Name:____

Problem 1 (50 points) Short Answers

Convert the decimal number -3 to:

a) **(5 points)** A 32 bit two's complement number in hexadecimal form

b) **(5 points)** A 32 bit one's complement number in hexadecimal form

c) **(5 points)** A 32 bit sign and magnitude number in hexadecimal form

d) **(5 points)** Assuming the MIPS memory model, i.e. memory is byte addressed, the processor works on 32-bit (word) data and word memory accesses must be word aligned, which of the following hexadecimal memory addresses accesses are valid?

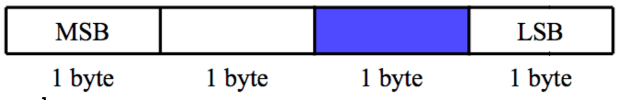
1) 0x12345678
2) 0x24A19E8A
3) 0xAAABBC23
4) 0x84F2FFED
5) 0x00004400

e) **(20 points)** Convert the following C code to MIPS

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

Name:____

f) (10 points) Write the body of a function in either C or MIPS that returns the second least significant byte (shown shaded in the picture) of a 32-bit integer using bit-wise logical operations and shifts. No control flow (e.g., loops/ifs) are needed.



```
char get_2nd_byte(int x) {





}
```

Problem 2 (40 points): Binary Search

a) (35 points) Convert the following C code into MIPS. Do not use pseudoinstructions and follow all MIPS function calling conventions.

```
/*
   Binary Search Algorithm.
   INPUT:
        data is a array of integers SORTED in ASCENDING order,
        toFind is the integer to search for,
        start is the minimum array index,
        end is the maximum array index
   OUTPUT:
        position of the integer toFind within array data,
        -1 if not found
 */

int binary_search(int *data, int toFind, int start, int end)
 {
    //Get the midpoint.
    int mid = start + (end - start)/2;   //Integer division

    //Stop condition.
    if (start > end)
       return -1;
    else if (data[mid] == toFind)      //Found?
       return mid;
    else if (data[mid] > toFind)       //Data is greater than
                                       //toFind, search lower half
       return binary_search(data, toFind, start, mid-1);
    else                               //Data is less than toFind,
                                       //search upper half
       return binary_search(data, toFind, mid+1, end);
 }
```

Name:____

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

Name:____

Problem 3 (45 points) Understanding MIPS Programs
Consider the following MIPS assembly code:

```
scooby:
     ori $t0, $0, 0
     ori $t1, $0, 0
     lui $at, 0x3FFF
     ori $t2, $at, 0xFFFF
shaggy:
     slt $at, $t0, $a1
     beq $at, $0, velma
     ori $at, $0, 4
     mul $t3, $t0, $at
     add $t3, $a0, $t3
     lw $t3, 0($t3)
     slt $at, $t1, $t3
     beq $at, $0, fred
     addu $t1, $0, $t3
fred:
     slt $at, $t3, $t2
     beq $at, $0, daphne
     addu $t2, $0, $t3
daphne:
     addi $t0, $t0, 1
     j shaggy
velma:
     sw $t1, 0($a2)
     sw $t2, 0($a3)
     jr $ra
```

a) **(20 points)** Translate the function `scooby` 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.

Name:____

c) **(20 points)** Convert the following instructions from the code above into 32 bit hexadecimal number. Assume that the address of the first instruction (`ori $t0, $0, 0`) is located at address `0x00400028`.

(5 points) `beq $at, $0, velma`

(5 points) `j shaggy`

(5 points) `lw $t3, 0($t3)`

(5 points) `slt $at, $t3, $t2`