CSCI-21 Programming Assignment #1, due 2/14/22

In the Settings menu of QTSpim set Bare Machine ON, Allow Pseudo Instructions OFF, Load Exception Handler OFF, Delayed Branches OFF, Delayed Loads OFF, Mapped IO OFF, Load Exception File OFF. Email me the source file(s) attached to an email with "`CS21 Your Name Assign 1`" in the subject line. For these programming exercises, you may use ***ONLY*** these instructions (you will not need all of these):

```
and nor or ori sll srl xor
```

Run the programs by verifying the value of the PC is `0x00400000` (it defaults there if you set it in the options in QTSpim, or set it by right clicking on the “PC =” register field in the “Int Regs [16]” pane of the QTSpim window and selecting “Change Register Contents”). Then, single step (pushing F10) or by multiple step (push F11 and enter a number of steps), observing the results in the SPIM register display window. For these, try to use as few registers as possible.

Exercise 1:
Start a program with the instruction that puts a single '1' bit into the low order bit of register eight (`$8` or `$t0`, but use `$t0`):

```
        ori     $t0, $zero, 0x01
```

Now, by using only shift logical instructions and register to register logic instructions (you may not use any other immediate instructions), put the pattern `0x99999999` into register `$t1`. You may not use another `andi`, `ori` or `xori` instruction to set another bit from scratch: you must work only from the single '1' bit you set in the first instruction or from bits created directly from that '1' bit. You will need to use more registers than just `$t1`. See how few instructions and registers you need to do this. Doing this in approximately twelve instructions and using four or five registers is reasonable. You can do this in fewer. Again, you MAY NOT create any other '1' bits from "scratch" for this: you may use only the single '1' bit you started with or bits created directly from that single '1' bit.

Exercise 2:
Again, start a program with the instruction that puts the single '1' bit into register `$t0`:

```
        ori     $t0, $zero, 0x01
```

Now, by using only shift logical instructions and register to register logic instructions (use NO more immediate instructions), put the pattern `0xA5A5A5A5` into register `$t1`. Do not use another `andi`, `ori` or `xori` instruction. You will need to use other registers than just `$t1`. See how few instructions you need to do this. Again, you can do this in approximately ten to twelve instructions. Again, you may only work from the initial '1' bit or bits created directly from that single '1' bit.

Exercise 3:
Again, start a program with the instruction that puts the single '1' bit into register `$t0`:

```
        ori     $t0, $zero, 0x01
```

Now, by using only shift logical instructions and register to register logic instructions (use NO more immediate instructions), put the pattern `0xFFFFFFFF` into register `$t1`. Do not use another `andi`, `ori` or `xori` instruction. You will need to use other registers than just `$t1`. **Do not destroy the '1' bit you set in the first instruction.** See how few instructions you need to do this. You can do this in very few instructions. (You can actually do this in one instruction, but you cannot do it that way for this because of the restriction that you must work from the original '1' bit set in the first instruction). Again, you may only work from the one initial '1' bit or bits created directly from that single '1' bit.