Lab Sheet 4 for CS F342 Computer Architecture

Semester 1 – 2019-20

Goals for the Lab: We build up on Lab 2 and Lab 3 and explore control (conditional) statements and loops. We also explore logical operations.

New Instructions:

1. bne application (beq, bgtz, bltz, bgez, blez are the other conditional branch opcodes)
2. Two groups of instructions: Branches and unconditional jump instructions.
3. Below are the list of instructions where the branching address is within 16 bits and register values are 32 bit
   - **beq/ bne** Branches if the quantities of two registers are equal / Not Equal.
   - **bne** Branches if the quantities of two registers are NOT equal.
   - **bgtz / bgez / bltz / blez** Branches if a quantity in a register is greater than zero / greater
   - than or equal to zero / less than zero / less than or equal to zero.

4. Below are the list of instructions where the branching address is within 26 bits.
   - **j** Jump to an address
   - **jal** Jump to an address, and store the return address in a register (for functions to be covered later).

5. Below are the list of instructions where the branching address is within 32 bits.
   - **jr** Jump to an address stored in a register
   - **jalr** Jump to an address stored in a register, and store the return address in another register (for functions to be covered later).

Exercise 1: Write MIPS Assembly code to find whether the input integer is even or odd.

Hint: Build up from Lab sheet 2, to read and integer into a register and then **andi** it with 0x1 (test for last bit as zero). Thereafter jump to report even if the result is zero.

```
# if $a0 has the value
andi $a0, $a0, 1
bne $a0, $zero, odd_label
# print that the value is even
j exit_label
odd_label:
     # print that the value is odd
exit_label:
      li $v0, 10 # exit program
      syscall
```

Exercise 2: Write MIPS Assembly code to find the length of the given string. Presently the string is coming from labelled data. Modify the code below to use **j** (Jump) and **beq** instead of **bne**.

```
.data
mystr : .asciiz "This is given string"

.text
main: la $a0, mystr # find the strlen for string at this address
      li $v0, -1    # $v0 has length start at 1 because '\0' counted below

slen_0: lb $t0, ($a0)     # load current byte, move to next
        addi $a0, $a0, 1
        addi $v0, $v0, 1 # but first increment result
        bne $t0, $zero, slen_0 # if current byte isn't '\0', repeat

      move $a0, $v0          # display result of strlen
      li $v0, 1
      syscall
      li $v0, 10 # exit program
      syscall
```

Exercise 3: Write MIPS Assembly code to print all the multiples of the given number between 0 and 100. Your program should allow the user to give the input number. Sample test case : Input: 25 : Output: 25 50 75

Hint: For loop implementation you can use pseudo instructions given below.

Pseudoinstruction	Translation
bge $rt, $rs, LABEL	slt $t0, $rt, $rs beq $t0, $zero, LABEL
bgt $rt, $rs, LABEL	slt $t0, $rs, $rt bne $t0, $zero, LABEL
ble $rt, $rs, LABEL	slt $t0, $rs, $rt beq $t0, $zero, LABEL
blt $rt, $rs, LABEL	slt $t0, $rt, $rs bne $t0, $zero, LABEL

Exercise 4: Decode (disassemble) the following instructions.

1. 12640007
2. 08100010
3. 08100010
4. 1d600004
5. 05400003
6. 05210002
7. 19000001