return 1;
      else
            return 2*hanoi(N-1) + 1;

}
```

Name:____

b) (5 points) How many bytes does your code require? Extra credit for the smallest correct code.

c) (5 points) What is the maximum number of `hanoi` frames that you have on your stack at any given time when the initial function call is `hanoi(4)`?

Name:____

Problem 5 (35 points) MIPS Reverse Compile

Consider the following MIPS assembly code:

```
calculon:        addi  $t0, $zero, 1
                 add   $v0, $zero, $zero
clamps:          slt   $at, $t0, $a1
                 beq   $at, $zero, flexo
                 sll   $t1, $t0, 2
                 add   $t1, $t1, $a0
                 lw    $t2, 0($t1)
                 lw    $t3, -4($t1)
                 slt   $at, $t3, $t2
                 beq   $at, $zero, scruffy
                 add   $t0, $t0, 1
                 j     clamps
flexo:           addi  $v0, $zero, 1
scruffy:         jr    $ra
```

a) **(10 points)** Translate the function `calculon` above into a high-level language like C or Java. Your function header should list the types of any arguments and return values. Also, your code should be as concise as possible, without any gotos or pointer arithmetic. We will not deduct points for syntax errors unless they are significant enough to alter the meaning of your code.

b) **(5 points)** Describe briefly, in English, what this function does.

c) **(20 points)** Convert the following instructions from the code above into 32 bit hexadecimal number. Assume that the address of the first instruction (`addi $t0, $zero, 1`) is located at address `0x00400018`.

(5 points) `beq $at, $zero, scruffy`

(5 points) `j clamps`

(5 points) `lw $t3, -4($t1)`

(5 points) `sll $t1, $t0, 2`