Name:____

Problem 1: (40 points) Short Answers (20 minutes)

1. **(5 points)** I have N bits to represent data, and every bit pattern has a unique meaning. I want to represent 3 times as many things. How many more bits do I need?

2. **(10 points)** We are given two nibbles, A (=`0xF`) and B (=`0b0010`), and we wish to calculate their `SUM = A + B`. We only have a nibble to store the SUM result. What is SUM if all three nibbles (A, B, SUM) were:
   Algorithm: SUM = encode(decode-into-decimal(A) + decode-into-decimal(B))

	SUM (Single hex Character)	Decimal number SUM encodes	Was there overflow?
…sign magnitude?	0x		
…ones complement?	0x		
…unsigned?	0x		
…twos complement?	0x		

3. **(10 points)**: Write MIPS assembly code that takes an integer variable `in` and transforms it such that the nth bit inverted (*i.e.,* 0->1, 1->0). This can be done using just bit-wise logical operations and shifts. No control flow instructions are needed. Assume the variable `in` is stored in `$a0` and the variable `n` is stored in `$a1`. You should put the result in register `$v0`.

   Example: flip_nth_bit(0x04F2, 7)→ 0x0472

   0 4 F 2
   0000 0100 **1**111 0010

   $\Downarrow$

   0000 0100 **0**111 0010
   0 4 7 2

Name:____

4. **(10 points):** Write the MIPS assembly code for the following pieces of C code, assuming the following declarations:

```
int A[32];
int **b;
int *c;
int i;
```

and the base address of the array `A` is in register `$s0`, the variable `b` is in register `$s1`, the variable `i` is in register `$s2` and the variable c is stored in register `$s3`

`a)`**(5 points)** `A[i] = **b;`

b) **(5 points)** `c = &A[7];`

5. **(2.5 points)** Which is not an accurate representation of the number -3?
   a) 0x83
   b) 0xFE
   c) 0xFD
   d) 0xFC
   e) All of the above are correct

6. **(2.5 points)** What is the smallest hexadecimal number that makes the following true: `0x_________ | 0x01431856 = 0x13579BDF`?
   `a) 0x13569B9D`
   `b) 0x1254838F`
   `c) 0x12148389`
   `d) 0x11148389`
   `e) 0x13579BAF`

Name:____

Problem 2: (25 points) Understanding MIPS Programs (20 minutes)

```
        addi    $s0, $0, 0
        add     $s1, $0, $0
        add     $s2, $0, $0
jane:   slt     $t0, $s2, $s3
        beq     $t0, $0, george
        sll     $t0, $s2, 2
        add     $t0, $t0, $s5
        lw      $t0, 0($t0)
        andi    $t0, $t0, 0x0001
        bne     $t0, $0, elroy
        addi    $s1, $s1, 1
        j judy
elroy:  addi    $s0, $s0, 1
judy:   addi    $s2, $s2, 1
        j jane
george: …
```

a) (20 points): Translate the assembly code above into C. You should include a header that lists the types of any variables. Also, your code should be as concise as possible, without explicit pointers. We will not deduct points for syntax errors unless they are significant enough to alter the meaning of your code. You are not allowed to use go to statements as CSE legend Edsger Dijkstra considered them harmful. He was a very smart man; you are strongly advised to listen to him.

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

Name:____

Problem 3: (20 points) Data Structures (20 minutes)

The strcmp functions compares one string to another. It is defined as:

```
int strcmp( const char *str1, const char *str2 )
```

It performs an unsigned comparison of *s1* to *s2*. It starts with the first character in each string and continues with subsequent characters until the corresponding characters differ or until the end of the strings is reached. strcmp returns a value that is:

< 0 if *s1* is less than *s2*

== 0 if *s1* is the same as *s2*

> 0 if *s1* is greater than *s2*

More precisely, if the strings differ, the value of the first nonmatching character in *s2* subtracted from the corresponding character in *s1* is returned.

(5 points) Implement this function in C

(10 points) Translate your function above into MIPS assembly

Name:____

Problem 4: (15 points) Simple Arithmetic (20 minutes)

Translate the following code into MIPS assembly:

Z = (A+B) * (C+D) * (E+F)

Assume that variables A-F, Z are in registers $s0-$s6, respectively (e.g. C = $s2). Also, assume that you cannot overwrite variables A-F since they will all be used later in the program.

For this question, you can use the "mul" pseudo-instruction. It has this form:

```
mul $1, $2, $3, where $1 = $2 * $3.
```

It does the same thing as:

```
mult $2, $3
mflo $1
```

Do not use mult and mflo.

a) Assume that you have sequential processor where add and mul take 1 and 3 cycles, respectively. Write the code such that it uses the <u>minimum</u> number of registers and cycles. How many additional registers (other than $s0-$s6) does your code require? How many cycles does your code need?

Name:____

b) Assume that you have a different type of processor architecture (call it VLIW) that can perform one ADD operation and one MULTIPLY operation during every cycle. Rewrite the code to take advantage of this and use the minimum number of cycles. The VLIW processor uses a faster implementation of MULTIPY, which only takes 1 cycle, the same as the cycle time of ADD. To make it easier to grade, please write mul and add that are executed in the same cycle on one line, for example:

```
mul $1, $2, $3          add $4, $2, $8
```

How many cycles does your code need?