Computer Architecture

Assignment 1

Problem 1

Look up the Antikythera Mechanism and write a few sentences about what it is. Some claim this to be the most ancient computer known. What do you think? Does it satisfy the definition of computer used in this course?

Problem 2

Show the bit fields and give the binary and hex code for the instruction `sub $s0, $s1, $s2`.

Problem 3

Show the bit fields and give the binary and hex code for the instruction `beq $t1, $t4, L2` if the `L2` label is located 12 instructions before the `beq`.

Problem 4

Compile the following C++ code into MIPS:

```
if (x > 2*y)
    x++;
else
    y--;
```

Assume `x` is in `$s0` and `y` is in `$s1`. Write the MIPS code and then write the hex code for it. (MARS can help with this but I'd suggest using it to check your work rather than blindly copying. In particular, make sure you can figure out the branch offsets on your own.)

Problem 5

What assumptions about the data types of `x` and `y` did you make in Problem 4?

Problem 6

Write a MIPS program to multiply two 2x2 matricies. Use the following data:

```
      .data
A:    .word   1 2 3 4
B:    .word   5 6 7 8
n:    .word   2
```

The matrices are stored in row-major order (this means $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ and $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$). You may add any other data you may need. Draw upon your software engineering skills to try to make this program as general as possible. It should be able to multiply 3x3 and 4x4 matricies by just changing n and adding more data in A and B. (If you can't do that, don't worry too much and just hard code the 2x2 multiplication, but it's worth trying.)

You may assume the matrix elements are unsigned and small enough not to cause overflow. Use the **`multu $t0, $t1`** instruction to multiply two registers (you can use any registers as the operands, it doesn't have to be **`$t0`** and **`$t1`**) and then **`mflo $t9`** to move the product into **`$t9`** (again, it doesn't have to be **`$t9`**, you are indeed free to move the product wherever your heart desires.)

Use MARS to write and test the program and submit your .asm file via Sakai. Please name your file with the first five letters of your last name followed by 01. For example, my file would be called seiff01.asm.