**1 Instruction Set Architecture**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Class** | **Instruction** | **Usage** | **Meaning** | **Opcode** | **Function Code** |
| Arithmetic | Add  Subtract | ADD rd, rs, rt  SUB rd, rs, rt | rd <- (rs) + (rt)  rd <- (rs) - (rt) | 000000  000000 | 000000  000001 |
| Logic | AND  OR  XOR | AND rd, rs, rt  OR rd, rs, rt  XOR rd, rs, rt | rd <- (rs) & (rt)  rd <- (rs) | (rt)  rd <- (rs) ⊕ (rt) | 000000  000000  000000 | 000100  000101  000110 |
| Shift and comp | SLA  SRA  SRL  NOT | SLA rs, rt  SRA rs, rt  SRL rs, rt  NOT rs, rt | rs <- rs << rt[0]  rs <- rs >>>rt[0]  rs <- rs <<<rt[0]  rs <- comp(rt) | 000000  000000  000000  000000 | 001000  001001  001010  000111 |
| Arithmetic Immediate | Add immediate  Sub immediate | ADDI rs, imm  SUBI rs, imm | rs <- (rs) + imm  rs <- (rs) - imm | 000001  000010 | X  X |
| Logical  Immediate | And immediate  OR immediate  XOR immediate | ANDI rs, imm  ORI rs, imm  XORI rs, imm | rs <- (rs) & imm  rs <- (rs) | imm  rs <- (rs) ⊕ imm | 000011  000100  000101 | X  X  X |
| Shift immediate | SLA immediate  SRA immediate  SRL immediate | SLAI rs, imm  SRAI rs, imm  SRLI rs, imm | rs <- (rs) << imm[0]  rs <- (rs) << imm[0]  rs <- (rs) << imm[0] | 000110  000111  001000 | X  X  X |
| Memory | Load Word  Store Word  Load Stack ptr  Store Stack ptr | LD rt, imm(rs)  ST rt, imm(rs)  LDSP rt, imm(rs)  STSP rt, imm(rs) | rt <- Mem[rs + imm]  Mem[rs + imm] <- rt  SP <- Mem[rs + imm]  Mem[rs + imm] <- rt | 001001  001010  001011  001100 | X  X  X  X |
| Branch | Unconditional B  Branch if < 0  Branch if > 0  Branch if == 0 | BR L  BMI rs, L  BPL rs, L  BZ rs, L | Goto L  If (rs < 0) Goto L  If (rs > 0) Goto L  If (rs == 0) Goto L | 001101  001110  001111  010000 | X  X  X  X |
| Move | Move val(R) -> val(R) | Move rt, rs | rt <- (rs) | 010001 | X |
| Stack Operation | Push  Pop  Call | PUSH rt  POP rs  CALL imm | Sp(rs) <- (rt)  rs <- sp(rt)  PC <= PC + imm  PC<= Mem[sp[ ; sp <= sp + 4 | 010010  010011 | X  X  X |

2 **Instruction Format and Encoding**

2.1 R-Format Instruction

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Opcode | rs | rt | rd | shamt | func |
| 6 bits | 5 bits | 5 bits | 5 bits | 6 bits | 5 bits |

|  |  |  |
| --- | --- | --- |
| **Instructions** | **Opcode** | **Function** |
| ADD  SUB | 000000  000000 | 000000  000001 |
| AND  OR  XOR | 000000  000000  000000 | 000100  000101  000110 |
| SLA  SRA  SRL  NOT | 000000  000000  000000  000000 | 001000  001001  001010  000111 |
|  |  |  |

2.2 I-Format Instructions

|  |  |  |  |
| --- | --- | --- | --- |
| **Opcode** | **rs** | **Don’t Care** | **Immediate** |
| 6 bits | 5 bits | 5 bits | 16 bits |

|  |  |
| --- | --- |
| **Instructions** | **Opcode** |
| ADDI  SUBI | 000001  000010 |
| ANDI  ORI  XORI | 000011  000100  000101 |
| SLAI  SRAI  SLAI | 000110  000111  001000 |

2.2 Memory Access Instructions

|  |  |  |  |
| --- | --- | --- | --- |
| **Opcode** | **rs** | **rt** | **Immediate** |
| 6 bits | 5 bits | 5 bits | 16 bits |

|  |  |
| --- | --- |
| **Instructions** | **Opcode** |
| LD | 001001 |
| ST | 001010 |
| LDSP | 001011 |
| STSP | 001110 |

2.3 Stack Instructions

|  |  |  |  |
| --- | --- | --- | --- |
| **Opcode** | **Don’t care** | **rt** | **Don’t care** |
| 6 bits | 5 bits | 5 bits | 16 bits |

|  |  |
| --- | --- |
| **Instruction** | **Opcode** |
| Push | 010010 |

|  |  |  |
| --- | --- | --- |
| **Opcode** | **rs** | **Don’t care** |
| 6 bits | 5 bits | 21 bits |

|  |  |
| --- | --- |
| **Instruction** | **Opcode** |
| Pop | 010011 |

|  |  |  |
| --- | --- | --- |
| **Opcode** | **Don’t care** | **Immediate** |
| 6 bits | **10** bits | 16 bits |

|  |  |
| --- | --- |
| **Instruction** | **Opcode** |
| Call | 010110 |

|  |  |
| --- | --- |
| **Opcode** | **Don’t care** |
| 6 bits | **26** bits |

|  |  |
| --- | --- |
| **Instruction** | **Opcode** |
| RET | 010111 |

2.4 J1-Format Instructions (Conditional Jump to Label Dependin on Register Value)

|  |  |  |  |
| --- | --- | --- | --- |
| **Opcode** | **rs** | **Don’t care** | **L** |
| 6 bits | 5 bits | 5 bits | 16 bits |

|  |  |
| --- | --- |
| **Instruction** | **opcode** |
| BMI | 001110 |
| BPL | 001111 |
| BZ | 010000 |

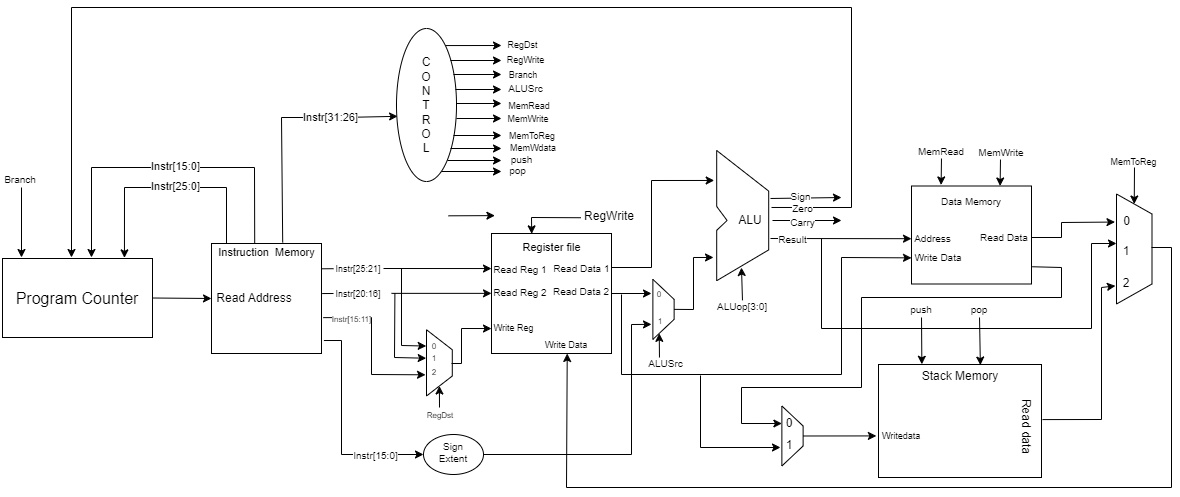
2.6 J3-Format Instructions (Jump to Label Unconditionally)

|  |  |
| --- | --- |
| **Opcode** | **L** |
| 6 bits | 26 bits |

|  |  |
| --- | --- |
| **Instruction** | **opcode** |
| BR | 001101 |

3 Register Usage Convention

|  |  |  |
| --- | --- | --- |
| **Register** | **Function** | **Register**  **Code** |
| $R0 | Zero Register, always stores the constant value 0 | 00000 |
| $R1-$R15 | General Purpose Register | 00001-01111 |
| $PC | Program Counter | 10000 |
| $SP | Stack Pointer | 11110 |



**4. Control Unit Design For various Instructions (Step by Step)**

1. **ADD (R-type Destination Reg(rd))**

|  |  |
| --- | --- |
| **Instruction Fetch** | IR 🡨 Mem[PC]  Next PC 🡨 PC + 4 |
| **Instruction Decode** | Read Data 1 🡨 Reg[rs]  Read Data 2 🡨 Reg[rt] |
| **Execution** | Result(ALU) 🡨 RD1 + RD2 |
| **Memory Access** | PC 🡨 Next PC |
| **Write Back** | Reg[rd] 🡨 Result(ALU) |

1. **NOT (Destination Reg(rs))**

|  |  |
| --- | --- |
| **Instruction Fetch** | IR 🡨 Mem[PC]  Next PC 🡨 PC + 4 |
| **Instruction Decode** | Read Data 1 🡨 Reg[rs]  Read Data 2 🡨 Reg[rt] |
| **Execution** | Result(ALU) 🡨 ~RD2 |
| **Memory Access** | PC 🡨 Next PC |
| **Write Back** | Reg[rs] 🡨 Result(ALU) |

**3. ADDI (Immediate type instructions Destination Reg(rs))**

|  |  |
| --- | --- |
| **Instruction Fetch** | IR 🡨 Mem[PC]  Next PC 🡨 PC + 4 |
| **Instruction Decode** | Read Data 1 🡨 Reg[rs]  Imm 🡨 (IR)16 ##IR15-0 |
| **Execution** | Result(ALU) 🡨 RD1 + Imm |
| **Memory Access** | PC 🡨 Next PC |
| **Write Back** | Reg[rs] 🡨 Result(ALU) |

**4. Load Word (Destination Reg(rt))**

|  |  |
| --- | --- |
| **Instruction Fetch** | IR 🡨 Mem[PC]  Next PC 🡨 PC + 4 |
| **Instruction Decode** | Read Data 1 🡨 Reg[rs]  Imm 🡨 (IR)16 ##IR15-0 |
| **Effective Address Calculation** | Result(ALU) 🡨 RD1 + Imm |
| **Memory Access** | PC 🡨 Next PC  Read data(Mem) 🡨 Mem[Result(ALU)] |
| **Write Back** | Reg[rt] 🡨 Read data(Mem) |

**5. Store Stack Pointer**

|  |  |
| --- | --- |
| **Instruction Fetch** | IR 🡨 Mem[PC]  Next PC 🡨 PC + 4 |
| **Instruction Decode** | Read Data 1 🡨 Reg[rs]  Imm 🡨 (IR)16 ##IR15-0 |
| **Effective Address Calculation** | Result(ALU) 🡨 RD1 + Imm |
| **Memory Access** | PC 🡨 Next PC  Mem[Result(ALU)] 🡨 SP(stack pointer) |
| **Write Back** | No operations |

**6. Unconditional Branch**

|  |  |
| --- | --- |
| **Instruction Fetch** | IR 🡨 Mem[PC]  Next PC 🡨 PC + 4 |
| **Instruction Decode** | Imm 🡨 (IR)26 ##IR25-0 |
| **Effective Address Calculation** | No operations |
| **Memory Access** | PC 🡨 26-bit Immediate address |
| **Write Back** | No operations |

**7. Conditional Branch**

|  |  |
| --- | --- |
| **Instruction Fetch** | IR 🡨 Mem[PC]  Next PC 🡨 PC + 4 |
| **Instruction Decode** | Read Data 1 🡨 Reg[rs]  Imm 🡨 (IR)16 ##IR15-0 |
| **Execution** | Sign <- ( Read\_Data 1 op 0 ) |
| **Memory Access** | If(sign) PC 🡨 Imm  Else PC 🡨 Next PC |
| **Write Back** | No operations |

**8. Push (Stack Operations )**

|  |  |
| --- | --- |
| **Instruction Fetch** | IR 🡨 Mem[PC]  Next PC 🡨 PC + 4 |
| **Instruction Decode** | Read Data 2 🡨 Reg[rt] |
| **Execution** | SP 🡨 SP + 4 |
| **Memory Access** | Stack\_Mem[SP] 🡨 Read Data 2 |
| **Write Back** | No operations |

**9. Call (Stack operations)**

|  |  |
| --- | --- |
| **Instruction Fetch** | IR 🡨 Mem[PC]  Next PC 🡨 PC + 4 |
| **Instruction Decode** | Imm 🡨 (IR)16 ##IR15-0 |
| **Execution** | SP 🡨 SP + 4 |
| **Memory Access** | Stack\_Mem[SP] 🡨 Next PC |
| **Write Back** | No operations |

**10. RET**

|  |  |
| --- | --- |
| **Instruction Fetch** | IR 🡨 Mem[PC]  Next PC 🡨 PC + 4 |
| **Instruction Decode** | No operations |
| **Execution** | Stack\_Address 🡨 SP  SP 🡨 SP - 4 |
| **Memory Access** | Next PC 🡨 Stack\_Mem[Stack\_add] |
| **Write Back** | No operations |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instr | Opcode | Func | RegDst | ALUSrc | MemToReg | RegWrite | MemWrite | MemRead | Branch | ALUOp | Push | Pop | MemWdata |
| ADD | 000000 | 000000 | 10 | 0 | 00 | 1 | 0 | 0 | 0 | 0010 | 0 | 0 | 0 |
| SUB | 000000 | 000001 | 10 | 0 | 00 | 1 | 0 | 0 | 0 | 0010 | 0 | 0 | 0 |
| AND | 000000 | 000100 | 10 | 0 | 00 | 1 | 0 | 0 | 0 | 0010 | 0 | 0 | 0 |
| OR | 000000 | 000101 | 10 | 0 | 00 | 1 | 0 | 0 | 0 | 0010 | 0 | 0 | 0 |
| XOR | 000000 | 000110 | 10 | 0 | 00 | 1 | 0 | 0 | 0 | 0010 | 0 | 0 | 0 |
| SLA | 000000 | 001000 | 10 | 0 | 00 | 1 | 0 | 0 | 0 | 0010 | 0 | 0 | 0 |
| SRA | 000000 | 001001 | 10 | 0 | 00 | 1 | 0 | 0 | 0 | 0010 | 0 | 0 | 0 |
| SRL | 000000 | 001010 | 10 | 0 | 00 | 1 | 0 | 0 | 0 | 0010 | 0 | 0 | 0 |
| NOT | 000000 | 000111 | 10 | 0 | 00 | 1 | 0 | 0 | 0 | 0010 | 0 | 0 | 0 |
| ADDI | 000001 | X | 00 | 1 | 00 | 1 | 0 | 0 | 0 | 0000 | 0 | 0 | 0 |
| SUBI | 000010 | X | 00 | 1 | 00 | 1 | 0 | 0 | 0 | 0001 | 0 | 0 | 0 |
| ANDI | 000011 | X | 00 | 1 | 00 | 1 | 0 | 0 | 0 | 0100 | 0 | 0 | 0 |
| ORI | 000100 | X | 00 | 1 | 00 | 1 | 0 | 0 | 0 | 0101 | 0 | 0 | 0 |
| XORI | 000101 | X | 00 | 1 | 00 | 1 | 0 | 0 | 0 | 0110 | 0 | 0 | 0 |
| SLAI | 000110 | X | 00 | 1 | 00 | 1 | 0 | 0 | 0 | 0111 | 0 | 0 | 0 |
| SRAI | 000111 | X | 00 | 1 | 00 | 1 | 0 | 0 | 0 | 1000 | 0 | 0 | 0 |
| SRLI | 001000 | X | 00 | 1 | 00 | 1 | 0 | 0 | 0 | 1001 | 0 | 0 | 0 |
| LD | 001001 | X | 01 | 1 | 01 | 1 | 0 | 1 | 0 | 1010 | 0 | 0 | 0 |
| ST | 001010 | X | 00 | 0 | 00 | 0 | 1 | 0 | 0 | 1010 | 0 | 0 | 0 |
| LDSP | 001011 | X | 00 | 1 | 01 | 0 | 0 | 1 | 0 | 1010 | 1 | 0 | 0 |
| STSP | 001100 | X | 00 | 1 | 00 | 0 | 1 | 0 | 0 | 1010 | 0 | 1 | 1 |
| BR | 001101 | X | 00 | 0 | 00 | 0 | 0 | 0 | 1 | 1011 | 0 | 0 | 0 |
| BMI | 001110 | X | 00 | 0 | 00 | 0 | 0 | 0 | 1 | 1011 | 0 | 0 | 0 |
| BPL | 001111 | X | 00 | 0 | 00 | 0 | 0 | 0 | 1 | 1011 | 0 | 0 | 0 |
| BZ | 010000 | X | 00 | 0 | 00 | 0 | 0 | 0 | 1 | 1011 | 0 | 0 | 0 |
| MOVE | 010001 | X | 01 | 0 | 00 | 1 | 0 | 0 | 0 | 1100 | 0 | 0 | 0 |
| PUSH | 010010 | X | 00 | 0 | 00 | 0 | 0 | 0 | 0 | 1100 | 1 | 0 | 0 |
| POP | 010011 | X | 00 | 0 | 10 | 1 | 0 | 0 | 0 | 0000 | 0 | 1 | 0 |
| CALL | 010100 | X | 00 | 0 | 00 | 0 | 0 | 0 | 0 | 1100 | 1 | 0 | 0 |
| RET | 010101 | X | 00 | 0 | 10 | 1 | 0 | 0 | 0 | 0000 | 0 | 1 | 0 |
| NOP | 010110 | X | 00 | 0 | 00 | 0 | 0 | 0 | 0 | 0000 | 0 | 0 | 0 |