

16 BIT PROCESSOR

[Document subtitle]



**Name: Hasib Mahmud Id: 1621065042**

**Name: Sadman Alam Id: 1610544042**

**Instruction Set Architecture**

**R-type:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **opcode** | **rs** | **rt** | **rd** | **shamt** |

**[15 – 12] [13 – 10] [09 – 06] [05 – 03] [02 - 00]**

**I-type:**

|  |  |  |  |
| --- | --- | --- | --- |
| **opcode** | **rs** | **rt** | **Immediate** |

**[15 – 12] [13 – 10] [09 – 06] [05 – 0]**

**Operations:**

|  |  |
| --- | --- |
| **R-type** | **I-type** |
| ADD | LW |
| SUB | SW |
| AND | BEQ |
| OR | BNE |
| NOR | ADDI |
| SLL | **ANDI** |
| SRL | **ORI** |
| SLT |  |

**Register List:**

|  |  |  |
| --- | --- | --- |
| **Serial No** | **Register’s Name** | **Binary Value** |
| 1. | $zero | 000 |
| 2. | $t0 | 001 |
| 3. | $t1 | 010 |
| 4. | $t2 | 011 |
| 5. | $t3 | 100 |
| 6. | $s1 | 101 |
| 7. | $s2 | 110 |
| 8. | $s3 | 111 |

**Compiler Input:**

add $t0, $zero, $zero //Sum = 0

addi $t1, $zero, 1 //Counter = 1

addi $t2, $zero, 10 //$t2 = 10

slt $t3, $t1, $t2 //loop: if $t1 < $t2, set $t3 = 1

beq $t3, $zero, 3 //if $t3 = 0, then end loop

add $t0, $t0, $t1 //$t0 = $t0 + $t1

addi $t1, $t1, 1 //$t1 += 1

beq $zero, $zero, -5 //to loop

sw $t0, 0($s1) // store $t0 to 0($s1)

**Compiler Output**:

0000 000 000 001 000

1001 000 010 000001

1001 000 011 001010

0111 010 011 100 000

1110 000 100 000011

0000 001 010 001 000

1001 010 010 000001

1110 000 000 111011

1011 001 110 000000

**Instruction Description:**

**R-type:**

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

·        Operation: $d = $s + $t

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

·        Operation: $d = $s - $t

And: It AND’s two register values and stores the result in destination register. Basically, it sets some bits to 0.

·        Operation:  $d = $s && $t

Or: It OR’s two register values and stores the result in destination register. Basically, it sets some bits to 1.

·        Operation:  $d=$s || $t

Nor: It NOR’s two register values and stores the result in destination register. Sometimes we use nor to get NOT of register value.

·        Operation:  $d=$s nor $t

Sll: It shifts bits to the left and fill the empty bits with zeros. The shift amount is depended on the offset value.

·        Operation: $d= $s << offset

Srl: It shifts bits to the right and fill the empty bits with zeros. The shift amount is depended on the offset value.

·        Operation: $d= $s >> offset

Slt: If $s is less than $t, $d is set to one. It gets zero otherwise.

·        Operation:   if $s < $t $d = 1

else $d = 0

**I-type:**

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

·        Operation: $d = MEM[$s + offset]

SW: It stores specific value from register to memory.

·        Operation: MEM[$d + offset] = $s

Beq: 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.

·        Operation: if ($s==$t) jump to offset

else goto next line

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 ($s!=$t) jump to offset

else goto next line

**Control Unit and ALU Control:**

|  |  |
| --- | --- |
| **Instruction** | **Opcode** |
| R-type | 0000 (shamt) |
| lw | 1010 |
| sw | 1011 |
| beq | 1110 |
| bne | 1111 |
| addi | 1001 |
| andi | 1100 |
| ori | 1101 |

|  |  |
| --- | --- |
| **ALU Control Line** | **Function** |
| 000 | And |
| 001 | Or |
| 010 | Nor |
| 011 | Add |
| 100 | Sub |
| 101 | Sll |
| 110 | Srl |
| 111 | Slt |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction Opcode** | **ALUOp** | **Instruction operation** | **OPcode Field** | **Desired ALU Action** | **ALU control input** |
| LW | 00 | Load Word | xxxx | Add | 000 |
| SW | 00 | Store Word | xxxx | Add | 000 |
| BEQ | 01 | Branch Equal | xxxx | Subtract | 001 |
| BNE | 01 | Branch Not Equal | xxxx | Subtract | 001 |
| Addi | 11 | Add Immediate | 1001 | Add | 000 |
| Andi | 11 | And Immediate | 1000 | And | 010 |
| Ori | 11 | Or Immediate | 1101 | Or | 011 |
| R-type | 10 | Add | 0000 | Add | 000 |
| R-type | 10 | Sub | 0001 | Sub | 001 |
| R-type | 10 | And | 0010 | And | 010 |
| R-type | 10 | Or | 0011 | Or | 011 |
| R-type | 10 | Nor | 0100 | Nor | 100 |
| R-type | 10 | Sll | 0101 | Sll | 101 |
| R-type | 10 | Srl | 0110 | Srl | 110 |
| R-type | 10 | Slt | 0111 | Slt | 111 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **RegDst** | **AluSrc** | **Mem2reg** | **RegWrt** | **MemRd** | **MemWrt** | **Beq** | **AluOp0** | **AluOp1** | **Bne** |
| R-format | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| Lw | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Sw | x | 1 | x | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| Beq | x | 0 | x | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| Bne | x | 0 | x | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Addi | x | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | x |
| Andi | x | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | x |
| Ori | X | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | x |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **AluOp** | | **Opcodes** | | | | **ALU** |
| **AluOp1** | **AluOp0** | **OP3** | **OP2** | **OP1** | **OP0** | **Operation** |
| 0 | 0 | X | X | X | X | 000 |
| X | 1 | X | X | X | X | 001 |
| 1 | 1 | 1 | 0 | 0 | 1 | 000 |
| 1 | 1 | 1 | 1 | 0 | 0 | 010 |
| 1 | 1 | 1 | 1 | 0 | 1 | 011 |
| 1 | X | 0 | 0 | 0 | 0 | 000 |
| 1 | X | 0 | 0 | 0 | 1 | 001 |
| 1 | X | 0 | 0 | 1 | 0 | 010 |
| 1 | X | 0 | 0 | 1 | 1 | 011 |
| 1 | X | 0 | 1 | 0 | 0 | 100 |
| 1 | X | 0 | 1 | 0 | 1 | 101 |
| 1 | X | 0 | 1 | 1 | 0 | 110 |
| 1 | X | 0 | 1 | 1 | 1 | 111 |

Limitations:

* Proper execution is not possible due to insufficient knowledge and planning.
* We could have followed MIPS processor structure very closely but had to do MIPS’s R Type function codes’ work with Opcodes.
* This processor and compiler are not optimized.

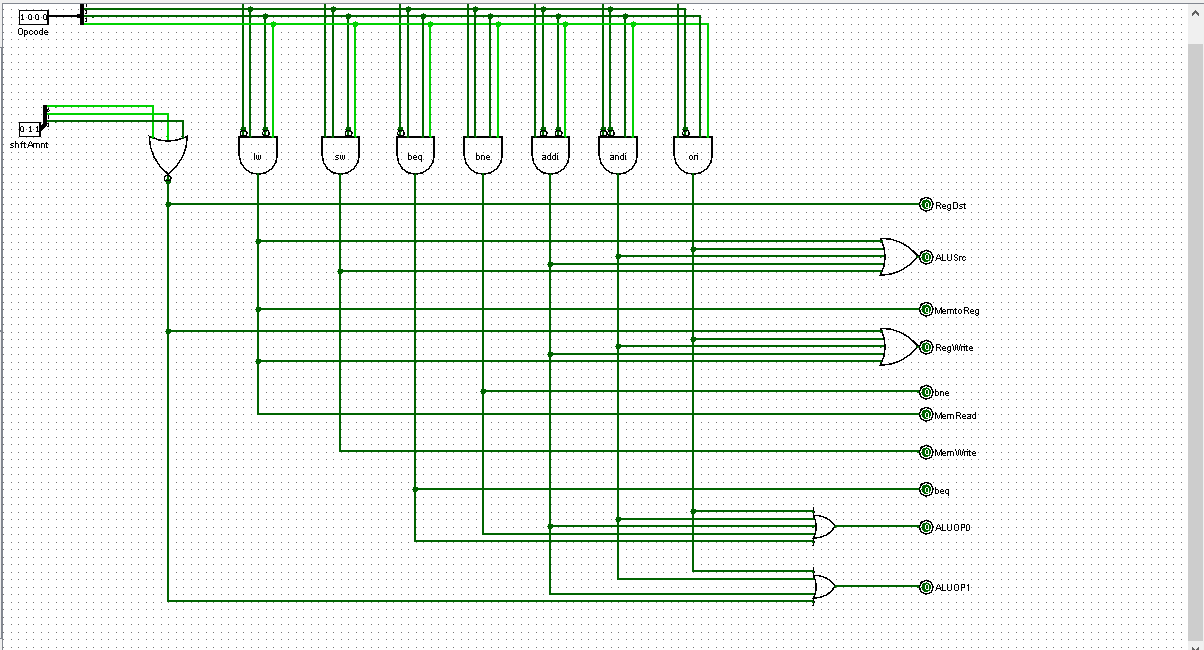
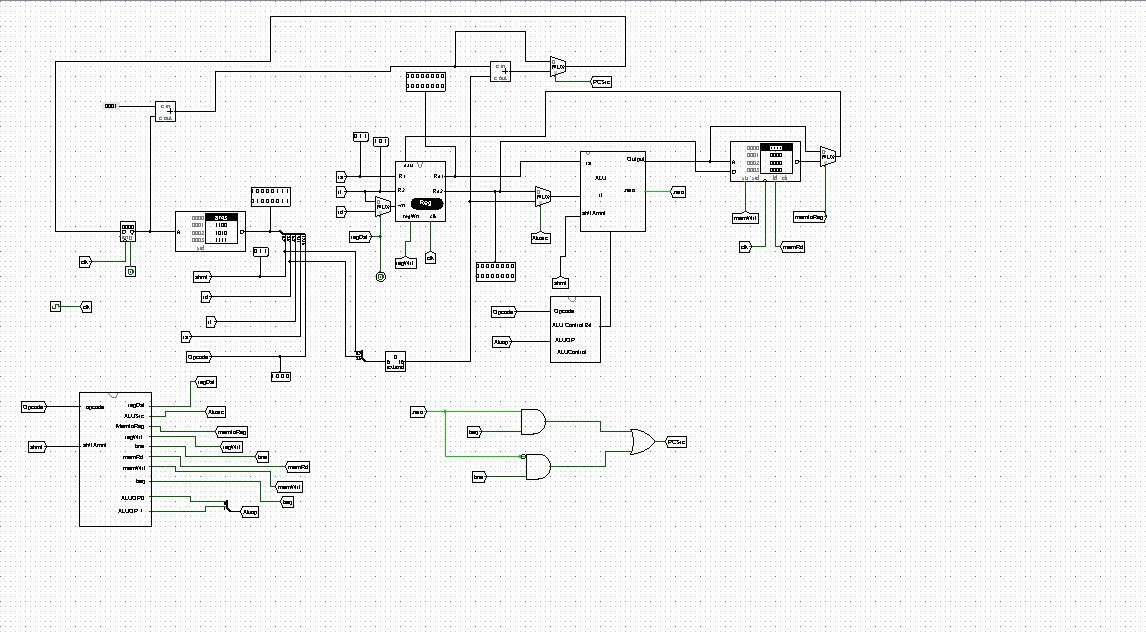


FIGURE: CONTROL UNIT

  
FIGURE: DATAPATH