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 6 times as many things. How many more bits do I need?

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

3. **(5 points)** What is the smallest hexadecimal number that makes the following true: `0x_____ | 0x01431856 = 0x13579BDF`?

4. **(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		

Name:_____

5. **(5 points)** Suppose we have defined the C structure:

```
struct player {
    int id;
    int numGoals;
    char name[8];
};
```

Also, we declare: `struct player players[3];`
such that `players` starts at `0x10000000`.

What is the value of `playerTwo` after:
`struct player *playerTwo = players + 2;`
Write your answer in hexidecimal.

6. **(10 points)**: Write MIPS assembly code that takes an integer variable `in` and transforms it such that the n^{th} bit is set (*i.e.,* it is changed to '1') while the remaining bits stay the same. This must 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`.

   Examples:
   set_nth_bit(0x00000000, 7)→ 0x00000080
   set_nth_bit(0xABCDEF0, 4)→ 0xABCDEF0 (4^{th} bit was already set (= 1))
   set_nth_bit(0x11111111, 9)→ 0x11111311

Name:_____

Problem 2: (30 points) Pointers and Arrays (15 minutes): Write the MIPS assembly code for the following pieces of C code, assuming the following declarations:

```
int A[10] = {0,1,2,3,4,5,6,7,8,9};
int *b;
int *c;
int i = 4;
```

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`. Assume that the initial value of `A` is `0x5fbffa60`.

a) **(5 points)** `b = &A[1];`

b) **(5 points)** `c = &A[i+2];` (Note: you cannot assume that i = 4 when translating to MIPS; you must write the MIPS code exactly corresponding to the C statement above.)

c)**(5 points)** `i = *c;`

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

Name:_____

After the execution of all the code above in the order that it appears:
e)**(3 points)** What are the values of each element in the `A` array?

A[10] ={ , , , , , , , , , }

f)**(3 points)** What is the value of `b` in hex?

g)**(3 points)** What is the value of `c` in hex?

h)**(1 points)** What is the value of `i` in decimal?

Problem 3: (15 points) String Manipulation (15 minutes)

The `strspn` function returns the index of the first character in str1 that doesn't match any character in str2

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

In other words, the function returns length of the initial portion of str1 containing only characters that appear in str2. Therefore, if all of the characters in str1 are in str2, the function returns the length of the entire str1 string, and if the first character in str1 is not in str2, the function returns zero.

Example:

```
int main ()
{
  int i;
  char strtext[] = "129th";
  char cset[] = “0123456789";

  i = strspn (strtext,cset);
  printf ("The length of initial number is %d.\n", i);
  return 0;
}
```

Output:

```
The length of initial number is 3.
```

```
If strtext = 301t29h and cset = “13245t76890”, the return value
is 6.
```

Name:_____

(15 points) Implement this function in C

Name:_____

Problem 4: (15 points) 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) **(5 points)** Assume that you have sequential processor where add and mul take 1 and 3 cycles, respectively. Write the code such that it uses the minimum 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) **(10 points)** 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?