Chapter 8

CENTRAL PROCESSING UNIT

Big Endian vs Little Endian

- Endianness relates to BYTE ADDRESSING ORDER.

- Little endian means the lower significant bytes get the lower addresses.
- Big endian means the higher significant bytes gets the lower addresses.

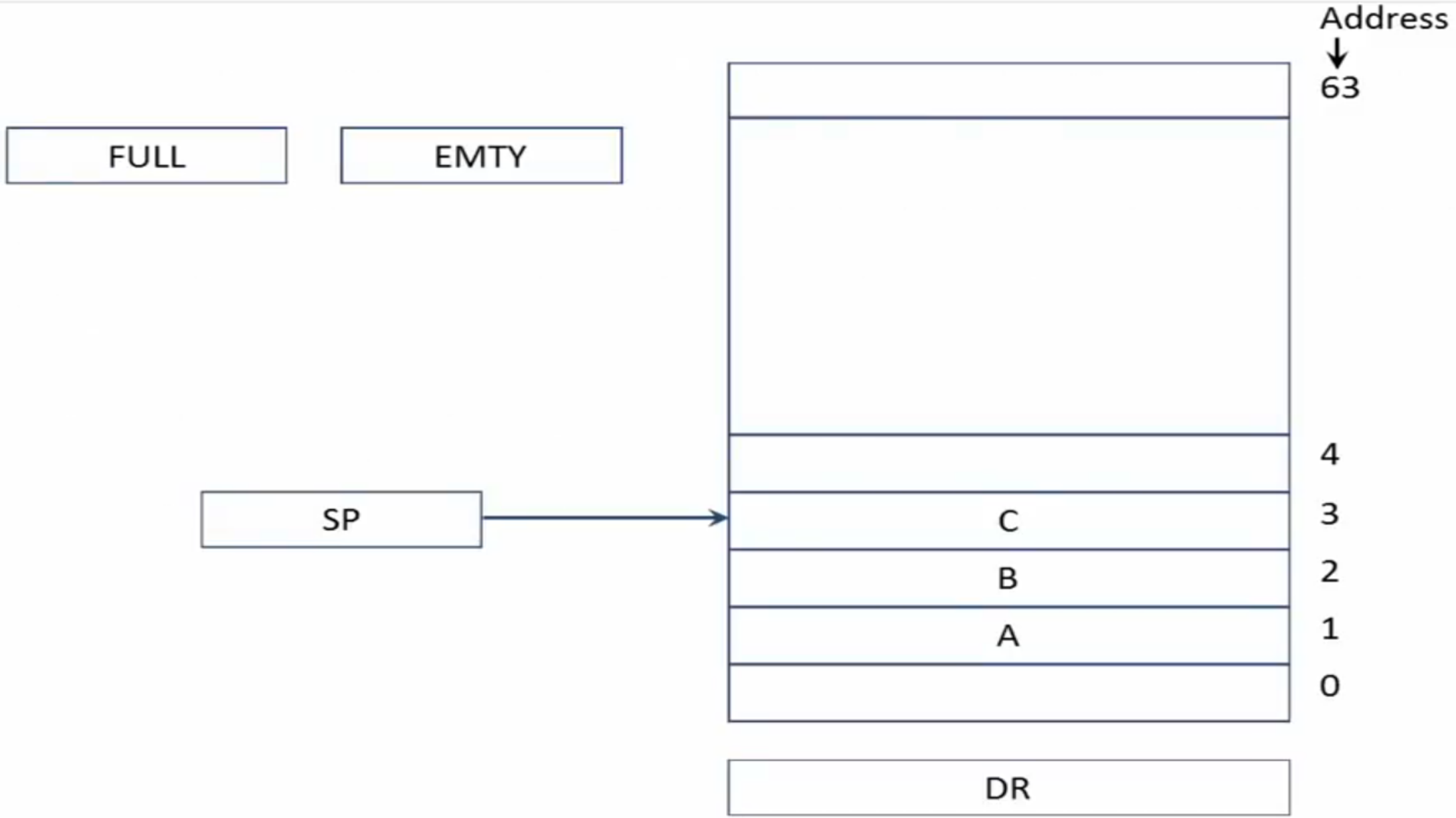
Example. -12 is represented in memory as 1111 1111 1111 0100
LITTLE ENDIAN REPRESENTATION:
000000: F4
000001: FF
BIG ENDIAN REPRESENTATION:
000000: FF
000001: F4

STACK ORGANIZATION

Stack Organization

- A stack is a storage device that stores information in such a manner that the item stored last is the first item retrieved (LIFO).
- The register that holds the address for the stack is called a stack pointer (SP) because its value always points at the top item in the stack.
- There are two types of stack organization
  1. Register stack – built using registers
  2. Memory stack – logical part of memory allocated as stack

Register Stack



Register Stack

- **PUSH Operation**

  SP ← SP + 1

  M[SP] ← DR

  IF (SP= 0) then (FULL ← 1)

  EMTY ← 0

- **POP Operation**

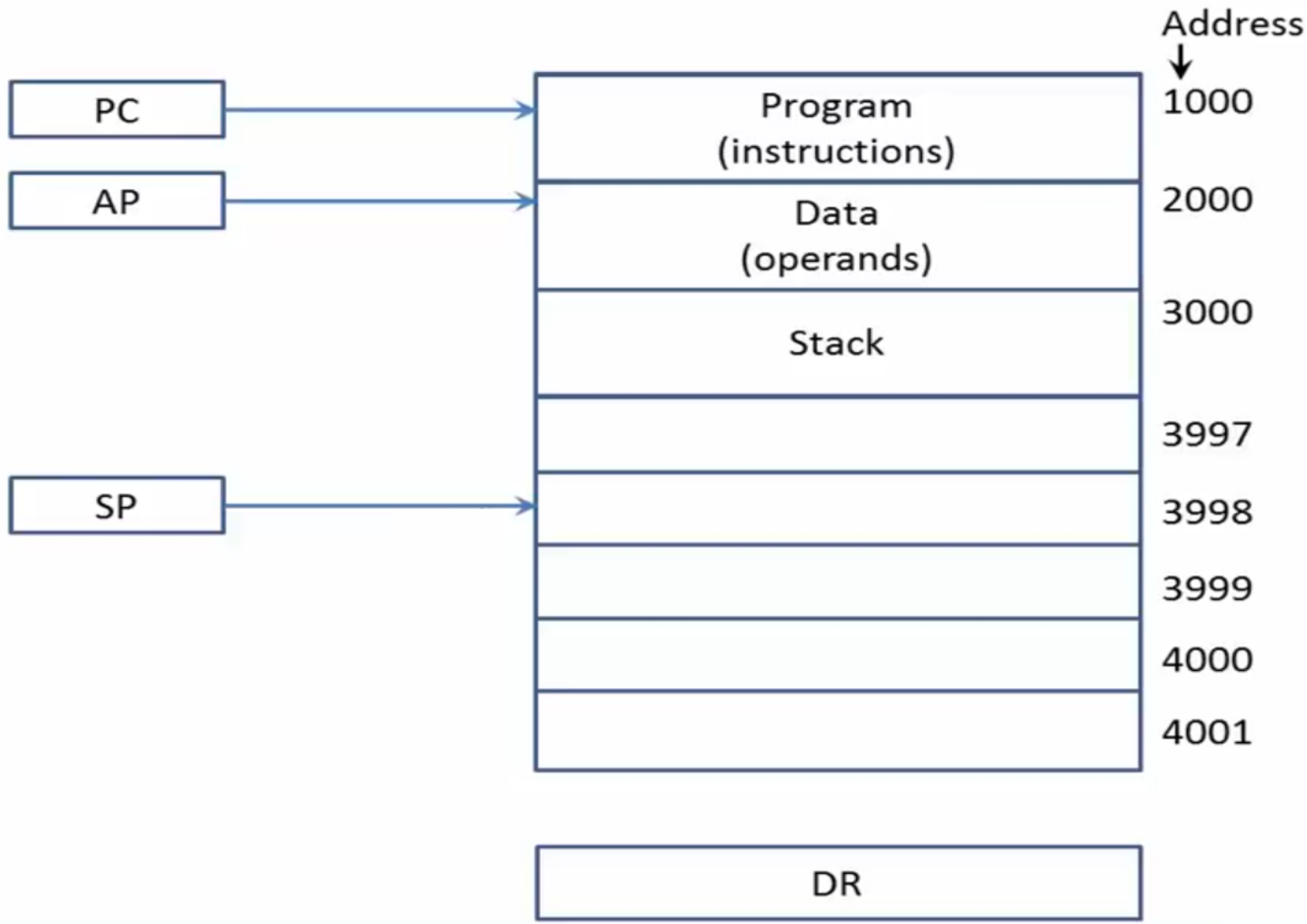
  DR ← M[SP]

  SP ← SP - 1

  IF (SP= 0) then (EMTY ← 1)

  FULL ← 0

Memory Stack



Memory Stack

- **PUSH Operation**

  $SP \leftarrow SP - 1$

  $M[SP] \leftarrow DR$

- **POP Operation**

  $DR \leftarrow M[SP]$

  $SP \leftarrow SP + 1$

Reverse Polish Notation

- The common mathematical method of writing arithmetic expressions imposes difficulties when evaluated by a computer.
- The Polish mathematician Lukasiewicz showed that arithmetic expressions can be represented in prefix notation as well as postfix notation.

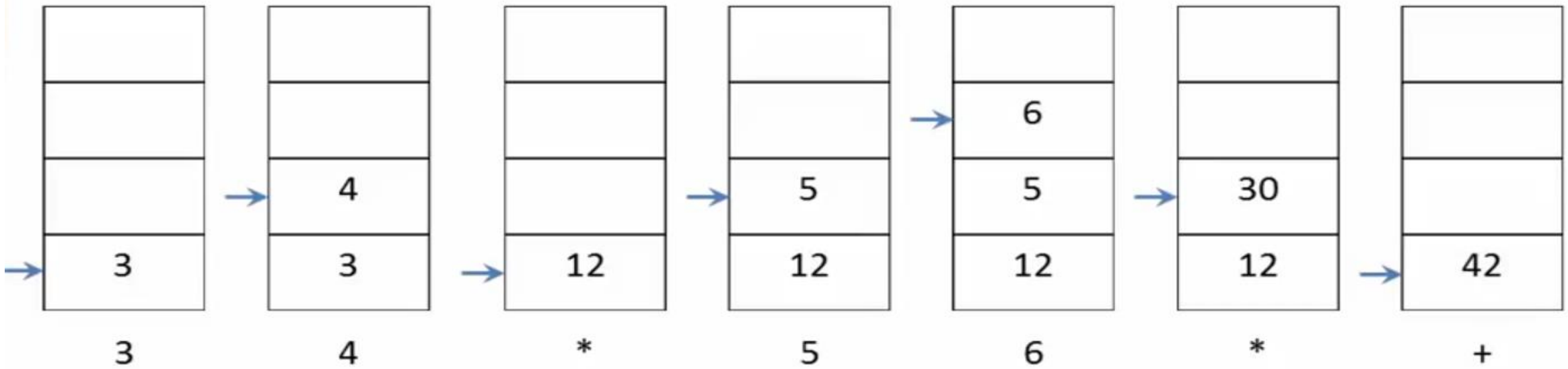
Infix	Prefix or Polish	Postfix or reverse Polish
$A + B$	$+ AB$	$AB +$

$A * B + C * D \longrightarrow AB * CD * +$

Reverse Polish

Evaluation of Arithmetic Expression

(3 * 4) + (5 * 6) ⟶ 3 4 * 5 6 * +



INSTRUCTION FORMATS

Instruction Formats

- Instructions are categorized into different formats with respect to the operand fields in the instructions.
  1. Three Address Instructions
  2. Two Address Instruction
  3. One Address Instruction
  4. Zero Address Instruction
  5. RISC Instructions

Three Address Instruction

- Computers with three-address instruction formats can use each address field to specify either a processor register or a memory operand.
- The program in assembly language that evaluates X = (A + B) * (C + D) is shown below.

```
ADD    R1, A, B       R1←M[A]+M[B]
ADD    R2, C, D       R2←M[C]+M[D]
MUL    X, R1, R2      M[X]←R1 * R2
```

- The advantage of three-address format is that it results in short programs when evaluating arithmetic expressions.
- The disadvantage is that the binary-coded instructions require too many bits to specify three addresses.

Two Address Instruction

- Two address instructions are the most common in commercial computers. Here again each address field can specify either a processor register or a memory word.
- The program to evaluate X = (A + B) * (C + D) is as follows:

```
MOV    R1, A          R1← M[A]
ADD    R1, B          R1← R1+ M[B]
MOV    R2, C          R2← M[C]
ADD    R2, D          R2← R2+ M[D]
MUL    R1, R2         R1← R1  * R2
MOV    X, R1          M[X]← R1
```

One Address Instruction

- One address instructions use an implied accumulator (AC) register for all data manipulation.
- For multiplication and division these is a need for a second register.
- However, here we will neglect the second register and assume that the AC contains the result of all operations.
- The program to evaluate X = (A + B) * (C + D) is

```
LOAD   A        AC← M[A]
ADD    B        AC← AC+M[B]
STORE  T        M[T]←AC
LOAD   C        AC← M[C]
ADD    D        AC← AC+M[D]
MUL    T        AC← AC*M[T]
STORE  X        M[X]←AC
```

Zero Address Instruction

- A stack-organized computer does not use an address field for the instructions ADD and MUL.
- The PUSH and POP instructions, however, need an address field to specify the operand that communicates with the stack.
- The program to evaluate X = (A + B) * (C + D) will be written for a stack-organized computer.
- To evaluate arithmetic expressions in a stack computer, it is necessary to convert the expression into reverse polish notation.

```
PUSH   A        TOS← M[A]
PUSH   B        TOS← M[B]
ADD             TOS← (A+B)
PUSH   C        TOS← M[C]
PUSH   D        TOS← M[D]
ADD             TOS← (C+D)
MUL             TOS← (C+D) * (A+B)
POP    X        M[X] ← TOS
```

RISC Instruction

- The instruction set of a typical RISC processor is restricted to the use of load and store instructions when communicating between memory and CPU.
- All other instructions are executed within the registers of the CPU without referring to memory.
- A program for a RISC type CPU consists of LOAD and STORE instructions that have one memory and one register address, and computational-type instructions that have three addresses with all three specifying processor registers.
- The following is a program to evaluate X = (A + B) * (C + D)

```
LOAD   R1, A          R1← M[A]
LOAD   R2, B          R2← M[B]
LOAD   R3, C          R3← M[C]
LOAD   R4, D          R4← M[D]
ADD    R1, R1, R2     R1← R1+R2
ADD    R3, R3, R4     R3← R3+R4
MUL    R1, R1, R3     R1← R1*R3
STORE  X, R1          M[X] ← R1
```

ADDRESSING MODES

Addressing Modes

- The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.
- Computers use addressing mode techniques for the purpose of accommodating one or both of the following provisions:
  1. To give programming versatility to the user by providing such facilities as pointers to memory, counters for loop control, indexing of data, and program relocation.
  2. To reduce the number of bits in the addressing field of the instruction.

Addressing Modes

1. Implied Mode
2. Immediate Mode
3. Register Mode
4. Register Indirect Mode
5. Autoincrement or Autodecrement Mode
6. Direct Address Mode
7. Indirect Address Mode
8. Relative Addressing Mode
9. Indexed Addressing Mode
10. Base Register Addressing Mode

1. Implied Mode

- Operands are specified *implicitly* in the definition of the instruction.
- For example, the instruction "complement accumulator (CMA)" is an implied-mode instruction because the operand in the accumulator register is implied in the definition of the instruction.
- In fact, all register reference instructions that use an accumulator and zero address instructions are implied mode instructions.

2. Immediate Mode

- Operand is specified in the instruction itself.
- In other words, an immediate-mode instruction has an operand field rather than an address field.
- The operand field contains the actual operand to be used in conjunction with the operation specified in the instruction.
- Immediate mode of instructions is useful for initializing register to constant value.
- E.g. MOV R1, 05H

3. Register Mode

- Operands are in registers that reside within the CPU.
- The particular register is selected from a register field in the instruction.
- E.g. MOV AX,BX

  move value from BX to AX register

4. Register Indirect Mode

- In this mode the instruction specifies a register in the CPU whose contents give the address of the operand in memory.
- Before using a register indirect mode instruction, the programmer must ensure that the memory address of the operand is placed in the processor register with a previous instruction.
- The advantage of this mode is that address field of the instruction uses fewer bits to select a register than would have been required to specify a memory address directly.
- E.g. MOV [R1], R2

  value of R2 is moved to the memory location specified in R1.

5. Autoincrement or Autodecrement Mode

- This is similar to the register indirect mode expect that the register is incremented or decremented after (or before) its value is used to access memory.
- When the address stored in the register refers to a table of data in memory, it is necessary to increment or decrement the register after every access to the table. This can be achieved by using the increment or decrement instruction.

6. Direct Address Mode

- In this mode the effective address is equal to the address part of the instruction.
- The operand resides in memory and its address is given directly by the address field of the instruction.
- E.g. ADD 457

7. Indirect Address Mode

- In this mode the address field of the instruction gives the address where the effective address is stored in memory.
- Control fetches the instruction from memory and uses its address part to access memory again to read the effective address.
- The effective address in this mode is obtained from the following computational:

  Effective address = address part of instruction + content of CPU register

8. Relative Address Mode

- In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address.
- The address part of the instruction is usually a signed number which can be either positive or negative.

Effective address = address part of instruction + content of PC

9. Indexed Addressing Mode

- In this mode the content of an index register is added to the address part of the instruction to obtain the effective address.
- The indexed register is a special CPU register that contain an index value.
- The address field of the instruction defines the beginning address of a data array in memory.
- Each operand in the array is stored in memory relative to the begging address.

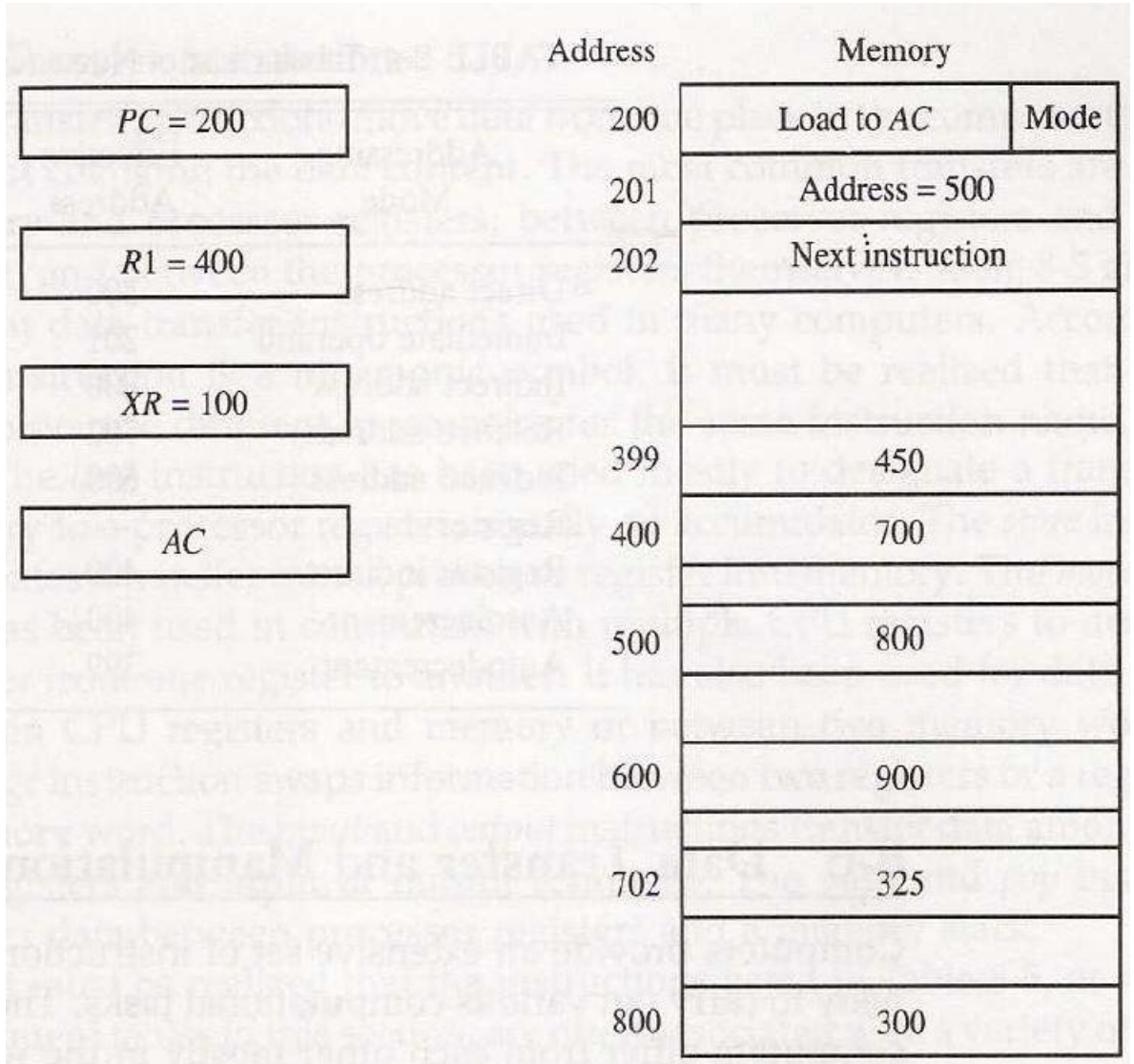
Effective address = address part of instruction + content of index register

10. Base Register Addressing Mode

- In this mode the content of a base register is added to the address part of the instruction to obtain the effective address.
- A base register is assumed to hold a base address and the address field of the instruction gives a displacement relative to this base address.
- The base register addressing mode is used in computers to facilitate the relocation of programs in memory.

Effective address = address part of instruction + content of base register

EXAMPLE – ADDRESSING MODES



Addressing Mode	Effective Address	Content of *AC*
Direct address	500	800
Immediate operand	201	500
Indirect address	800	300
Relative address	702	325
Indexed address	600	900
Register	—	400
Register indirect	400	700
Autoincrement	400	700
Autodecrement	399	450