For our 8-bit datapath, we have an instruction set architecture with the following description:

* All instructions have fixed-length of 16-bits.
* We have three types of instructions: R-type, I-type, & J-type

R-type (Register instructions): **add, inv, and, or, clr**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **opcode** | **rs** | **rt** | **rd** |  | **padding** |
| 4-bits | 2-bits | 2-bits | 2-bits |  | 6-bits zero |

**opcode:** unique operation id

**rs and rt:** source registers

**rd:** destination register

**Examples:**

and $rd, $rs, $rt

and $1, $2, $3

* For inv instruction, the rs field would be left unused. While only the $rt field is as source register and $rd register as destination register (set the value of rs field to 0).
* For clr instruction, the rs field would be left unused. Only the rt field is used as the source register and rd field as the destination register address (set the value of rs field to 0, rt and rd fields should have the same value in this case).

**Examples:**

inv $rd, $rst

clr $rt

I-type (Immediate Instructions): **addi, andi, ori, sra, sll, lw, sw**

|  |  |  |  |
| --- | --- | --- | --- |
| **Opcode** | **rs** | **rd(rt)** | **immediate** |
| 4-bits | 2-bits | 2-bits | 8-bits constant |

**opcode:**  unique operation id

**rs:** source registers **rd:** destination register

**Examples:**

sra $rd, $rs, 1

sra $3, $1, 1

lw $rt, 10($rs)

lw $2, 10($3)

sw $rt, 67($rs)

sw $1, 67($0)

* For **sra (Shift Right Arithmetic)** and **sll (Shift Left Logical)** instructions, the **rd register** will be equal to **rs register** shifted to **right or left** by amount of immediate value (for our case it is always 1, shift by 1). For **sra** the sign of the number is also preserved (arithmetic shift to right).
* For **load (lw)** instruction, the **rs register** includes the **memory base address**, **rt register** is the **target register** andthe **immediate** filed contains the **address offset**.
* For **store (sw)** instruction, the **rs register** includes the **memory base address** and **rt register** is the **source register** andthe **immediate** filed contains the **address offset**.

J-type (Jump Instructions): **beq, bne**

|  |  |  |  |
| --- | --- | --- | --- |
| **opcode** | **rs** | **rt** | **immediate** |
| 4-bits | 2-bits | 2-bits | 8-bits constant |

**opcode:** unique operation id

**rs:**  source registers **rd:** destination register

**Examples:**

beq $rs, $rt, BranchLabel

beq $1, $3, BranchLabel

* For **beq** instruction, the **rs and rt registers** are the **source registers** or **the sides of comparison** and the **immediate** fieldcontainsthe **instruction offset which you denote as a textual label in your assembly instruction**.

Instruction Opcode to ALUOp bits translation table:

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | Instruction OpCode [3:0] | ALUOp [2:0] | Description |
| lw | **0000** | **000** | **Load Word Instruction (lw), ALU add operation is selected to add the offset address to the base address** |
| sw | **0001** | **000** | **Store Word Instruction (sw), ALU add peration is selected to add the offset address to the base address** |
| add | **0010** | **000** | **add instruction, ALU add operation is selected** |
| addi | **0011** | **000** | **add immediate instruction, ALU add operation is selected** |
| inv | **0100** | **001** | **inv instruction, ALU inv operation is selected** |
| and | **0101** | **010** | **and instruction, ALU and operation is selected** |
| andi | **0110** | **010** | **and immediate instruction, ALU and operation is selected** |
| or | **0111** | **011** | **or instruction, ALU or operation is selected** |
| ori | **1000** | **011** | **or immediate instruction, ALU and operation is selected** |
| sra | **1001** | **100** | **sra instruction, ALU arithmetic shift to right operation is selected** |
| sll | **1010** | **101** | **sll instruction, ALU logical shift to left operation is selected** |
| beq | **1011** | **110** | **beq instruction, ALU branch if equal operation is selected** |
| bne | **1100** | **111** | **bne instruction, ALU branch if not equal operation is selected** |
| clr | **1101** | **010** | **and instruction, ALU and operation is selected, first operand will be zero** |
| - | **1110** | **-** | **-** |
| - | **1111** | **-** | **-** |