

**Project description:**

Name : Khandaker Mobashyer Hossain

ID : 1611240042

Section : 02

Date of Submission : 29-04-2018

North South University

CSE 332 Project

We are to design a 20 bit single cycle CPU that can perform R-Type, I-Type and J-Type Instruction.

**ISA Design:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 bit | | 3 bit | | 3 bit | | 3 bit | | 3 bit | | 4 bit | |
| opcode | | rs | | rt | | rd | | shamt | | funct | |
| 19 | | 16 | 15 | 13 | 12 | 10 | 9 | 7 | 6 | 4 | 3 | 0 | |

**R-Type:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 4 bit | | 3 bit | | 3 bit | | 10 bit | |
| opcode | | rs | | rt | | immediate | |
| 19 | | 16 | 15 | 13 | 12 | 10 | 9 | 0 | |

**I-Type:**

|  |  |  |  |
| --- | --- | --- | --- |
| 4 bit | | 16 bit | |
| opcode | | target address | |
| 19 | | 16 | 15 | 0 | |

**J-Type:**

Here, rs = First Source Register

rt = Second Source Register

rd = Destination Register

funct = Function

shamt = Shift Distance

imm = immediate

**Instructions List:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **R-Type** | | | | | |
| **Instruction** | **Name** | **Action** | **opcode** | | **funct** |
| ADD rd, rs, rt | Addition | rd = rs + rt | 0000 | | 0000 |
| SUB rd, rs, rt | Subtraction | rd = rs – rt | 0000 | | 0001 |
| MULT rd, rs, rt | Multiplication | rd = rs \* rt | 0000 | | 0010 |
| DIV rd, rs, rt | Division | rd = rs / rt | 0000 | | 0011 |
| SLT rd, rt, rt | Set On Less Than | rd=rs < rt | 0000 | | 0100 |
| AND rd, rs, rt | AND | rd = rs & rt | 0000 | | 0101 |
| OR rd, rs, rt | OR | rd = rs | rt | 0000 | | 0110 |
| NOT rd, rt | NOT | rd = ~rt | 0000 | | 0111 |
| NOR rd, rs, rt | NOR | rd = ~(rs | rt) | 0000 | | 1000 |
| NAND rd, rs, rt | NAND | rd = ~(rs & rt) | 0000 | | 1001 |
| XOR rd, rs, rt | XOR | rd = rs ^ rt | 0000 | | 1010 |
| XNOR rd, rs, rt | XNOR | rd = ~(rs ^ rt) | 0000 | | 1011 |
| SLL rd, rt, shamt | Shift Left Logical | rd = rt << shamt | 0000 | | 1100 |
| SRL rd, rt, shamt | Shift Right Logical | rd = rt >> shamt | 0000 | | 1101 |
| JALR rd rs | Jump And Link Register | rd = pc; pc = rs | 0000 | | 1110 |
| JR rs | Jump Register | pc = rs | 0000 | | 1111 |
|  | | | | | |
| **I-Type** | | | | | |
| **Instruction** | **Name** | **Action** | | **opcode** | |
| SW rt, offset(rs) | Store Word | M[offset + rs] = rt | | 0001 | |
| LW rt, offset(rs) | Load Word | rt = M[offset + rs] | | 0010 | |
| BEQ rs, rt, offset | Branch On Equal | if(rs==rt) than pc = pc + offset | | 0011 | |
| BNE rs, rt, offset | Branch On Not Equal | if(rs!=rt) than pc = pc + offset | | 0100 | |
| ADDI rt, rs, imm | Add immediate | rt = rs + imm | | 0101 | |
| SUBI rt, rs, imm | Sub immediate | rt = rs – imm | | 0110 | |
| ANDI rt, rs, imm | AND immediate | rt = rs & imm | | 0111 | |
| ORI rt, rs, imm | OR immediate | rt = rs | imm | | 1000 | |
| NOTI rt, imm | NOT immediate | rt = ~ imm | | 1001 | |
| NORI rt, rs, imm | NOR immediate | rt = ~(rs | imm) | | 1010 | |
| NANDI rt, rs, imm | NAND immediate | rt = ~(rs & imm) | | 1011 | |
| XORI rt, rs, imm | XOR immediate | rt = rs ^ imm | | 1100 | |
| XNORI rt, rs, imm | XNOR immediate | rt = ~(rs ^ imm) | | 1101 | |
|  | | | | | |
| **J-Type** | | | | | |
| **Instruction** | **Name** | **Action** | | **opcode** | |
| J target | Jump | pc[0-15] = target;  pc[16-19] = (pc+1)[16-19]; | | 1110 | |
| JAL target | Jump And Link | r7 = pc+1; pc[0-15] = target;  pc[16-19] = (pc+1)[16-19]; | | 1111 | |

**Registers:**

|  |  |
| --- | --- |
| **Name** | **Address** |
| $r0 | 000 |
| $r1 | 001 |
| $r2 | 010 |
| $r3 | 011 |
| $r4 | 100 |
| $r5 | 101 |
| $r6 | 110 |
| $r7 | 111 |

**ALU Control Bits:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction**  **Opcode** | **ALUOp** | **Instruction**  **Operation** | **Function Field** | **Desired ALU Action** | **ALU Control Input** |
| SW | 0000 | Store Word | XXXX | ADD | 0000 |
| LW | 0000 | Lord Word | XXXX | ADD | 0000 |
| BEQ | 0001 | Branch on Equal | XXXX | SUB | 0001 |
| BNE | 0001 | Branch on Not Equal | XXXX | SUB | 0001 |
| ADDI | 0000 | ADD immediate | XXXX | ADD | 0000 |
| SUBI | 0001 | SUB immediate | XXXX | SUB | 0001 |
| ANDI | 0010 | AND immediate | XXXX | AND | 0101 |
| ORI | 0011 | OR immediate | XXXX | OR | 0110 |
| NOTI | 0100 | NOT immediate | XXXX | NOT | 0111 |
| NORI | 0101 | NOR immediate | XXXX | NOR | 1000 |
| NANDI | 0110 | NAND immediate | XXXX | NAND | 1001 |
| XORI | 0111 | XOR immediate | XXXX | XOR | 1010 |
| XNORI | 1000 | XNOR immediate | XXXX | XNOR | 1011 |
| R-Type | 1001 | ADD | 0000 | ADD | 0000 |
| R-Type | 1001 | SUB | 0001 | SUB | 0001 |
| R-Type | 1001 | MUL | 0010 | MUL | 0010 |
| R-Type | 1001 | DIV | 0011 | DIV | 0011 |
| R-Type | 1001 | Set On Less Than | 0100 | Set On Less Than | 0100 |
| R-Type | 1001 | AND | 0101 | AND | 0101 |
| R-Type | 1001 | OR | 0110 | OR | 0110 |
| R-Type | 1001 | NOT | 0111 | NOT | 0111 |
| R-Type | 1001 | NOR | 1000 | NOR | 1000 |
| R-Type | 1001 | NAND | 1001 | NAND | 1001 |
| R-Type | 1001 | XOR | 1010 | XOR | 1010 |
| R-Type | 1001 | XNOR | 1011 | XNOR | 1011 |
| R-Type | 1001 | Shift Left Logical | 1100 | Shift Left Logical | 1100 |
| R-Type | 1001 | Shift Right Logical | 1101 | Shift Right Logical | 1101 |

**ALU Control Truth Table:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ALUOp** | | | | **Funct Field** | | | | **Operation** |
| **ALUop3** | **ALUop2** | **ALUop1** | **ALUop0** | **F3** | **F2** | **F1** | **F0** |
| 0 | 0 | 0 | 0 | X | X | X | X | 0000 |
| 0 | 0 | 0 | 1 | X | X | X | X | 0001 |
| 0 | 0 | 1 | 0 | X | X | X | X | 0101 |
| 0 | 0 | 1 | 1 | X | X | X | X | 0110 |
| 0 | 1 | 0 | 0 | X | X | X | X | 0111 |
| 0 | 1 | 0 | 1 | X | X | X | X | 1000 |
| 0 | 1 | 1 | 0 | X | X | X | X | 1001 |
| 0 | 1 | 1 | 1 | X | X | X | X | 1010 |
| 1 | 0 | 0 | 0 | X | X | X | X | 1011 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0000 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0001 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0010 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0011 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0100 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0101 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0110 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0111 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1000 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1001 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1010 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1011 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1100 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1101 |

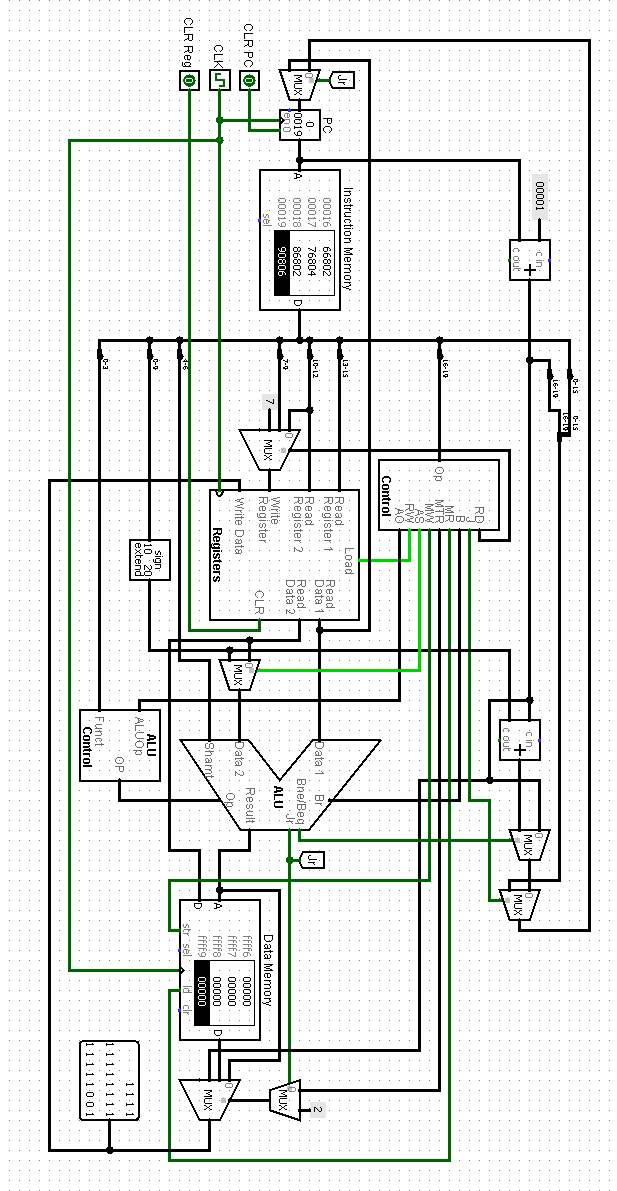
**Control Unit Signal Table:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Signals** | | | | | | | | |
| **Opcode** | **RegDst** | **Jump** | **Branch** | **MemRead** | **MemToReg** | **Memwrite** | **ALUSrc** | **RegWrite** | **ALUOp** |
| 0000 | 01 | 0 | 00 | 0 | 00 | 0 | 0 | 1 | 1001 |
| 0001 | X | 0 | 00 | 0 | X | 1 | 1 | 0 | 0000 |
| 0010 | 00 | 0 | 00 | 1 | 01 | 0 | 1 | 1 | 0000 |
| 0011 | X | 0 | 01 | 0 | X | 0 | 0 | 0 | 0001 |
| 0100 | X | 0 | 10 | 0 | X | 0 | 0 | 0 | 0001 |
| 0101 | 00 | 0 | 00 | 0 | 00 | 0 | 1 | 1 | 0000 |
| 0110 | 00 | 0 | 00 | 0 | 00 | 0 | 1 | 1 | 0001 |
| 0111 | 00 | 0 | 00 | 0 | 00 | 0 | 1 | 1 | 0010 |
| 1000 | 00 | 0 | 00 | 0 | 00 | 0 | 1 | 1 | 0011 |
| 1001 | 00 | 0 | 00 | 0 | 00 | 0 | 1 | 1 | 0100 |
| 1010 | 00 | 0 | 00 | 0 | 00 | 0 | 1 | 1 | 0101 |
| 1011 | 00 | 0 | 00 | 0 | 00 | 0 | 1 | 1 | 0110 |
| 1100 | 00 | 0 | 00 | 0 | 00 | 0 | 1 | 1 | 0111 |
| 1101 | 00 | 0 | 00 | 0 | 00 | 0 | 1 | 1 | 1000 |
| 1110 | X | 1 | 00 | 0 | 00 | 0 | X | 0 | XXXX |
| 1111 | 10 | 1 | 00 | X | 10 | 0 | X | 1 | XXXX |

**Example Instructions:**

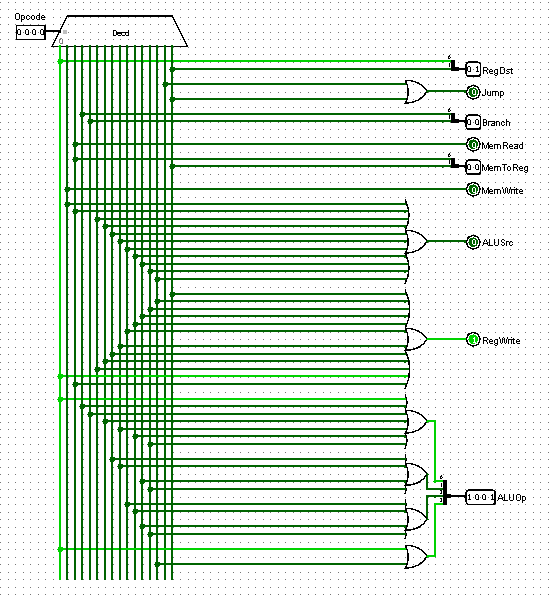
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Instruction** | **Action** | **Binary** | **Hex** |
| 01. | ADD $r1 $r2 $r3 | $r1 = $r2 + $r3 | 0000 010 011 001 000 0000 | 4C80 |
| 02. | SUB $r1 $r2 $r3 | $r1 = $r2 - $r3 | 0000 010 011 001 000 0001 | 4C81 |
| 03. | MULT $r1 $r2 $r3 | $r1 = $r2 \* $r3 | 0000 010 011 001 000 0010 | 4C82 |
| 04. | DIV $r1 $r2 $r3 | $r1 = $r2 / $r3 | 0000 010 011 001 000 0011 | 4C83 |
| 05. | SLT $r1 $r2 $r3 | $r1 = $r2 < $r3 | 0000 010 011 001 000 0100 | 4C84 |
| 06. | AND $r1 $r2 $r3 | $r1 = $r2 & $r3 | 0000 010 011 001 000 0101 | 4C85 |
| 07. | OR $r1 $r2 $r3 | $r1 = $r2 | $r3 | 0000 010 011 001 000 0110 | 4C86 |
| 08. | NOT $r1 $r3 | $r1 = ~ $r3 | 0000 010 011 001 000 0111 | 4C87 |
| 09. | NOR $r1 $r2 $r3 | $r1 = ~($r2 | $r3) | 0000 010 011 001 000 1000 | 4C88 |
| 0a. | NAND $r1 $r2 $r3 | $r1 = ~($r2 & $r3) | 0000 010 011 001 000 1001 | 4C89 |
| 0b. | XOR $r1 $r2 $r3 | $r1 = $r2 ^ $r3 | 0000 010 011 001 000 1010 | 4C8A |
| 0c. | XNOR $r1 $r2 $r3 | $r1 = ~($r2 ^ $r3) | 0000 010 011 001 000 1011 | 4C8B |
| 0d. | SLL $r4 $r5 3 | $r4 = $r5 << 3 | 0000 000 101 100 011 1100 | 163C |
| 0e. | SRL $r4 $r5 4 | $r4 = $r5 >> 4 | 0000 000 101 100 100 1101 | 164D |
| 0f. | JALR $r4 $r3 | $r4 = pc; pc = $r3 | 0000 011 000 100 000 1110 | 620E |
| 10. | JR $r5 | pc = $r5 | 0000 101 000 000 000 1111 | A00F |
| 11. | SW $r4 $r3 5 | M[5 + $r3] = $r4 | 0001 011 100 0000000101 | 17005 |
| 12. | LW $r2 $r3 4 | $r2 = M[4 + $r3] | 0010 011 010 0000000100 | 26804 |
| 13. | BEQ $r4 $r3 5 | if($r3 == $r4) than pc = pc + 5 | 0011 011 100 0000000101 | 37005 |
| 14. | BNE $r2 $r3 4 | if($r3 != $r2) than pc = pc + 4 | 0100 011 010 0000000100 | 46804 |
| 15. | ADDI $r2 $r3 3 | $r2 = $r3 + 3 | 0101 011 010 0000000011 | 56803 |
| 16. | SUBI $r2 $r3 2 | $r2 = $r3 – 2 | 0110 011 010 0000000010 | 66802 |
| 17. | ANDI $r2 $r3 4 | $r2 = $r3 & 4 | 0111 011 010 0000000100 | 76804 |
| 18. | ORI $r2 $r3 2 | $r2 = $r3 | 2 | 1000 011 010 0000000010 | 86802 |
| 19. | NOTI $r2 6 | $r2 = ~ 6 | 1001 000 010 0000000110 | 90806 |
| 1a. | NORI $r2 $r3 5 | $r2 = ~ ($r3 | 5) | 1010 011 010 0000000101 | A6805 |
| 1b. | NANDI $r2 $r3 3 | $r2 = ~ ($r3 & 3) | 1011 011 010 0000000011 | B6803 |
| 1c. | XORI $r2 $r3 4 | $r2 = $r3 ^ 4 | 1100 011 010 0000000100 | C6804 |
| 1d. | XNORI $r2 $r3 6 | $r2 = ~ ($r3 ^ 6) | 1101 011 010 0000000110 | D6806 |
| 1e. | J 12 | pc[0-15] = 12;  pc[16-19] = (pc+1)[16-19]; | 1110 0000000000001100 | E000C |
| 1f. | JAL 13 | $r7 = pc+1; pc[0-15] = 13;  pc[16-19] = (pc+1)[16-19]; | 1111 0000000000001101 | F000D |

**Circuit Diagram:**



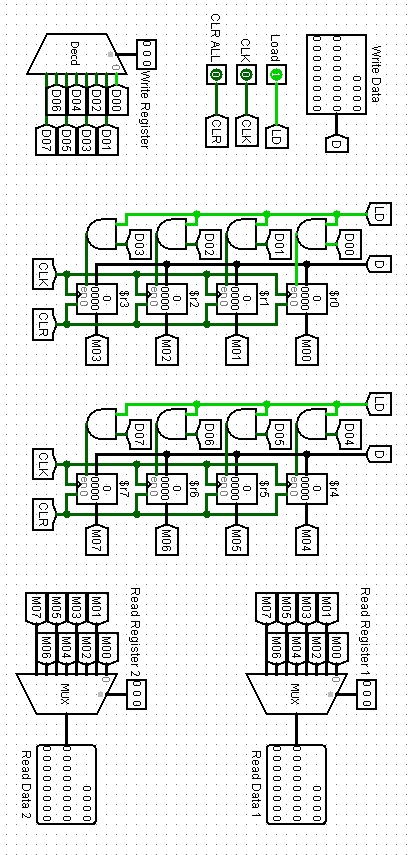
**Figure: 20-bit Single Cycle CPU**

**Circuit Diagram:**



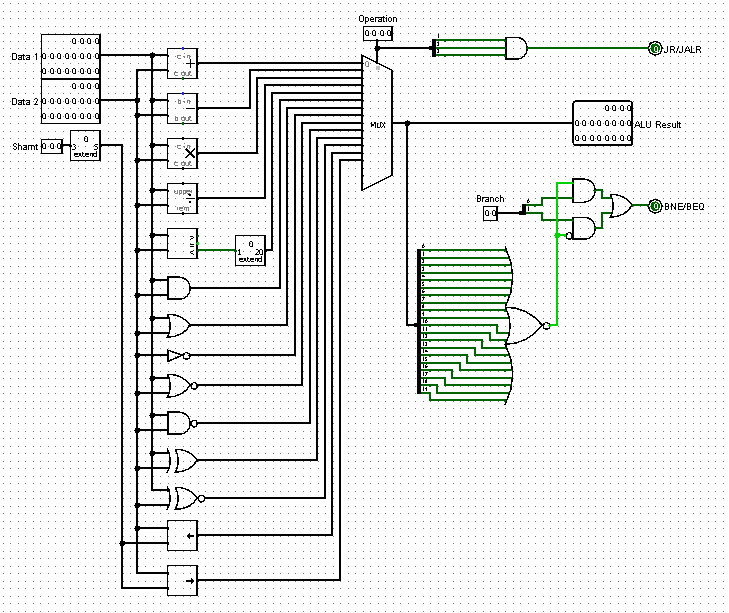
**Figure: Control Unit**

**Circuit Diagram:**



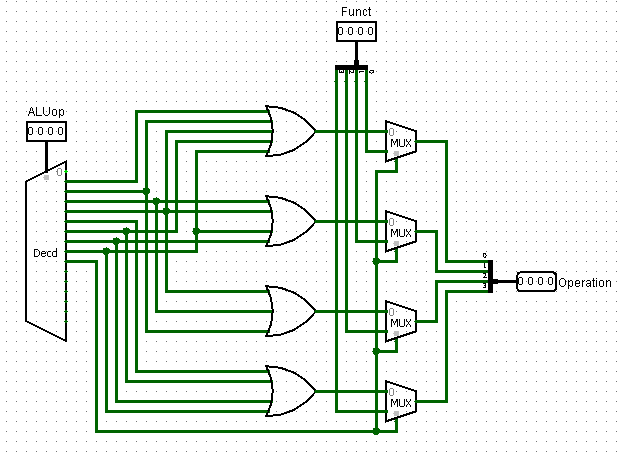
**Figure: Registers**

**Circuit Diagram:**



**Figure: ALU and Shift Unit**

**Circuit Diagram:**



**Figure: ALU Control Unit**