

**CSE 332 Project**

Report Submission Date: 25th April, 2018

Section: 3

Name of group members:

|  |  |
| --- | --- |
| Name | ID |
| Mahbuba Tasmin | 1610064042 |
| Fatma Harun | 1610592042 |
| Umme Habiba Maliha | 1612150042 |
| Rubaith Adnan | 1510538642 |

Submitted to:

Dr.Tanzilur Rahman

## Objective

* To design a 8-bit single cycle CPU that has separate Data and Instruction memory.

## Number of operands

* 2 operands will be in action.
* The names will be as follows: rd(Register Destination) ,rs (Register Source)

## type of operands

* Register based

## number of operations & Reason

* 8 operations
* Since we have selected 3 bits for the op code that generates 23 = 8 number of operations

## number of formats

* 2 types of formats
* Name: R type, I type

## description of formats

**R-Type Format: -**

* Each field is 2 bits in length
* OP is an operation code or opcode that selects a specific operation
* rs is the source register
* rd is the destination register

For example: add $rd, $rs

|  |  |  |
| --- | --- | --- |
| OP  3bits | RD 2 bits | RS 2 bits |

**I-Type Format: -**

* Load work, store work, branch type, & immediate type are I-type
* For the lw,sw and bne instructions, one dedicated register will be used as $acc, which will serve the purpose of rs register in 32 bit MIPS style processor.
* RD is a a destination register for the other I-type instructions
* This format allows to jump instruction lines up to 2^3 = 8 lines and we will use this format to jump between instruction lines through “bne” instruction.

For Example: lw $rd, offset

|  |  |  |
| --- | --- | --- |
| OP  3 bits | RD 2 bits | Immediate  3 bits |

## list of registers

|  |  |  |
| --- | --- | --- |
| **REGISTER NAME** | **BINARY VALUE** | **USAGE** |
| $acc | 00 | Special registers ( for accumulator uses purpose ) |
| $s1 | 01 | Regular Registers |
| $s2 | 10 | Regular registers |
| $t1 | 11 | Temporary register |

## types of operation

* We will have 5 types of operations (namely: Arithmetic, Logical, Data transfer and Conditional branch)
* In addition, we have a special instruction named Disp , under I type format , that will be solely used to accommodate the 2 external display with the processor.
* We have total 8 different operation

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation Types** | **Instruction Name** | **Op code** | **Syntax** |
| Arithmetic | Add | 000 | Add $rd, $rs |
| sub | 001 | sub $rd, $rs |
| addi | 110 | Addi $rd, offset |
| Data Transfer | Lw | 100 | Lw $rd,offset |
| sw | 101 | Sw $rd,offset |
| Conditional Branch | bne | 011 | Bne $rd,offset |
| slt | 010 | Slt $rd,$rs |
| Special | disp | 111 | Disp $rd, 0 |

Instruction Description:

1.add: It adds two registers and stores the result in destination register.

Operation: $rd = $rs + $rd

Syntax: add $rd, $rs

2.sub: It subtracts two registers and stores the result in destination register.

Operation: $rd = $rd - $rs

Syntax: sub $rd, $rs

3.lw: It loads required value from the memory and write it back into the register.

Operation

$rd 🡨 Mem [[$ACC] + offset]

Syntax: lw $rd, offset

4. sw: It stores specific value from register to memory.

Operation:

$rd -- Mem [[$ACC] + offset]

Syntax: sw $rs, offset

5. bne: It checks whether the values of two register s are same or not. If it’s not same, it performs the operation located in the address at offset value.

Operation: if ($rd!=$ACC) jump to offset

else goto next line

Syntax: bne $rd, offset

7.SLt: If $rd is less than $rs, $rd is set to one. It gets zero otherwise.

Operation : if $rd < immediate $rd = 1;  else $rd = 0;

Syntax: slt $rd, immediate

8.Disp:

It displays the content of $rd in the two external 7 segment display connected to the processor .

Syntax : disp $rd, 000

## 7 segment display accomodation

As per the requirement of the processor to accommodate 2 external display with it, we have opted for a separate instruction “disp” in I type format.

Syntax: disp $rd , immediate [immediate assigned 0]

Operation:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | |  |  |  |  | | --- | --- | --- | --- | | Instruction memory(disp instruction ) | Register file (read the data of $rd) | Activates the control unit line of display | shows the data on 7 segment display | |  |

## Summary

* This is a 8-bit RISC Type CPU. As an ISA designer, we have chosen 2 type operands, 8 opcodes, 5 types of operations and 2 types of instruction format.
* This is a 8-bit CPU as a result there will be 4 registers in it. For the design, we followed “simplicity favors regularity”. So, we have assigned 3bits each to the opcodes for both formats. For $rd, $rs 2 bits is fixed for the R-Type register and for the I-type a 2 bit register is assigned with $rd along with immediate value of 3 bits.