CS224

SECTION 03

LAB04 PRELIMINARY REPORT

AYBÜKE CEREN DURAN/21302686

**b-)** I converted the code given into MIPS assembly via the link below(which contains converter):

https://www.eg.bucknell.edu/~csci320/mips\_web/

|  |  |  |
| --- | --- | --- |
| Instruction in Hex | MIPS code | Instruction Description |
| 8'h00: instr = 32'h20020005; | addi $v0, $zero, 5 | Add 5 to the register $v0 |
| 8'h04: instr = 32'h2003000c; | addi $v1, $zero, 0x000C | Put 0x000C in hex in register $v0 |
| 8'h08: instr = 32'h2067fff7; | addi $a3, $v1, 0xFFF7 | Add the immediate value 0xFF7 in hex and the value in $v1, put the result in register a3 |
| 8'h0c: instr = 32'h00e22025; | or $a0, $a3, $v0 | Or the values in a3 and v0 and put the result in a0 |
| 8'h10: instr = 32'h00642824; | and $a1, $v1, $a0 | And the values in v1 and a0 then put the result in a1 |
| 8'h14: instr = 32'h00a42820; | add $a1, $a1, $a0 | Add the value in a0 to a1 and put the result in a1 |
| 8'h18: instr = 32'h10a7000a; | beq $a1, $a3, 0x000A | If the values in a1 and a3 are equal, go to the location with 0x000A |
| 8'h1c: instr = 32'h0064202a; | slt $a0, $v1, $a0 | If v1 is less than a0, set a0 to 1 otherwise set it to 0 |
| 8'h20: instr = 32'h10800001; | beq $a0, $zero, 0x0001 | If v1 is not greater than a0 the go to 0x0001 |
| 8'h24: instr = 32'h20050000; | addi $a1, $zero, 0x0000 | Set a1 to 0 |
| 8'h28: instr = 32'h00e2202a; | slt $a0, $a3, $v0 | If a3 is less than v0 set a0 to 1 otherwise set a0 to 0 |
| 8'h2c: instr = 32'h00853820; | add $a3, $a0, $a1 | Add the values in a0 and a1 then put the result in a3 |
| 8'h30: instr = 32'h00e23822; | sub $a3, $a3, $v0 | Subtract v0 from a3 then put the result in a3 |
| 8'h34: instr = 32'hac670044; | sw $a3, 0x0044 $v1 | Store the value of a3 to 0x0044 offset of $v1 |
| 8'h38: instr = 32'h8c020050; | lw $v0, 0x0050 $zero | Load the value in 0x0050 offset of register 0 in $v0 |
| 8'h3c: instr = 32'h08000011; | j 0x0000011 | Jump to the memory location 0x0000011 |
| 8'h40: instr = 32'h20020001; | addi $v0, $zero, 0x0001 | Set v0 to 1 |
| 8'h44: instr = 32'hac020054; | sw $v0, 0x0054 $zero | Store the value of v0 to the 0x0054 offset of register zero |
| 8'h48: instr = 32'h08000012; | j 0x0000012 | Jump to the memory location 0x0000012 |

\\ you will add some instruction here

8’h4C: instr = 32….

you determine the upcode and modify cpu accordingly

**c-)** The instructions which I am assigned due to my section are : “jm” and “subi”.

RTL expressions of “jm” :

IM[PC]

PC ←mem[ R[rs] + ZeroExt(imm)]

RTL expressions of “subi”:

IM[PC]

RF[rt] ← RF[rs] – SignExt(immed)

PC←PC+4

**d-)** For “subi” , the datapath is changed as below:

A close up of a map

Description automatically generated

**e-)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruc-tion** | **Op5:0** | **Reg**  **Write** | **Reg**  **Dst** | **Alu**  **Src** | **Branch** | **Mem**  **Write** | **Mem**  **toReg** | **ALU**  **Op1:0** | **Jump** |
| R-type | 000000 | 1 | 1 | 0 | 0 | 0 | 0 | 1x | 0 |
| lw | 100011 | 1 | 0 | 1 | 0 | 0 | 1 | 00 | 0 |
| sw | 101011 | 0 | X | 1 | 0 | 1 | X | 00 | 0 |
| beq | 000100 | 0 | X | 0 | 1 | 0 | X | 01 | 0 |
| j | 000010 | 0 | X | X | X | 0 | X | XX | 1 |
| addi | 001000 | 1 | 0 | 1 | 0 | 0 | 0 | 00 | 0 |
| jm | 010101 | 0 | x | 1 | x | 0 | 0 | 00 | 1 |
| subi | 001110 | 1 | 0 | 1 | 0 | 0 | 0 | 01 | 0 |

**f-) jm:**

.text

# we know that s3 has an address inside

addi $s3, $zero, 0 # as default, put 0x0hex in s3

li $v0, 5

syscall

move $a0, $v0 # a0 has the value for the offset

sll $a0, $a0, 2

addi $t0, $zero, 0 # t0 the i counter

loop:

add $s3, $s3, 4

sub $a0, $a0, 4

bne $a0, $t0, loop

jr $s3

**subi test:**

subi $a7, $a6, 0x0011