Addressing modes

Implied/ Implicit Addressing Mode

The operands are implicitly specified in the instruction's definition. Consider the example, the instruction "complement accumulator" is an implied-mode instruction as the operand in the accumulator register is implied in the instruction definition. All register reference the instructions that use an accumulator are implied-mode instructions. Zero-address instructions in a stack-organized computer are also implied-mode instructions because the operands are implied to be on top of the stack.

Immediate Addressing Mode

The operand is defined in the instruction itself. This mode instruction has an operand field instead of an address field. The operand field contains the actual operand used with the specified operation in the instruction. The immediate-mode instructions help initialize registers to a constant value.

For example, ADD 8 will increment the value stored in the accumulator by 8.

Register Direct Addressing Mode

The operands that reside within the CPU are stored in the registers. The specific register is selected from a register field in the instruction. No reference to the memory is required to fetch the operand. The only difference between the Direct addressing mode and the register direct addressing mode is that the instruction address field refers to a CPU register instead of the main memory.

Advantage: This mode provides faster memory access to the operands.
Disadvantage: It has limited address space, and using multiple registers can help boost the performance, but it complicates the instructions.

Register Indirect Addressing Mode

The instruction defines a register in the CPU that stores the effective address of the operand in memory. Only one reference to the memory is required to fetch the operand. The specified register contains the address of the operand instead of the operand. The only difference between the Indirect addressing mode and the register indirect addressing mode is that the instruction address field refers to a CPU register.

For example: ADD R1, R2: Here, the content of R2 is added to R1. R1 R2 represents registers. **Advantage:** The instruction address field uses fewer bits to select a register than required to specify a memory address directly.

Auto-Increment Addressing Mode

This mode is a special Register Indirect Addressing Mode case. The register is incremented/decremented after or before its value is utilized.
EA = content of the register.

The content of the register is increment automatically by step size 'd' after accessing the operand, where the step size 'd' depends on the size of the accessed operand. Only one reference memory is required to fetch the operand.

Consider an example:

Here, the Effective Address (R) =178, and the operand in AC are 7. After loading R1 is incremented by 1 and becomes 179.

Auto-Decrement Addressing Mode

This mode is also a special Register Indirect Addressing Mode case.
EA = content of the register - step size.

The content of the register is decremented by step size 'd' after accessing the operand, where the step size 'd' depends on the size of the accessed operand. Only one reference memory is required to fetch the operand.

Consider an example:

Here, register R1 is decremented by 1 (400-1=399) prior to the instruction execution which implies that the operand loaded to the AC is of address 1099H instead of 1088H. Hence, the Effective Address is 1099H.

Direct Addressing Mode

The effective address of the operand resides in the address field of the instruction. The operand resides in the memory, and the address field of the instruction gives its address. Only one reference to the memory is required to fetch the operand, and no additional calculations need to be done to find the effective address of the operand. It is also known as absolute addressing mode.

For example, ADD X will increment the value stored in the accumulator by the value stored at X's memory.

Indirect Addressing Mode

The address field of the instruction gives the address of the memory location that contains the effective address of the operand. Two references to the memory are required to fetch the operand: The control fetches the instruction from memory and uses its address part to reaccess the memory that stores the effective address. This addressing mode slows down the execution as it requires multiple memory lookups to find the operand.

Displacement Addressing Mode

The indexed register content is added to the instruction's address part to obtain the effective address of the operand.

EA = A + (R)
Here, the address field holds two values, A: Base value R: displacement value.

Relative Addressing Mode

This mode is another version of the displacement address mode. The program counter's content is added to the instruction's address part to obtain the effective address.

EA = A + (PC)
Here, EA: Effective address, PC: program counter.

The instruction's address part is usually a signed number that can be positive or negative. After the instruction's address is fetched, the value of the program counter increases immediately, irrespective of whether the fetched instruction has been executed or not. PC: It contains the address of the next instruction to be executed.

Consider an example:

The program counter contains the number 422, and the address part of the instruction contains the number 17. The instruction at location 421 is read during the fetch phase, and the program counter is then incremented by one to 422 + 17 = 439.

Indexed Addressing Mode

The index register's content is added to the instruction's address to obtain the effective address.
EA = content of index register + Instruction address part

Base Register Addressing Mode

This mode is another version of the displacement address mode. To obtain the effective address, the base register's content is added to the instruction's address.
EA = A + (R)
A: Displacement, R: Pointer to the base address.

Stack Addressing Mode

The operand resides at the top of the stack. Consider an example, ADD: This instruction will POP two items from the stack, add them, and at last, will PUSH the result to memory references of Addressing Modes.
It helps in reducing the number of bits in the instruction's addressing field.
It facilitates pointers, indexing of data, and counters for loop controls.

Applications of Addressing Modes

Addressing Mode	**Applications**
Immediate Addressing Mode	Initialize the register to a constant value.
Direct Addressing Modes and Register Direct Addressing Mode	Helps access static data and implement variables.
Indirect Addressing Modes and Register Indirect Addressing Mode	Helps implement pointers and pass arrays as parameters.
Relative Addressing Mode	Helps in program relocation at runtime. And in changing the sequence of instructions during execution.
Index Addressing Mode	Helps in the array and record implementation.
Base Register Addressing Mode	Helps in writing relocatable code and handling recursive procedures.
Auto-Increment Addressing Mode and Auto-Decrement Addressing Mode	Helps implements loops and stacks.

Advantages of Addressing Modes

- They improve performance by efficiently utilizing the CPU cache and reducing the memory read latency.

- Addressing Modes are used for implementing complex data structures in memory as they provide mechanisms such as indexing.

- Program sizes can be reduced drastically as the code can be compacted which allows faster execution of instructions.

- Addressing modes give you the flexibility to use different ways of specifying the address of operands in your instructions.