CS2610: Computer Organization and Architecture

Lab 9: Report

Devansh Singh Rathore(111701011)

Objective

In this lab we designed a simple automatic temperature controller using MIPS assembly.

Problem

The temperature inside a laboratory should be maintained at 20 degree C. Design a temperature controller system that operates as below:

The system should poll for the outside temperature every 60 time units. The temperature thus obtained must be then adjusted to get the required room temperature (20 degree C).

After each poll, the difference between the temperature outside and the required room temperature must be stored in an array named ‘error_array’.

This process should continue until the outside temperature becomes equal to the required room temperature. It should be ensured that the temperature outside has become stable.

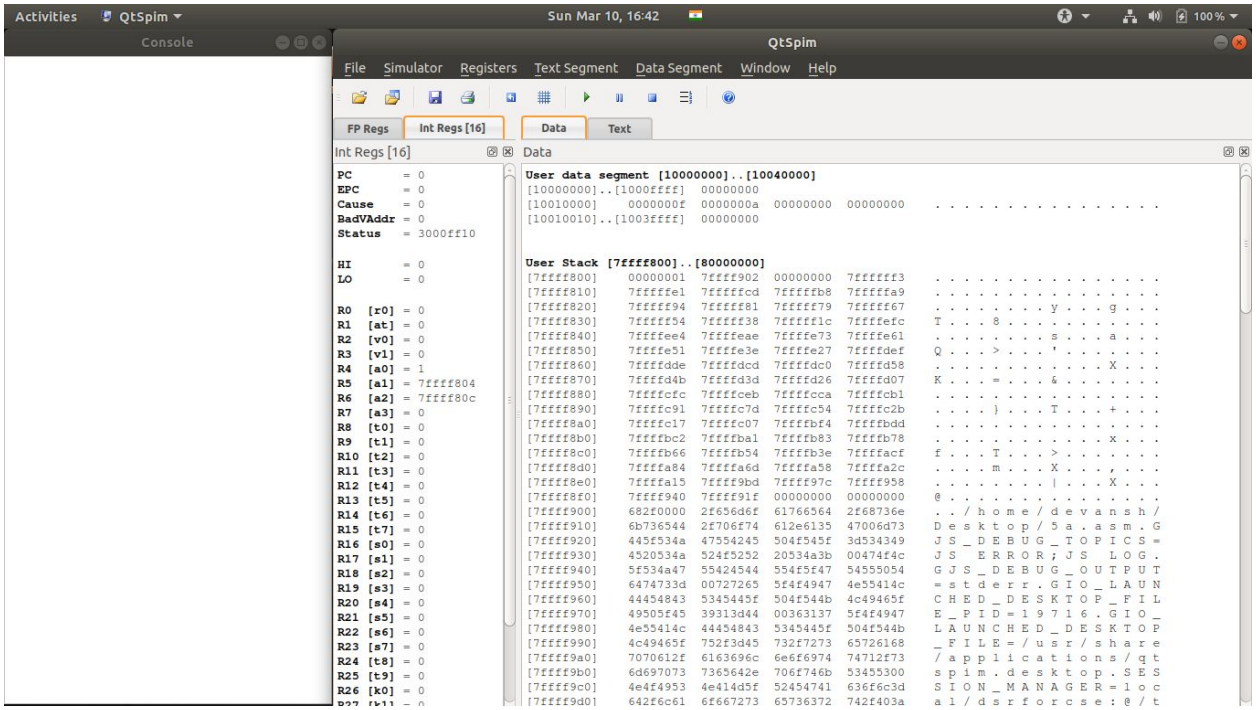
Once the temperature outside is stable, display the error_array.

Use loops, procedures, nested procedures for iterative and reusable operations respectively.

Provide comments and display statements wherever required.

Implementation

Initial Snapshot of Registers, Data Segments, and Console:



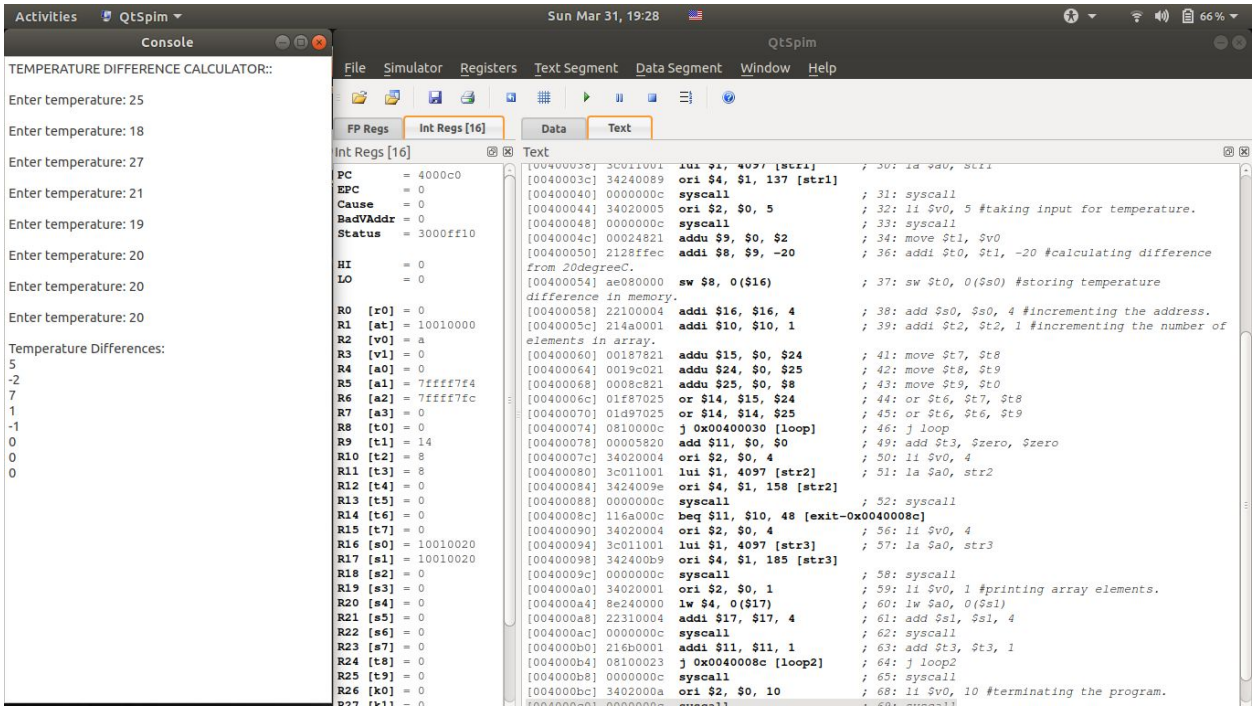
CODE:

```
.data
arr: .space 100                 #creating extra space for temp. differences
to be stored.
str0: .asciiz "TEMPERATURE DIFFERENCE CALCULATOR::\n"
str1: .asciiz "\nEnter temperature: "
str2: .asciiz "\nTemperature Differences: "
str3: .asciiz "\n"

.text
la $s0, arr                     #loading address of initial array.
move $s1, $s0                   #copying address to another location.
#Initializing temp. difference to a non-zero value so that it
#can enter the loop.
addi $t0, $zero, 1
#Initializing $t2 to zero which will store the number of elements #in the
array.
add $t2, $zero, $zero

li $v0, 4                       #printing Heading.
la $a0, str0
syscall
```

```
add $t7, $zero, 1            #Creating register to store last three
add $t8, $zero, 1            #temperature differences.
add $t9, $zero, 1
add $t6, $zero, 1            #For storing ‘OR’ of the three differences.

loop:
beq $t6, $zero, end           #Condition to exit loop, when last three
differences were 0.

li $v0, 4                     #printing string to request input.
la $a0, str1
syscall
li $v0, 5                     #taking input for temperature.
syscall
move $t1, $v0

addi $t0, $t1, -20            #calculating difference from 20 degree C.
sw $t0, 0($s0)                #storing temperature difference in memory.
add $s0, $s0, 4               #incrementing the address.
addi $t2, $t2, 1              #incrementing the number of elements in
array.

move $t7, $t8                 #shifting values so as to maintain last three
move $t8, $t9                 #temperature difference values.
move $t9, $t0
or $t6, $t7, $t8
or $t6, $t6, $t9             #calculating their bitwise or.
j loop

end:                          #end module to print the array
add $t3, $zero, $zero
li $v0, 4
la $a0, str2
syscall
loop2:                        #printing loop module
beq $t3, $t2, exit            #condition for terminating the printing loop.

li $v0, 4
la $a0, str3
syscall
li $v0, 1                     #printing array elements.
```

```
lw $a0, 0($s1)
add $s1, $s1, 4
syscall
add $t3, $t3, 1
j loop2
syscall

exit:                          #exit module
li $v0, 10                     #terminating the program.
syscall
```

CHANGES IN REGISTERS/DATA SEGMENTS(Final Snapshot after program execution):



EXPLANATION:

We can just maintain three registers storing the value of three previous temperature differences, and then ensure that there ‘OR’ is not equal to zero, otherwise exit the loop and print the temperature difference array stored in memory address on console.

‘OR’ value will be only zero if all three value are zero. This is to avoid case where sum is like eg. +1, -1, 0, which will add to zero, but not ‘OR’ will give zero.

Observations

Key observations in performing the above executions in mips:

1. For calling procedures and returning from procedures we need to use several jump operations jal, jr, j, etc.
2. To maintain array to store the values we can create a container of ‘.space’ type and then store value into corresponding addresses.
3. For taking integer input, we need to load immediate(5) into ‘v0’ register, for conducting the input operation.

Conclusions

We learnt a method for designing a simple automatic temperature controller using MIPS assembly, which involved concepts of procedure calling, arrays.