**LAB # 6**

**BIT MANIPULATION INSTRUCTIONS IN MIPS OBJECTIVE**

Learn to use MIPS bit manipulation instructions in assembly language programs.

**THEORY**

**BITWISE LOGICAL INSTRUCTIONS**

|  |  |  |  |
| --- | --- | --- | --- |
| Instructions | | Description | |
| and | rd, rs, rt | rd = rs & rt | |
| andi | rt, rs, immediate | rt | = rs & immediate |
| or | rd, rs, rt | rd = rs | rt | |
| ori | rt, rs, immediate | rd = rs | immediate | |
| nor | rd, rs, rt | rd = ! ( rs | rt ) | |
| xor | rd, rs, rt | To do a bitwise logical Exclusive OR. | |
| xori | rt, rs, immediate |  | |

The main usage of bitwise logical instructions are: ***to set, to clear, to invert***, and to ***isolate*** some selected bits in the destination operand. To do this, a source bit pattern known as a mask is constructed. The Mask bits are chosen based on the following properties of AND, OR, and XOR with Z represents a bit (either 0 or 1):

|  |  |  |
| --- | --- | --- |
| AND | OR | XOR |
| Z AND 0 = 0 | Z OR 0 = Z | Z XOR 0 = Z |
| Z AND 1 = Z | Z OR 1 = 1 | Z XOR 1 = ~Z |

**AND Instruction**

The AND instruction can be used to CLEAR specific destination bits while preserving the others. A zero mask bit clears the corresponding destination bit; a one mask bit preserves the corresponding destination bit.

**OR Instruction**

The OR instruction can be used to SET specific destination bits while preserving the others. A one mask bit sets the corresponding destination bit; a zero mask bit preserves the corresponding destination bit.

**XOR Instruction**

The XOR instruction can be used to INVERT specific destination bits while preserving the others. A one mask bit inverts the corresponding destination bit; a zero mask bit preserves the corresponding destination bit.

**Program#1:**

**Performing Bitwise AND Instruction with Mask 1**

**# Objective: Performs bitwise AND instruction with Mask 1. ################# Data segment #####################**

**.data input: .asciiz "\n enter an integer value: " # variable declaration**

**result: .asciiz "\n result is: "**

**################# Code segment #####################**

**.text**

**.globl main main:**

**li $t0,0xffffffff # 1 Mask**

**la $a0,input # Print input message li $v0,4 syscall**

**li $v0,5 # user input syscall**

**move $t1,$v0**

**and $t2,$t1,$t0 # AND instruction, $t2 = $t1 AND $t0 la $a0,result # Print result**

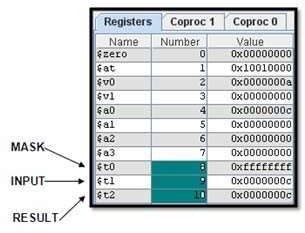
**message li $v0,4 syscall**

**move $a0,$t2 # Move AND instruction result in $a0 li $v0,1 # Print value of $t2**

**syscall**

**li $v0,10 # Exit program syscall**

**Output:**



12

12

**Program#2:**

**Performing Bitwise AND Instruction with Mask 0**

**# Objective: Performs bitwise AND instruction with Mask 0. ################# Data segment #####################**

**.data input: .asciiz "\n enter an integer value: " # variable declaration**

**result: .asciiz "\n result is: "**

**################# Code segment #####################**

**.text**

**.globl main**

**main:**

**li $t0,0x00000000 # 0 Mask**

**la $a0,input # Print input message li $v0,4**

**syscall**

**li $v0,5 # user input**

**syscall**

**move $t1,$v0**

**and $t2,$t1,$t0 # AND instruction, $t2 = $t1 AND $t0 la $a0,result # Print result message**

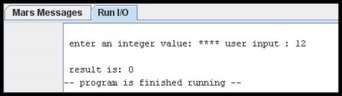
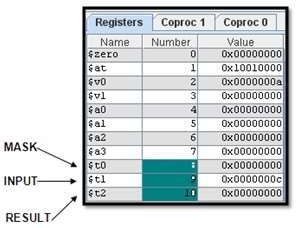
**li $v0,4 syscall**

**move $a0,$t2 # Move AND instruction result in $a0**

**li $v0,1 # Print value of $t2 syscall**

**li $v0,10 # Exit program syscall**

**Output:**



**Lab Task:**

**Complete the table by solving the bitwise instruction of all Logical gates. Add the code and output of the logical gates to show solution of MASK BITS given in the table.**

|  |  |  |
| --- | --- | --- |
| **Logic** | **Mask Bits** | |
|  | **0** | **1** |
| **AND** |  |  |
| **OR** |  |  |
| **NOT** |  |  |
| **XOR** |  |  |
| **XNOR** |  |  |
| **NOR** |  |  |
| **NAND** |  |  |