Activity 1

R- type Instruction (Opcode – 0110011)



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instructiom | Funct7 | rs2 | rs1 | Funct3 | rd | Opcode |
| ADD | 0000000 | Src2 | Src1 | 000 | Dest | 0110011 |
| SUB | 0100000 | Src2 | Src1 | 000 | Dest | 0110011 |
| SLL | 0000000 | Src2 | Src1 | 001 | Dest | 0110011 |
| SLT | 0000000 | Src2 | Src1 | 010 | Dest | 0110011 |
| SLTU | 0000000 | Src2 | Src1 | 011 | Dest | 0110011 |
| XOR | 0000000 | Src2 | Src1 | 100 | Dest | 0110011 |
| SRL | 0000000 | Src2 | Src1 | 101 | Dest | 0110011 |
| SRA | 0100000 | Src2 | Src1 | 101 | Dest | 0110011 |
| OR | 0000000 | Src2 | Src1 | 110 | Dest | 0110011 |
| AND | 0000000 | Src2 | Src1 | 111 | Dest | 0110011 |
| MUL | 0000001 | Multiplier | Multiplicity | 000 | Dest | 0110011 |
| MULH | 0000001 | Multiplier | Multiplicity | 001 | Dest | 0110011 |
| MULHSU | 0000001 | Multiplier | Multiplicity | 010 | Dest | 0110011 |
| MULHU | 0000001 | Multiplier | Multiplicity | 011 | Dest | 0110011 |
| DIV | 0000001 | Multiplier | Multiplicity | 100 | Dest | 0110011 |
| DIVU | 0000001 | Multiplier | Multiplicity | 101 | Dest | 0110011 |
| REM | 0000001 | Multiplier | Multiplicity | 110 | Dest | 0110011 |
| REMU | 0000001 | Multiplier | Multiplicity | 111 | Dest | 0110011 |

I-type Instruction



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instruction | Imm[11:0]  (Offset) | rs1  (Base) | Funct3  (Width) | rd  (Dest) | Opcode  (LOAD) |
|  |
| LB | Offset | Base | 000 | Dest | 0000011 |
| LH | Offset | Base | 001 | Dest | 0000011 |
| LW | Offset | Base | 010 | Dest | 0000011 |
| LBU | Offset | Base | 100 | Dest | 0000011 |
| LHU | Offset | Base | 101 | Dest | 0000011 |
| JALR | Offset | Base | 000 | Dest | 1100111 |
| ADDI | I- immediate | Src | 000 | Dest | 0010011 |
| SLTI | I- immediate | Src | 010 | Dest | 0010011 |
| SLTIU | I- immediate | Src | 011 | Dest | 0010011 |
| XORI | I- immediate | Src | 100 | Dest | 0010011 |
| ORI | I- immediate | Src | 110 | Dest | 0010011 |
| ANDI | I- immediate | Src | 111 | Dest | 0010011 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instruction | Imm[11:5] | Imm[4:0] | rs1 | Funct3 | rd | Opcode |
| SLLI | 0000000 | Shamt | Src | 001 | Dest | 0010011 |
| SRLI | 0000000 | Shamt | Src | 101 | Dest | 0010011 |
| SRAI | 0100000 | Shamt | Src | 101 | Dest | 0010011 |

S-type Instruction



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instruction | Imm[11:0] | rs1 | rs2 | Funct3 | Imm[4:0] | Opcode |
| SB | Offset[11:0] | Src | Base | 000 | Offset[4:0] | 0100011 |
| SH | Offset[11:0] | Src | Base | 001 | Offset[4:0] | 0100011 |
| SW | Offset[11:0] | Src | Base | 010 | Offset[4:0] | 0100011 |

U-type Instruction



|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | Imm[13:12] | rd | Opcode |
| AUIPC | U-immediate | Dest | 0010111 |
| LUI | U-immediate | Dest | 0110111 |

B-type Instruction



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Instruction | Imm[12, 10:5] | rs1 | rs2 | Funct3 | Imm[4:1,11] | Opcode |
| BEQ | Offset[12,10:5] | Src2 | Src1 | 000 | Offset[4:1,11] | 1100011 |
| BNE | Offset[12,10:5] | Src2 | Src1 | 001 | Offset[4:1,11] | 1100011 |
| BLT | Offset[12,10:5] | Src2 | Src1 | 100 | Offset[4:1,11] | 1100011 |
| BGE | Offset[12,10:5] | Src2 | Src1 | 101 | Offset[4:1,11] | 1100011 |
| BLTU | Offset[12,10:5] | Src2 | Src1 | 110 | Offset[4:1,11] | 1100011 |
| BGEU | Offset[12,10:5] | Src2 | Src1 | 111 | Offset[4:1,11] | 1100011 |

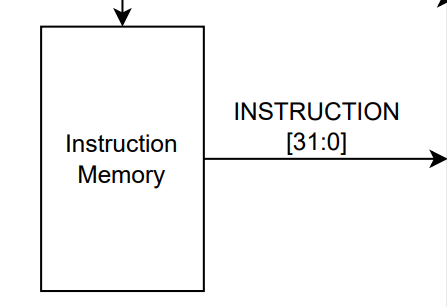
J-type Instruction



|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | Description | Opcode | Frnct3 |
| JAL | Jump and link | 1101111 |  |

**Main Units**

1. Instruction Memory

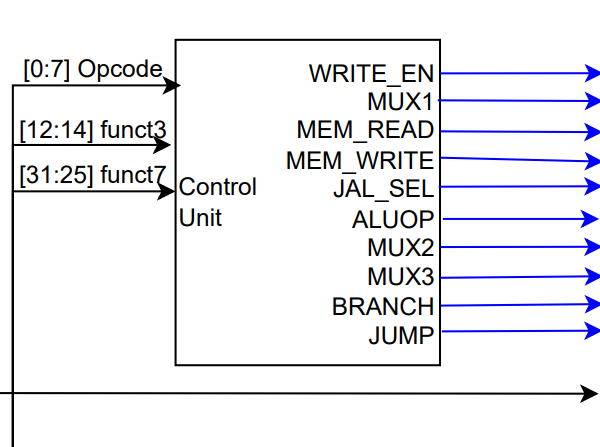


Instruction memory stores all the instructions (or programs) that the CPU needs to perform tasks. When the processor is ready to execute a task, it fetches the relevant instruction from this memory to know what to do next.

Input- PC

Output- 32-bit instruction

1. Control Unit



It fetches instructions from memory, decodes them to understand what needs to be done, and then sends out signals to the relevant parts of the CPU to execute those instructions.

Inputs

**1. opcode** - An opcode, short for "operation code," is a binary code that specifies the operation to be performed by a processor or microcontroller in order to execute a particular instruction.

**2. func3 -** func3 (short for function 3) is a field in the instruction format that specifies a subset of operations that can be performed by an instruction.

**3. func7 -** func7 (short for function 7) is a field in the instruction format that provides additional information about the operation to be performed by an instruction

Outputs(Control signals)

**1. WRITE\_ENABLE** - 1 bit signal. This is the control signal that determines whether data can be written into the register. When WRITE\_ENABLE is high it enables the write operation to the register file.

**2. MUX1\_SELECT** - 1 bit signal. This is the select signal for MUX1 which is used to select between ALU\_OUT and data memory output (DATA\_OUT). Then the selected data is directed to the register file.

**3. MEM\_READ** - 1 bit signal. This is the control signal that enables the output data from the memory unit.When the MEM\_READ signal is high, the memory unit is instructed to output data (DATA\_OUT) from the specified address, which is provided on the address lines (MEM\_ADDRESS).

**4. MEM\_WRITE** - 1 bit signal. This is the control signal that enables the input data to be written into the memory unit. When the MEM\_WRITE signal is high, the memory unit is instructed to write the data provided on the data lines (DATA\_IN) to the specified address on the address lines (MEM\_ADDRESS).

**5. JAL\_SELECT** - 1 bit signal. This is the select signal for the JAL multiplexer which is used to select between ALU\_RESULT and current PC+4 value.

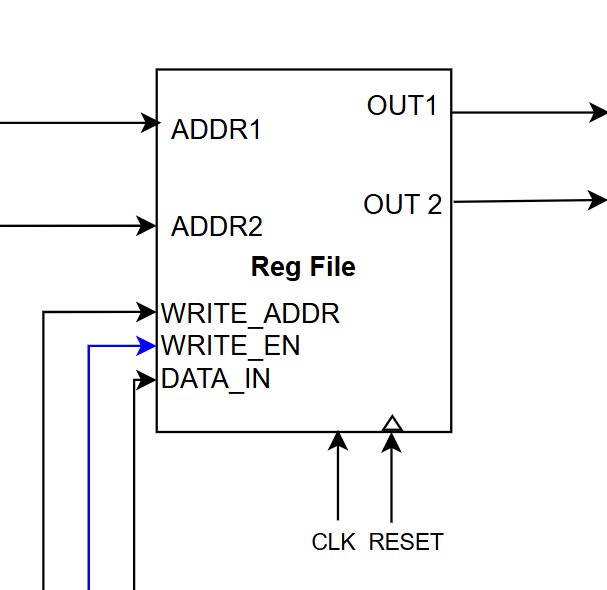
**6. ALUOP** - 5-bit signal for ALU. This signal specifies the arithmetic or logical operation to be performed by the ALU. It selects one of several operations that the ALU can perform, such as addition, subtraction, multiplication, bitwise AND, OR, XOR, and shift operations.

**7. MUX2\_SELECT** - 1 bit signal. This is the select signal for MUX2, which is used to select between register file output (DATA1) and PC value. Then the selected data is directed to the ALU for its operations.

**8. MUX3\_SELECT** - 1 bit signal. This is the select signal for MUX3, which is used to select between immediate value and register file output (DATA2). Then the selected data is directed to the ALU for its operations.

**9. BRANCH and JUMP** - 1-bit signals. These signals are used in the BRANCH JUMP CONTROLLER in order to distinguish whether the instruction is a branch or a jump. The BRANCH signal is used to conditionally execute a set of instructions based on a particular condition. The JUMP signal is used to unconditionally jump to a specific instruction address.

1. Register File



Inputs

**1. CLK** - In a register file, the clock signal is used to trigger the read or write operations to the registers.

**2. R** - The R signal is a read control signal that is used to enable a read operation from the register file.

**3. ADDR1 and ADDR2** - These are address signals that are used to specify the register addresses for a read or write operation.

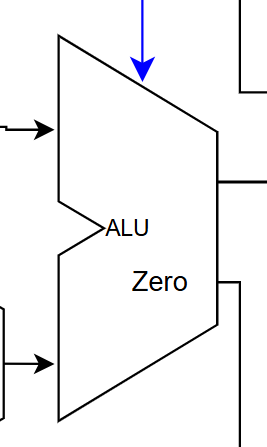
**4. WRITE\_ENABLE** - The write enable signal is used to enable a write operation to the register file.

**5. REG\_WRITE\_DATA** - The register write data signal is used to specify the data value to be written to a register in the register file during a write operation.

**6. WRITE\_REG\_ADDR** - The write register address signal is used to specify the address of the register to be written during a write operation.

Outputs

1. **DATA1 and DATA2** - DATA1 and DATA2 are typically used in the context of an arithmetic or logic operation in a computer system. They represent the two data operands that are being used in the operation.
2. ALU



Inputs

1. **MUX2 output** - The output of multiplexer 2 is one of the inputs to the ALU.

2. **MUX3 output** - The output of multiplexer 3 is one of the inputs to the ALU.

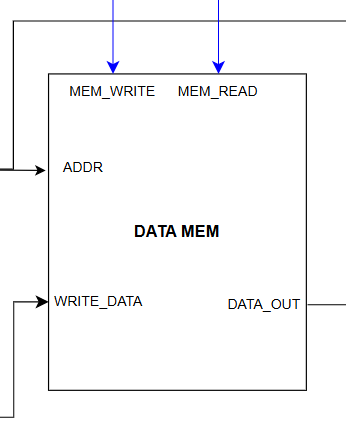
3. **ALUOP** - The ALU operation code is a control signal that specifies the operation to be performed by the ALU

Outputs

1. **ALU\_RESULT** - The ALU result is the output of the ALU operation.

2. **ZERO** - The ZERO flag is an output signal that indicates whether the result of the ALU operation is zero or not.

05. Data Memory



Inputs

**1. CLK** - The clock signal is an input that synchronizes the operation of the memory with other components in the system.

**2. MEM\_READ** - The MEM\_READ signal is an input that specifies whether the memory should read data from the specified address.

**3. MEM\_WRITE** - The MEM\_WRITE signal is an input that specifies whether the memory should write data to the specified address.

**4. MEM\_ADDRESS** - The MEM\_ADDRESS signal is an input that specifies the memory address to read from or write to.

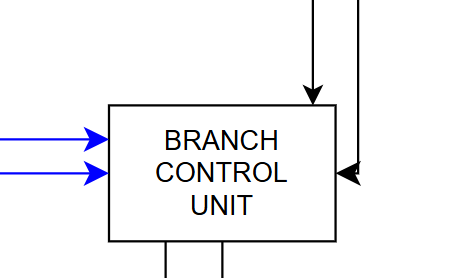
**5. funct3** - funct3 is an input that is used to decode the instruction and determine the operation to be performed on the memory.

**6. DATA\_IN** - The DATA\_IN signal is an input that provides data to be written to the specified memory address.

Outputs

**1. DATA\_OUT** - The DATA\_OUT signal is an output that provides the data read from the specified memory address.

1. Jump – Branch Controller



Inputs

**1. BRANCH** - The BRANCH input is a signal that indicates whether a branch instruction has been executed.

**2. JUMP** - The JUMP input is a signal that indicates whether a jump instruction has been executed.

**3. ZERO** - The ZERO input is a signal that indicates whether the ALU has produced a zero output.

**4. funct3** - Used to distinguish the condition of the branch instruction (Ex: Branch Less Than,Branch Greater Than or Equal, Branch Less Than Unsigned, etc)

**5. Jump/Branch** **Address** - This contains the target address of the Branch/ Jump instruction that has been executed.

Outputs

**1. Branch/ Jump signal** - This is used in PC MUX as the select bit to choose the next PC value

**2. Branch/ Jump PC address** - This contains the target address of the Branch/ Jump instruction that has been executed.

# Pipeline diagram with data path and control path

