Addressing modes

The different ways to which the location of an operand is specified in an instruction are referred as addressing mode.

computers use addressing mode techniques for the purpose of accomodating one or both of the following reasons.

(i) By providing facilities as pointers to memory, counters for loop control, indexing of data, and Program relocation.

(ii) To reduce the no of bits in the addressing field of the instruction.

(i) Implied Mode: In this mode the operands are specified implicitly in the definition of the instruction.

Ex: * complement accumulator is an implied-mode instruction.

* All register reference instructions that use an accumulator are implied-mode instructions.
* Zero-address instructions in a stack-organized computer are implied mode instructions. Since the operands are implied to be on the top of the stack.

(ii) Immediate Mode :- In this mode the operand is specified in the instruction itself.

Ex: Move 200immediate, Ro

OR

we can write

Move #200, Ro

- this instruction places the value 200 in register Ro.

(iii) Register mode: The operand is the contents of a Processor register; the name or address of the register is given in the instruction. A k-bit field can specify any one of 2^k registers.

Ex: Move LOCA, R_2

(iv) Absolute/Direct mode :- The operand is in a memory location, whose address is given explicitly in the instruction.

Ex: MOVE LOCA, R_2

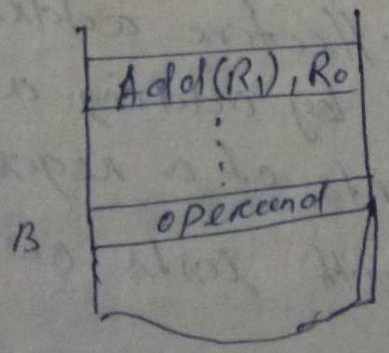
A declaration such as

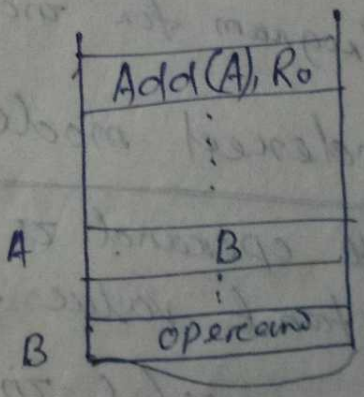
Integer A, B

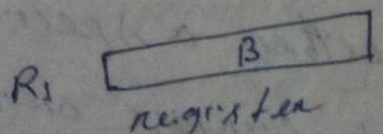
(v) Indirect mode: The effective address of the operand is the contents of a register or memory location whose address appears in the instruction.

* If address of the operand resides in a Processor register, then it is known as Register Indirect mode.

* The advantage of a register indirect mode instruction is that the address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.



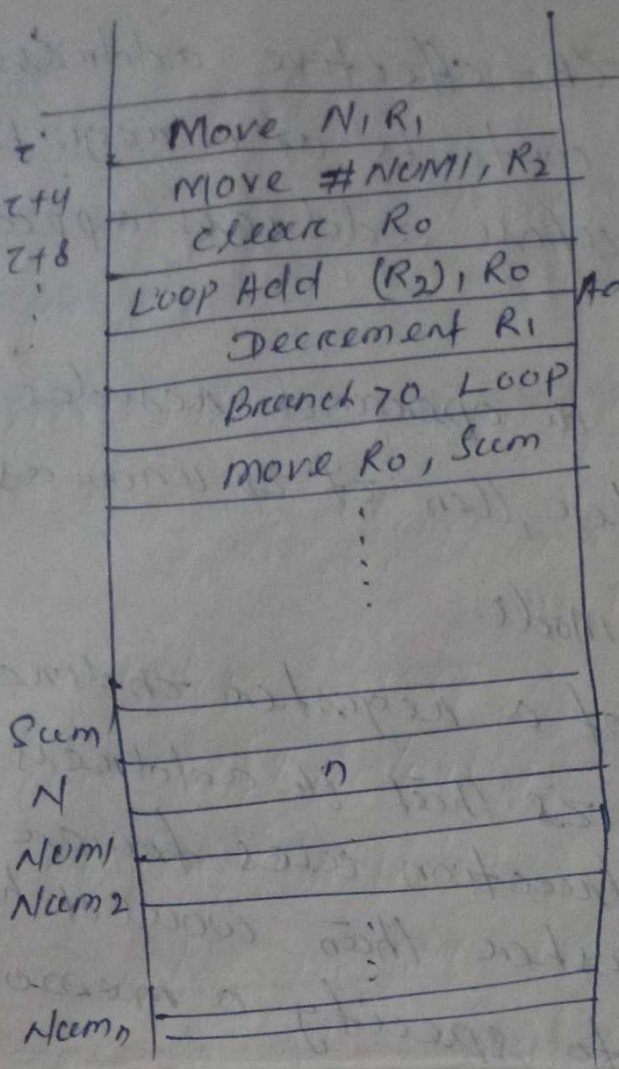




(i) Through a general-purpose register.

(ii) Through a memory location

* The register or memory location that contains the address of an operand is called a pointer. Indirect addressing mode is used as a pointer.



Program for indirect addressing mode.

(vc) Indexed mode: the effective address of the operand is generated by adding a constant value to the contents of a register

* It is useful in dealing with lists and arrays.
* The register used may be either a special register provided for this purpose or it may be any one of a set of general-purpose registers in the Processor. In both the cases it is referred as an index register.

3(c)

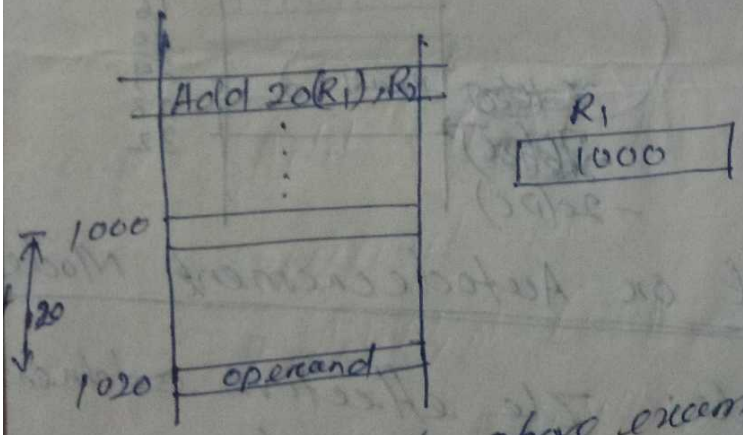
* We indicate the index mode as

X(Ri)

where X denotes the constant value contained in the instruction and Ri is the name of the register involved. The effective address of the operand is given by

EA = X + [Ri]

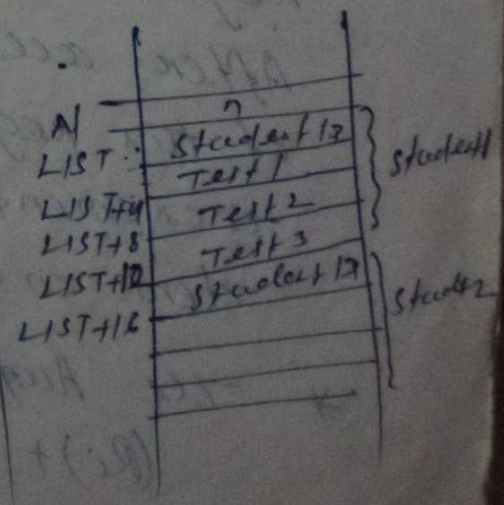
and the contents of the index register are not changed in the process of generating the effective address.



* Here in the above example, the index register R_1, contains the address of a memory location, and the value X defines an offset.

Program

```
Move  #LIST, R0
clear R1
clear R2
clear R3
move  N, R4
Add   4(R0), R1
Add   8(R0), R2
Add   12(R0), R3
Add   #16, R0
Decrement R4
Branch>0  Loop
move  R1, SUM1
move  R2, SUM2
move  R3, SUM3
```

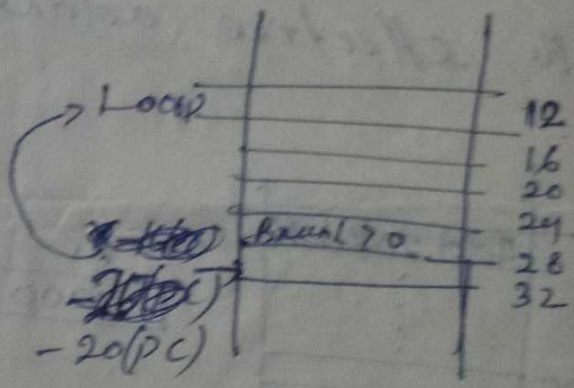


(vii) Relative Addressing mode: The effective address of the operand is generated by adding a constant value to the contents of the Program counter.

* We indicate this addressing mode by

X(PC)

* X(PC) can be used to address a memory location that is X bytes away from the location presently pointed to by the Program counter.



(viii) Autoincrement or Autodecrement Mode

Autoincrement mode: The effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are automatically incremented to point to the next item in a list.

* The Autoincrement mode is written as

(Ri)+

Autodecrement mode :- The content of a register specified to the instruction are first automatically decremented and are used as the effective address of the operand.

* The autodecrement mode is written as

  $-(R_i)$

Ex:

```
        Move    N, R1
        MOVE    #NUM1, R2
        clear   R0
Loop    Add     (R2)+, R0
        Decrement R1
        Branch>0 LOOP
        move    R0, SUM
```