CPS311 - COMPUTER ORGANIZATION

Two Short Examples Using the Simulated Multicycle Implementation

1. This program will add 1 to the contents of memory location 1000. (The demo sets this location to 42 to begin with)

```
AL:             lw       $2, 1000($0)
Hex ML:         address 00000000: 8c021000
 Cycle == 1:    ALUInputA ← register[0], ALUInputB ← immediate value
 Cycle == 2:    ALUOutput ← ALUInputA + ALUInputB
 Cycle == 3 (opcode == lw): register[2] ← M[ALUOutput]

AL:             addi     $2, $2, 1
Hex ML:         address 00000004: 20420001
 Cycle == 1:    ALUInputA ← register[2], ALUInputB ←  immediate value
 Cycle == 2:    ALUOutput ← ALUInputA + ALUInputB
 Cycle == 3:    register[2] ← ALUOutput

AL:             sw       $2, 1000($0)
Hex ML:         address 00000008: ac021000
 Cycle == 1:    ALUInputA ← register[0], ALUInputB ← immediate value
 Cycle == 2:    ALUOutput ← ALUInputA + ALUInputB
 Cycle == 3     (opcode == sw): M[ALUOutput] ← register[2]
```

2. Part 1 of Lab 5 - sum up the integers from 1 to n - n initially in $4; result ends up in $2 (Initial version without check for n = 0). Since we don't have a test driver, we'll set the initial value of $4 manually, and use a "marker" at end. Nops are not needed for this non-pipelined example.

```
AL:             addu     $2, $0, $0
Hex ML:         address 00000000: 00001021
 Cycle == 1:    ALUInputA ← register[0], ALUInputB ← register[0]
 Cycle == 2:    ALUOutput ← ALUInputA + ALUInputB
 Cycle == 3:    register[2] ← ALUOuput

AL:   loop:     addu     $2, $2, $4
Hex ML:         address 00000004: 00441021
 Cycle == 1:    ALUInputA ← register[2], ALUInputB ← register[4]
 Cycle == 2:    ALUOutput ← ALUInputA + ALUInputB
 Cycle == 3:    register[2] ← ALUOuput
                                           [ The original program used addiu,
AL:             addi     $4, $4, -1          but simulator only has addi ]
Hex ML:         address 00000008: 2084ffff
 Cycle == 1:    ALUInputA ← register[4], ALUInputB ← immediate value
 Cycle == 2:    ALUOutput ← ALUInputA + ALUInputB
 Cycle == 3:    register[4] ← ALUOutput

AL:             bne      $4, $0, loop
Hex ML:         address 0000000c: 1480fffd
 Cycle == 1:    (opcode == bne && register[rs] != register[rt]) :
                PC ← PC + sign-extend(I constant) * 4

"marker"        address 00000010: 1000ffff (infinite loop)
```