4 all purpose registers R0 (00), R1(01), R2(10), R3 (11)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction** | **Functionality** | **Opcode** | **Parity Bit** | **Example** | |
| load Rx, *imm* | $R1 = imm | 000 xx ii | 1 | Load R1, 3 | 1 000 01 11 |
| load Rx, Ry | Rx = Mem[Ry] | 000 xx yy | 0 | Load R1, (R2) | 0 000 01 10 |
| store *Rx, Ry* | Mem[Ry] = Rx | 011 xx yy | 1 | Store R0, (R1) | 1 011 00 01 |
| shl *Rx* | Rx shift left one bit, 0 shifted into LSB | 001 00 xx | 0 | Shl R0 | 0 001 00 00 |
| sll *Rx, Ry* | Rx = Rx \* (2^Ry) | 100 xx yy | 0 | sll R0, R1 | 0 100 00 01 |
| BezDec *imm* | If $R0 == 0, then PC = PC + imm, else $R0 = $R0 – 1, PC = PC + 1 | 010 01 ii | 0 | Bezdec 2 | 0 010 01 10 |
| xori *Rx, imm* | $R0 = Rx (EXCL) with imm | 110 xx ii | 0 | xori R1 2 | 0 110 01 10 |
| andi *Rx, imm* | $R0 = Rx (AND) with imm | 111 xx ii | 0 | Andi R1 3 | 1 111 01 11 |
| jump *‘branch’* | PC = PC + imm | 010 iiii | 1 | Jump 15 | 1 010 1111 |
| add Rx, Ry | Rx = Rx+Ry  #for addi first execute load instr, then add two registers | 001 xx yy | 1 | add R0, R1 | 1 001 00 01 |
| sub Rx, Ry | Rx = Rx – Ry  # also used to check if two branches are equal | 101 xx yy | 1 | sub R0, R1 | 1 101 00 01 |
| halt | Stop | 111 11 11 | 1 | stop | 1 111 11 11 |

**Machine Code for Program 1:**

#Assume everything is equal to zero at first

#$t1 = 00

#$t4 = 01

#$t5 = 10

#$t6 = 11

#$t7 = 5

#$t9 = 6

#$s0 = 7

addi $t6, $0, 1 1 001 11 01 #initialize register to equal to one

lw $t1, P($0) 1 100 11 00 #load variable value into register $t1

loop:

slt $t1, $0 1 101 00 00

beq $t1, $0, exit 1 100 01 ?? #end program

addi $t7, $0, -1 1 001 ?? 11 #-1 in 2s complement

addi $t5, $0, 5 1 001 10 ?? #5 value?

addi $t9, $0, 17 #add multiple times 1 001 ?? ?? #need 4 bits of 17

next:

beq $t5, $0, next2 1 100 01 ?? #jump to loop

add $t6, $t6, $t4 001 11 01 #store new value of $t6 as sum of $t6

and $t4

add $t5, $t5, $t7 001 10 ?? #store new value of $t5 as sum of $t5 and $t7

j next 1 010 ???? #jump to PC location of next

next2:

slt $s0, $t6, $t9 1 101 11 ?? # check if $t6 < $t9 and store result in $s0

bne $s0, $0, down 100 10 ?? #if $t6 < $t9, jump to down

subi $t6, $t6, 17 001 11 ?? #add $t6 to negative 2’s complement of 17 (so $t6 – 17)

j next2 010 ???? #jump to PC location of next2

down:

add $t5, $0, 5 001 10 ?? #set register $t5 to 5

add $t1, $t1, $t7 001 00 ?? #set $t1 = $t1 + $t7

add $t4, $0, $t6 001 01 11 #set register $t4 to $t6

j loop 010 ?? ?? #jump to PC location of loop

exit: sw $t6, R($0) 011 11 00 #store final result into R variable

000 00 00 #end program