Laboratory 3

Load/ Store , Jump & Branch instructions

Goals

After this laboratory exercise, you can know how to use load/store, jump and branch instructions. You can also implement high level programming language structures such as if-then-else, for-loop and switch-case statements in assembly.

Preparation

Before starting this laboratory, you should review the textbook and the previous laboratory exercises.

Assignments

Sample Code 1

This sample code implements "if-then-else" statement using some fundamental instructions, such as slt, addi, jump and branch.

if (i<=j)
x=x+1;
z=1;
else
y=y-1;
*z=2*z;*

Read this example carefully and try to clarify the function of each intructions by adding comments to each instruction. You should draw the algorithm chart for the if-then-else statement.

```
#Laboratory Exercise 3, Sample Code 1
start:
      slt   $t0,$s2,$s1       # j < i ?
      bne   $t0,$zero,else    # …
      addi  $t1,$t1,1         # …
      addi  $t3,$zero,1       # …
      j     endif             # …
else: addi  $t2,$t2,-1        # …
      add   $t3,$t3,$t3       # …
endif:
```

Sample Code 2

The following pseudo code shows the implementation of a loop statement that computes the sum of *n* elements of array A.

loop: *i = i + step;*
sum = sum + A[i];
if(i != n) goto loop;

Read and try to understand the following sample code, which implements the above loop statement in assembly. Given that the index i, the starting address of A, the element number of A, the step and the sum are stored $s1, $s2, $s3, $s4 and $s5, respectively.

```
#Laboratory 3, Sample Code 2
.text
loop: add   $s1,$s1,$s4       #i=i+step
      add   $t1,$s1,$s1       #t1=2*s1
      add   $t1,$t1,$t1       #t1=4*s1
      add   $t1,$t1,$s2       #t1 store the address of A[i]
      lw    $t0,0($t1)        #load value of A[i] in $t0
      add   $s5,$s5,$t0       #sum=sum+A[i]
      bne   $s1,$s3,loop      #if i != n, goto loop
```

Sample Code 3

A switch-case statement is used to change the control flow of program execution depending on the value of a variable or expression. In the following switch-case statement, the value of *test* can be 0, 1, or 2; each value of *test* corresponds to a different action.

```
switch(test) {
        case 0:
                a=a+1; break;
        case 1:
                a=a-1; break;
        case 2:
                b=2*b; break;
        }
```

Read and try to understand the following sample code, which implements the above switch/case statement. Assuming that **a** and **b** are stored in $s2 and $s3.

```
#Laboratory Exercise 3, Sample Code 3
.data
test: .word 1
.text
            la    $s0, test   #load the address of test variable
            lw    $s1, 0($s0) #load the value of test to register $t1
            li    $t0, 0      #load value for test case
            li    $t1, 1
            li    $t2, 2
            beq   $s1, $t0, case_0
            beq   $s1, $t1, case_1
            beq   $s1, $t2, case_2
            j     default
case_0:     addi  $s2, $s2, 1       #a=a+1
            j     continue
case_1:     sub   $s2, $s2, $t1     #a=a-1
            j     continue
case_2:     add   $s3, $s3, $s3     #b=2*b
            j     continue
default:
continue:
```

Assignment 1

Create a new project to implement the code in Sample Code 1. Initialize the i and j variables. Run the code on the MARS simulator step by step and observe the change in the memory and registers.

Assignment 2

Create a new project to implement the code in Sample Code 2. Initialize the i, n, step, sum variables and the array A. Run the code on the MARS simulator step by step and observe the change the memory and registers. Test the code for more cases.

Assignment 3

Create a new project to implement the code in Sample Code 3. Run the code on the MARS simulator step by step and observe the change in the memory and registers. Change the value of *test* to check if all the cases of the switch-case statement work correctly.

Assignment 4

Modify the Assignment 1 so that the condition of the branch is:

a. i <j
b. i >= j
c. i+j <= 0
d. i+j > m+n

Assignment 5

Modify the Assignment 2 so that the condition at the end of the loop is:

a. i < n
b. i <= n
c. sum >= 0
d. A[i] == 0

Assignment 6

Refering to all of the above instructions and statements, implement the following function in assembly:

- Assuming that we have an array, called A, of n integers ($n \leq 10$). Find an element that has the largest absolute value in array A.

Quizes

- Which registers are affected by a branch instruction?