LSU EE 4720 **Homework 2** **Due: 21 February 2022**

Problem 1: The code fragment below was taken from the course hex string assembly example. (The hex string example was not covered this semester. The full example can be found at `https://www.ece.lsu.edu/ee4720/2022/hex-string.s.html`.) The fragment below converts the value in register `a0` to an ASCII string, the string is the value in hexadecimal (though initially backward).

```
LOOP:
        andi $t0, $a0, 0xf    # Retrieve the least-significant hex digit.
        srl $a0, $a0, 4       # Shift over by one hex digit.
        slti $t1, $t0, 10     # Check whether the digit is in range 0-9
        bne $t1, $0, SKIP     # Don't forget that delay slot insn always exec.
        addi $t2, $t0, 48     # If 0-9, add 48 to make ASCII '0' - '9'.
        addi $t2, $t0, 87     # If 10-15, add 87 to make ASCII 'a' - 'z'.
SKIP:
        sb $t2, 0($a1)        # Store the digit.
        bne $a0, $0, LOOP     # Continue if value not yet zero.
        addi $a1, $a1, 1      # Move string pointer one character to the left.
```

(*a*) Show the encoding of the MIPS `bne t1, 0, SKIP` instruction. Include all parts, including—especially—the immediate. For a quick review of MIPS, including the register numbers corresponding to the register names, visit `https://www.ece.lsu.edu/ee4720/2022/lmips.s.html`.

(*b*) RISC-V RV32I has a `bne` instruction too, though it is not exactly the same. Show the encoding of the RV32I version of the `bne t1, 0, SKIP` instruction. For this subproblem assume that the `bne` will jump ahead two instructions, just as it does in the code sample above.

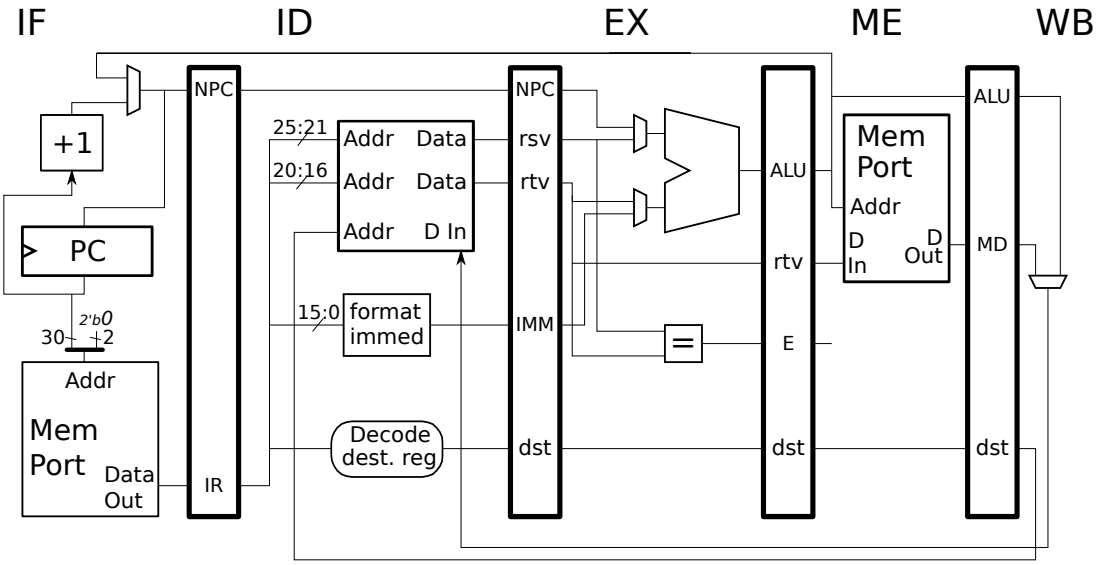
To familiarize yourself with RISC-V start by reading Chapter 1 of Volume I of the RISC-V specification, especially the Chapter 1 Introduction and Sections 1.1 and 1.3. Skip Section 1.2 unless you are comfortable with operating system and virtualization concepts. Other parts of Chapter 1 are interesting but less relevant for this problem. Also look at Section 2.5 (Control Transfer Instructions). The spec can be found in the class references page at `https://www.ece.lsu.edu/ee4720/reference.html`.

(*c*) Consider the four-instruction sequence from the code above:

```
        slti $t1, $t0, 10     # Check whether the digit is in range 0-9
        bne $t1, $0, SKIP     # Don't forget that delay slot insn always exec.
        addi $t2, $t0, 48     # If 0-9, add 48 to make ASCII '0' - '9'.
        addi $t2, $t0, 87     # If 10-15, add 87 to make ASCII 'a' - 'z'.
SKIP:
```

Re-write this sequence in RISC-V RV32I, and take advantage of RISC-V branch behavior to reduce this to three instructions (plus possibly one more instruction before the loop). For this problem one needs to focus on RISC-V branch behavior. Assume that the RISC-V `slti` and `addi` instructions are identical to their MIPS counterparts at the assembly language level. It is okay to retain the MIPS register names. *Hint: One change needs to be made for correctness, another for efficiency.*

Problem 2: *Note: The following problem was assigned in each of the last six years, and its solution is available. DO NOT look at the solution unless you are lost and can't get help elsewhere. Even in that case just glimpse.* Appearing below are **incorrect** executions on the illustrated implementation. For each one explain why it is wrong and show the correct execution.



(*a*) Explain error and show correct execution.

```
# Cycle          0  1  2  3  4  5  6  7
lw r2, 0(r4)     IF ID EX ME WB
add r1, r2, r7      IF ID EX ME WB
```

(*b*) Explain error and show correct execution.

```
# Cycle          0  1  2  3  4  5  6  7
add r1, r2, r3   IF ID EX ME WB
lw r1, 0(r4)        IF ID -> EX ME WB
```

(*c*) Explain error and show correct execution.

```
# Cycle          0  1  2  3  4  5  6  7
add r1, r2, r3   IF ID EX ME WB
sw r1, 0(r4)        IF ID -> EX ME WB
```

(*d*) Explain error and show correct execution.

```
# Cycle          0  1  2  3  4  5  6  7
add r1, r2, r3   IF ID EX ME WB
xor r4, r1, r5      IF ----> ID EX ME WB
```