Group 12

Name of Group Members

Koushik Chandra Sarker 2018000000024 Section: 3

Md. Tanvir Rahman Chowdury 2018000000028 Section: 2

Md Abir Hossain 2018000000055 Section: 2

Mia Md Mufrid 2018000000116 Section: 2

How many operands?

 Answer: 3 operands (e.g.: d = s + t)

Types of operand?

 Answer: Mixed

How many Operations?

 Answer: 8 Opcodes

Types of operations? (Arithmetic, logical, branch type?? How many from each category? List  the opcodes and respective binary values)

Answer: 5 Types

|  |  |  |
| --- | --- | --- |
| Operation Type | Operations | Register Type |
| Arithmetic | ADD | (R Type) |
| Logical | AND,OR | (R Type) |
| Data Transfer | ADDi, LW, SW | (I Type) |
| Conditional Branch | BEQ | (I Type) |
| Unconditional Jump | J | (J Type) |

No. of format of instruction (how many different formats?)

Answer: 3 Types of formats. R-Type, I-Type and J-Type

Describe each of the format (fields and field length)

Answer:

**R-Type Format: -**

o OP is 4 bit

o RD,RS,RT and Sa are 2 bits

o OP is an operation code or opcode that selects a specific operation

o RS and RT are the first and second source registers

o RD is the destination register

o Sa is the shift amount register

For example: add $s, $t, $d

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP  4 bits | RD  2 bits | RS  2 bits | RT  2 bits | Sa  2 bits |

**I-Type Format: -**

o Load work, store work, branch type, & immediate type are I-type

o RS is a source register, an address for loads and stores, or an operand for branch  and immediate arithmetic instructions

o RD is a source register for branches, but a destination register for the other I type instructions

For Example: addi $d, $s, offset

|  |  |  |  |
| --- | --- | --- | --- |
| OP  4 bits | RD  2 bits | RS  2 bits | Address/Immediate 4 bits |

**J-Type Format: -**

o Jump type is J-Type

o Target address

For Example: J EXIT

|  |  |
| --- | --- |
| OP  4 bits | Target  8 bits |

Format (fields and field length)

R-type

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operations | Opcode (OP)  4 bits | | | | Destination  (RD)  2 bits | | Source Reg  (RS)  2 bits | | Target  Reg (RT) 2 bits | | Shift  Amount(Sa)  2 bits | |
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADD | 0 | 0 | 0 | 0 | RD | | RS | | RT | | Sa | |
| SUB | 0 | 0 | 0 | 1 | RD | | RS | | RT | | Sa | |
| SLL | 0 | 0 | 1 | 0 | RD | | RS | | RT | | Sa | |
| SRL | 0 | 0 | 1 | 1 | RD | | RS | | RT | | Sa | |
| AND | 0 | 1 | 0 | 0 | RD | | RS | | RT | | Sa | |
| OR | 0 | 1 | 0 | 1 | RD | | RS | | RT | | Sa | |
| Xor | 0 | 1 | 1 | 0 | RD | | RS | | RT | | Sa | |

I-type

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Operations | Opcode (OP)   4 bits | | | | Source Reg (RS)   2 bits | Target Reg  (RT)   2 bits | Immediate   4 bits |
| LW | 0 | 1 | 1 | 1 | RS | RT | Immediate |
| SW | 1 | 0 | 0 | 0 | RS | RT | Immediate |
| ADDi | 1 | 0 | 0 | 1 | RS | RT | Immediate |
| BEQ | 1 | 0 | 1 | 0 | RS | RT | Immediate |
| BNE | 1 | 0 | 1 | 1 |  |  |  |

J-type

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operations | Opcode (OP)   4 bits | | | | Target   8 bits |
| Jump | 1 | 1 | 0 | 0 | target |

1. List of Registers?

Answer:

|  |  |  |
| --- | --- | --- |
| **Name of Registers** | **Register Number** | **Value Assigned (2 Bits)** |
| $t0 | 0 | 00 |
| $t1 | 1 | 01 |
| $t2 | 2 | 10 |
| $t3 | 3 | 11 |

Instruction Description

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | Operation | Syntax | Description |
| ADD | $s2(Destination)=$s1(Source)+$t1(Target) | Add $s2,$s1,$t1 | It adds two register and stores the result in destination register. It cannot handle negative value. |
| SUB | $s2(Destination)=$s1(Source)-$t1(Target) | Sub $s2,$s1,$t1 | It substruct two register’s data and stores the result in destination register. It cannot handle negative value. |
| AND | $s2(Destination)=$s1(Source) && $t1(Target) | And $s2,$s1,$t1 | It AND’s two register’s values and stores the result in destination register. Basically, it sets some bits to 0. |
| OR | $s2(Destination)=$s1(Source)||$t1(Target) | Or $s2,$s1,$t1 | It OR’s two register values and stores the result in destination register. Basically, it sets some bits to 1. |
| Sll | $s2(Destination)=$s1(Source)<<im.v | Sll $s2,$s1,2 |  |
| Srl | $s2(Destination)=$s1(Source)>>im.v | Srl $s2,$s1,2 |  |
| ADDi | $s2(Destination)=$s1(Source)+offset | Addi $s2,$s1,offset | It adds a value from register with an integer value and stores the result in destination register. |
| LW | $s2(Destination) = MEM[$s1(Source) + offset] | Lw $s2, offset($s1) | It loads required value from the memory and write it back into the register. |
| SW | MEM[$s1(Destination)+ offset] = $s2(Source) | Sw $s2, offset($s1) | It stores specific value from register to memory. |
| BEQ | if ($s1==$t1) jump to offset | Beq $s1, $t1, offset | It checks whether the values of two register s are same or not. If it’s same it performs the  operation located in the address at offset value. |
| J | Jump to offset | Jump 7 | Direct jump to target. |

**Summary:**

• This is a 12-bit RISC Type CPU. As an ISA designer, I have chosen 3 mixed type operands,  8 opcodes, 5 types of operations, 3 types of instruction format, and finally I have given  names and values of 4 different registers.

• This is a 12-bit CPU as a result there will be 12 registers in it. For the design, I believe

simplicity favors regularity. So, I have assigned 4bits to the opcodes and it is same for

every type. For R-type I have assigned 2 bits each for RD, RS, RT and Sa. For I-type I have assigned 2 bits each for RS, RT and 4 bits for immediate. For J-type 8 bits target.

• Smaller is faster. Though it is only a 12-bit CPU, it will handle very common operations. In this CPU, I tried to use as many as operations it can handle also keeping register’s values in my mind.