Minimal Instruction Set Processor for Basic Arithmetic and Logic

Operations

Team: CompArch Kids

Team members

* Sameeth (23114017)
* Sonika (23114019)
* Raviteja (23114040)
* Koushik (23114063)
* Aasish Varma (23114069)

**Defining Instruction Set Architecture**

**Definition:** The semantics of all the instructions supported by a processor is known as the instruction set architecture (ISA). This includes the semantics of the instructions themselves, along with their operands, and interfaces with peripheral devices.

**Designing an ISA for Simple Processor:**

Each instruction is 32 bits wide and has the following general structure:

**Instruction Encoding**

3 Address Format:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Opcode | IMM Bit | Dest.reg | Reg1 | Reg2 | - |
| 31 – 29 | 28 | 27 – 23 | 22 – 18 | 17 – 13 | 12 – 0 |

Example:

1. Add r1, r2, r3
2. Sub r1, r2, r3
3. And r1, r2, r3
4. Or r1, r2, r3
5. Mul r1, r2, r3

2 Address Format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Opcode | IMM Bit | Dest.Reg | Reg1 | IMM |
| 31 - 29 | 28 | 27 – 23 | 22 – 18 | 17 - 0 |

Example:

1. Add r1, r2, #Immediate
2. Sub r1, r2, #Immediate
3. And r1, r2, #Immediate
4. Or r1, r2, #Immediate

1 Address Format:

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | IMM Bit | Dest.Reg | IMM |
| 31-29 | 28 | 27-23 | 22 - 0 |

Example:

1. MOV r1, #Immediate

----------------------------------------------------------------------------------------------

* **ALUOp (3 bits)**: This is the opcode for our ISA, it tells the ALU which operation to perform.
* **IMM (1 bit)**: Indicates whether the instruction uses an immediate value or not.
* **Write Reg (5 bits)**: The destination register where the result gets stored after execution of instruction.
* **Read Reg1 (5 bits)**: The first operand register.
* **Read Reg2 (5 bits)**: The second operand register or base address register.
* **Immediate (18 bits)**: Used as an immediate value for arithmetic operations.

----------------------------------------------------------------------------------------------

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ALUOp** | IMM | Instruction | Operation | Example Syntax |
| 000 | 0 | ADD | Write Reg = Read Reg1 + Read Reg2 | ADD R1, R2, R3 |
| 000 | 1 | ADD  (Immediate) | Write Reg = Read  Reg1 + Immediate | ADD R1, R2, #7 |
| 001 | 0 | SUB | Write Reg = Read Reg1 - Read Reg2 | SUB R4, R2, R3 |
| 001 | 1 | SUB  (Immediate) | Write Reg = Read  Reg1 - Immediate | SUB R4, R2, #2 |
| 010 | 0 | AND | Write Reg = Read Reg1 & Read Reg2 | AND R5, R2, R3 |
| 010 | 1 | AND | Write Reg = Read  Reg1 & Immediate | AND R5, R2, #3 |
| 011 | 0 | OR | Write Reg = Read Reg1 | Read Reg2 | OR R1, R2, R3 |
| 011 | 1 | OR | Write Reg = Read  Reg1 | Immediate | OR R1, R2, #5 |
| 100 | 0 | MUL | Write Reg = Read Reg1 \* Read Reg2 | MUL R7, R2, R3 |
| 101 | 1 | MOV  (Immediate) | Write Reg = Immediate | MOV R8, #34 |

**Instruction Descriptions**

1. ADD (ALUOp = 000, IMM = 0):

Adds the contents of Read Reg1 and Read Reg2, and stores the result in Write Reg.

1. ADD (Immediate) (ALUOp = 000, IMM = 1):

Adds the contents of Read Reg1 and an immediate value, and stores the result in Write Reg.

1. SUB (ALUOp = 001, IMM = 0):

Subtracts the contents of Read Reg2 from Read Reg1, and stores the result in Write Reg.

1. SUB (Immediate) (ALUOp = 001, IMM = 1):

Subtracts an immediate value from Read Reg1, and stores the result in Write Reg.

1. AND (ALUOp = 010, IMM = 0):

Performs a bitwise AND between Read Reg1 and Read Reg2, and stores the result in Write Reg.

1. OR (ALUOp = 011, IMM = 0):

Performs a bitwise OR between Read Reg1 and Read Reg2, and stores the result in Write Reg.

1. MUL (ALUOp = 100, IMM = 0):

Multiplies the contents of Read Reg1 and Read Reg2, and stores the lower 32 bits of the result in Write Reg.

1. MOV (Immediate) (ALUOp = 101, IMM = 1):

Loads an immediate value directly into Write Reg.