Addressing Modes

The term addressing modes refers to how the operand of an instruction is specified. The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is executed. The different ways in which the location of an operand is specified in instruction are referred to as addressing modes.

Addressing Modes Definition

The addressing modes are used to specify the effective address in an instruction. The effective address contains the object. The object can be a data element or an address that must be evaluated in the instruction. The addressing modes are divided into two:

- Sequential Control Flow Addressing Mode: Focuses on data.
- Transfer of Control Flow Addressing Mode: Focusses on instruction.

Why are Addressing Modes Used?

Addressing modes shows the location of a required object in the computer. The object may be an instruction or data. The output of the addressing mode is an effective address [EA]. The effective address is the actual address of an object. Therefore we can say the object is equal to the content of the effective address. The addressing mode is implemented in the instruction in two ways:

- **Implicit**: The opcode itself specifies the addressing mode used.
- **Explicit**: The mode field is used in the instruction format to specify the type of addressing mode used.

Types of Addressing Modes

The addressing modes are categorized into two types: memory-based addressing mode and transfer control addressing mode. The memory-based addressing mode is further classified into different types that are as follows:

Memory-Based Addressing Modes

- **Implied Addressing Mode:** In this mode, the operands are implicitly specified in the instruction definition.
- **Immediate Addressing Mode:** In this mode, the operand is specified in the instruction itself, or we can say that an immediate mode instruction has an operand rather than an address.
- **Direct Register Addressing Mode:** In this mode, one of the operands is in registers, and the other is taken from memory.

- **Direct Addressing Mode:** In this mode, the address of the memory location that holds the operand is included in the instruction. The effective address is the address part of the instruction.
- **Indirect Addressing Mode:** In this mode, the address field of the instruction gives the address where the effective address is stored in memory.
- **Relative Addressing Mode:** In this mode, the content of the program counter is added to the address part of the instruction to calculate the effective address.
- **Indexed Addressing Mode:** In this mode, the effective address will be calculated as the addition of the content of the index register and the address part of the instruction.

Transfer Of Control Addressing Modes

- **PC Relative Addressing Mode:** This addressing mode is used to access the instruction within the segment. Therefore only one offset address is required.
- **Base register Addressing Mode:** This addressing mode is used to access the instructions between two segments. Therefore, a base address, as well as an offset, is required.

What is a Machine Instruction?

A binary code is used for specifying micro-operations for the computer. Machine Instructions are commands or programs written in the machine code of a machine (computer) that it can recognize and execute. In computer programming, machine instructions are low-level program instructions in binary. Without further processing, these instructions are directly decoded and executed by the computer microprocessor.

Instruction Code

A group of bits is used to instruct the CPU to perform a specific operation. Instructions are encoded as binary instruction codes.

Each instruction code contains an operation code, or opcode, which designates the overall purpose of the instruction. The number of bits allocated for the opcode determined how many different instructions the architecture supports.

Instruction Set

An instruction set is a group of commands for a CPU in machine language. The term can refer to all possible instructions for a CPU or a subset of instructions to enhance its performance in certain situations.

Instruction Cycles

The instruction cycle (also known as the fetch–decode–execute cycle, or simply the fetch-execute cycle) is the cycle that the central processing unit (CPU) follows from

boot-up until the computer has shut down to process instructions. The instruction cycle consists of the following phases that are as follows.

- Fetching an instruction from memory.
- Decoding the instruction.
- Reading the effective address from memory in case of the instruction has an indirect address.
- Execution of the instruction.
- Writing the results back to the memory.

What is an Instruction Format?

An instruction consists of bits, which are grouped up to make fields. Some fields in instruction format are as follows

1. Opcode, which tells about the operation to be performed.
2. Address field designating a memory address or a processor register.
3. Mode field specifying how the operand or effective address is determined.

Different Types of Instruction Formats

Some common types are three address instruction format, Two address instruction format, One address instruction format, and Zero address instruction format.

Three Address Instruction Format

This system contains three address fields (address of operand1, address of operand2 and address where the result needs to be put). The next instruction address is held in a CPU register called Program Counter (PC).

Bits

Add	Result Address	OP1 address	OP2 address
8	24	24	24

Here, the number of bytes required to encode an instruction is 10 bytes.

Each address requires 24 bits = 3 bytes.

Since there are three addresses and one opcode field.

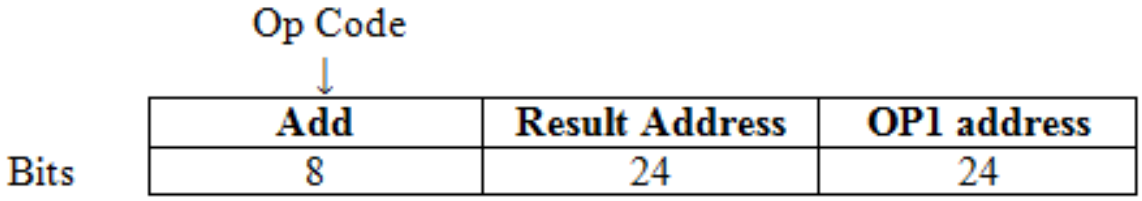
Therefore 3 × 3 + 1 = 10 bytes.

The number of memory access required is 7 words.

4 words for instruction fetch, 2 words for operand fetch, and 1 word for the result to be placed back in memory.

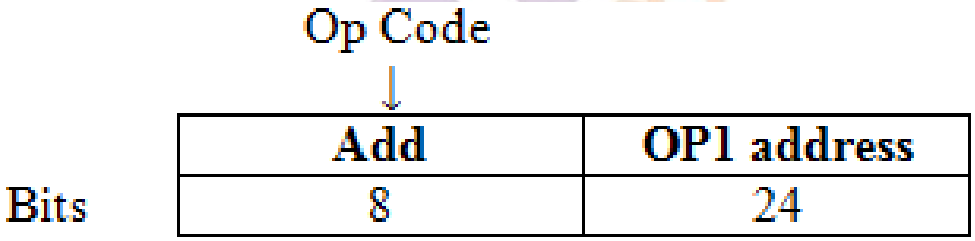
Two Address Instruction Format

Two addresses and an operation field are there in this format. The result is stored in either of the operand address, i.e., either an address of the first or second operand's address. CPU register called Program Counter (PC) contains the address of the next instruction.



One Address Instruction Format

One address field and an operation field. This address is of the first operand. The second operand and the result are stored in a CPU register called Accumulator Register (AR). Since a machine has only one accumulator, it must not be explicitly mentioned in the instruction. A CPU register (i.e., Program Counter (PC) holds the address of the next instruction. In this scenario, two additional instructions are required to load and store the accumulator contents.

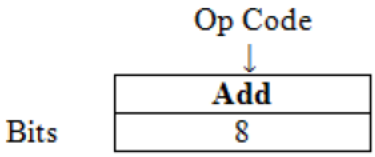


The number of bits required to encode an instruction is 4 bytes. i.e., each address requires 24 bits = 3 bytes. Since there are one address and one operation code field, 1* 3 + 1= 4 bytes.

The number of memory access required is 3 words, i.e., 2 words for instruction fetch +1 word for code for operand fetch.

Zero Address Instruction Format

Stack is included in the CPU for performing arithmetic and logic instructions with no addresses. The operands are pushed onto the stack from memory, and ALU operations are implicitly performed on the top elements of the stack. The address of the next instruction is held in a CPU register called the program counter.



e.g., Add

Top of stack ← Top of stack + second top of the stack.

Addressing Modes with Example

Having discussed the definition of addressing modes, the types of addressing modes, and the uses of addressing modes, we shall now understand the addressing modes with examples:

Example 1:- Consider we have the following values in the given memory locations:

Location	Value
1000	1300
1100	1200
1200	800
1300	1200

Consider that the index register R1 store 200 and is always implicitly used for the indexed addressing mode. What data is loaded into the accumulator if the instruction is “LOAD 1000” for direct addressing mode, indirect addressing mode, and base (indexed) addressing modes, respectively?

Answer: 1300, 1200, 800

Example 2:- Assume the base register contains 32856. The program counter is currently at the 25687-memory location. What is the branch address if the address field of jump instruction contains –30 in the address field and the instruction is designed with the base register addressing modes?

Answer: 32826

What are the Types of Instructions?

The instructions are classified into three categories: data transfer instructions, data manipulation instructions, and program control instructions. These are further classified that are as follows:

- **Data Transfer Instructions**

  Data transfer instructions cause the transfer of data from one location to another without changing the information content. The shared transfers may be between memory and processor registers, between processor registers and input/output.

Typical Data Transfer Instructions

Name	Mnemonic
LOAD	LD
STORE	ST
MOVE	MOV
EXCHANGE	XCH
INPUT	IN
OUTPUT	OUT
PUSH	PUSH
POP	POP

- **Data Manipulation Instructions**

  Data manipulation instructions perform operations on data and provide computational capabilities for the computer. There are three types of data manipulation instructions: Arithmetic instructions, Logical and bit manipulation instructions, and Shift instructions.

Typical Arithmetic Instructions

Name	Mnemonic
INCREMENT	INC
DECREMENT	DEC
ADD	ADD
SUBTRACT	SUB
MULTIPLY	MUL
DIVIDE	DIV
ADD WITH CARRY	ADDC
SUBTRACT WITH BORROW	SUBB
NEGATIVE	NEG

Typical Logical and Bit Manipulation Instructions

Name	Mnemonic
CLEAR	CLR
COMPLEMENT	COM
AND	AND
OR	OR
EXCLUSIVE OR	XOR
CLEAR CARRY	CLRC
SET CARRY	SETC
COMPLEMENT CARRY	COMC
ENABLE INTERRUPT	EI
DISABLE INTERRUPT	DI

Typical Shift Instructions

Name	Mnemonic
LOGICAL SHIFT RIGHT	SHR
LOGICAL SHIFT LEFT	SHL
ROTATE RIGHT	ROR
ROTATE LEFT	ROL

- **Program Control Instructions**

Program control instructions specify conditions for altering the content of the program counter, while data transfer and manipulation instructions specify conditions for data processing operations. The change in the value of a program counter as a result of the execution of a program control instruction causes a break in the sequence of instruction execution.

Typical Program Control Instructions

Name	Mnemonic
BRANCH	BR
JUMP	JMP
SKIP	SKP
CALL	CALL
RETURN	RET
COMPARE	CMP
TEST	TST

Program Interrupt

The program interrupts are used to handle various problems that arise out of normal program sequences. Program interrupts transfer the program control from a currently

running program to another service program due to an external or internal generated request.

Control returns to the original program after the service program is executed. The interrupt is classified into two categories that are as follows:

Based on Masking

1. **Maskable Interrupts:** It may be a hardware or a software interrupt that can be masked for the future.
2. **Non-Maskable Interrupts:** A non-maskable interrupt (NMI) is a hardware interrupt that standard interrupt-masking techniques cannot ignore. It typically occurs to signal attention for non-recoverable hardware errors.

Based on devices

1. **External interrupt:** External interrupts come from Input-Output (I/O) devices or a timing device.
2. **Internal interrupt:** Internal interrupts arise from illegal or erroneous use of an instruction or data. External and internal interrupts from signals that occur in the hardware of the CPU.
3. **Software interrupt:** A Software interrupt is initiated by executing an instruction.

Complex Instruction Set Computer (CISC)

Computer architecture is described as the design of the instruction set for the processor. A computer with many instructions is classified as a complex instruction set computer. The CISC processors typically have 100 to 250 instructions.

The instructions in a typical CISC processor provide direct manipulation of operands residing in memory. As more instructions and addressing modes are incorporated into a computer, more hardware logic is needed to implement and support them, which may cause the computations to slow down.

Reduced Instruction Set Computer (RISC)

RISC architecture reduces the execution time by simplifying the computer's instruction set. In the RISC processors, there are relatively few instructions and few addressing modes. In RISC processors, all operations are done within the CPU's registers.