

Addressing Modes

Computer System Architecture

By

M. Morris Mano

Addressing Modes

- Specify the way the operands are selected during program execution.
- Usage
 - To give programming flexibility to the user
 - pointers to memory, counters for loop control, indexing of data,
 - To reduce the number of bits in the addressing field of the inst.

Operation cycle of the computer

- **Instruction Cycle- 3 phases**
 - 1) Fetch the inst. from memory.
 - 2) Decode the inst. (determine the operation, addressing mode and location of the operands)
 - 3) Execute the inst.
- **Program Counter (*PC*)**
 - Keeps track of the insts. in the program stored in memory.
 - Holds the address of the inst. to be executed next.
 - Incremented each time an inst. is fetched from memory

Addressing Mode of the Instruction

Some computers use

1) Distinct Binary Code

- Instruction Format



2) Single Binary Code

- Designates both Operation and Addressing Mode.

Different Types of addressing mode

1. Implied Addressing Mode
2. Immediate Addressing Mode
3. Direct Addressing Mode
4. Indirect Addressing Mode
5. Register Direct Addressing Mode
6. Register Indirect Addressing Mode
7. Displacement Addressing Mode (combines the direct addressing and register addressing modes)
 1. Relative Addressing Mode
 2. Indexed Addressing Mode
 3. Base Addressing Mode
8. Auto Increment and Auto Decrement Addressing Mode

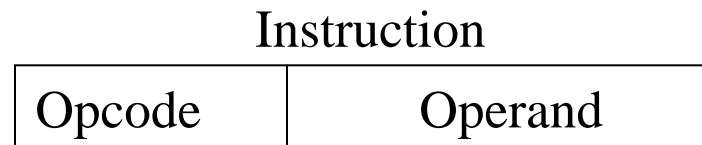
1. Implied Mode

- Operands are specified implicitly in the instruction.
- No address field is required
- 0-address inst. are implied mode inst.
- Used by Stack-organized computer
- *Examples:*
 - **COM** : Complement Accumulator
 - Operand in AC is implied in the inst.
 - **ADD**
 - Operands are implied to be on top of the stack.
- *Effective Address (EA) = AC or Stack[SP]*

2. Immediate Mode

- Operand is specified in the inst. Itself.
- Operand field contains the actual operand.
- Useful for initializing registers to a constant value.
- **Advantage:** No memory Reference, fast
- **Disadvantage:** Limited operand magnitude
- **Example:**

- **LD #NBR**
- **Mov R1, #200**



3. Register Mode

- Operands are in registers.
- Register is selected from a register field in the inst.
 - k-bit register field can specify any one of 2^k registers.
- **Advantage:** No memory reference, shorter instructions, faster instruction fetch, very fast execution
- **Disadvantage:** Limited address space as limited number of registers

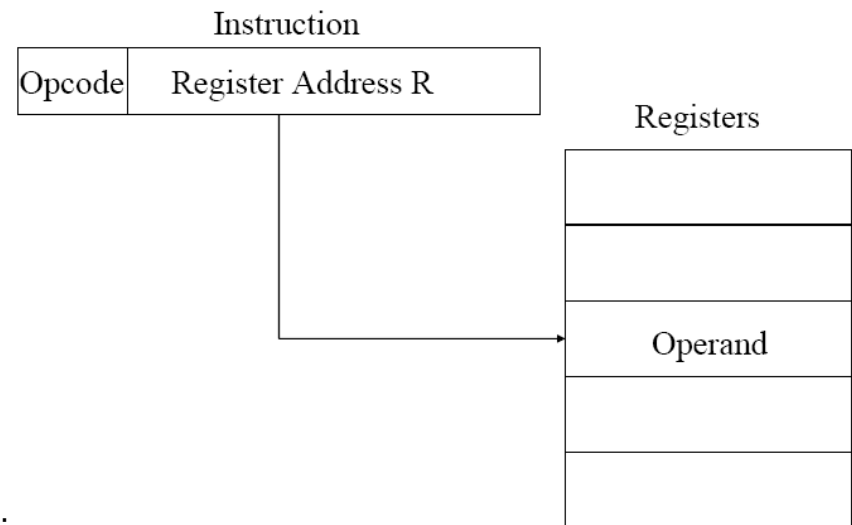
- **Example:**

- **LOAD R1**

$AC \leftarrow R1$

Implied Mode

- **MOV R1,R2**



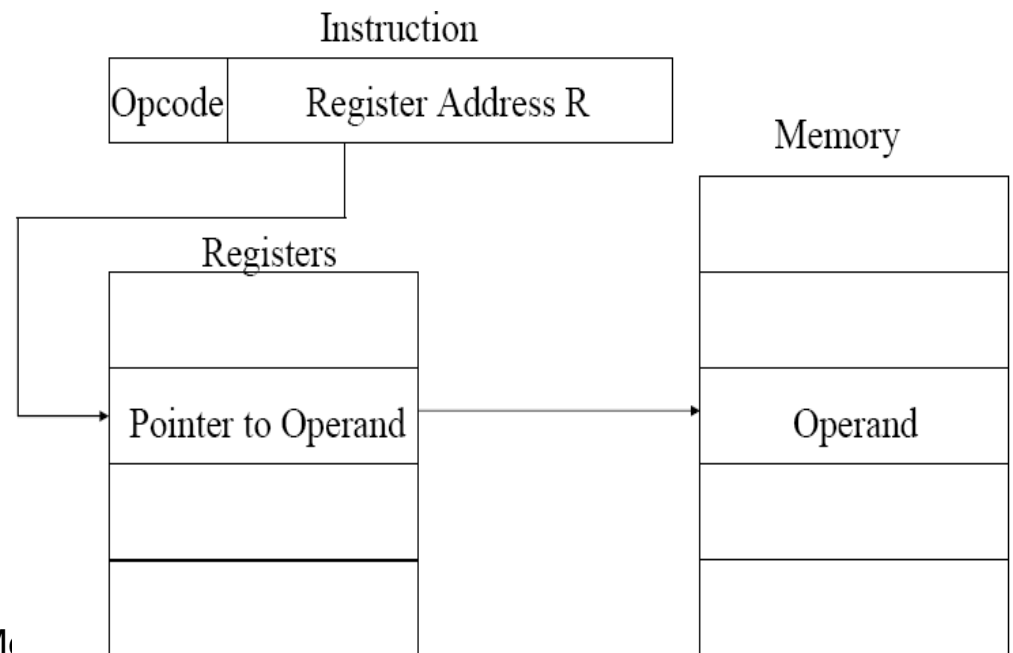
4. Register Indirect Mode

- Register contains the address of the operand.
- **Advantage:** address field of the inst. uses fewer bits to select a register than bits required to select a memory address
- **Disadvantage:** Extra memory reference
- **Example:**

– **LOAD (R1)**

$AC \leftarrow M[R1]$

– **MOV R1,(R2)**



5. Autoincrement or Autodecrement Mode

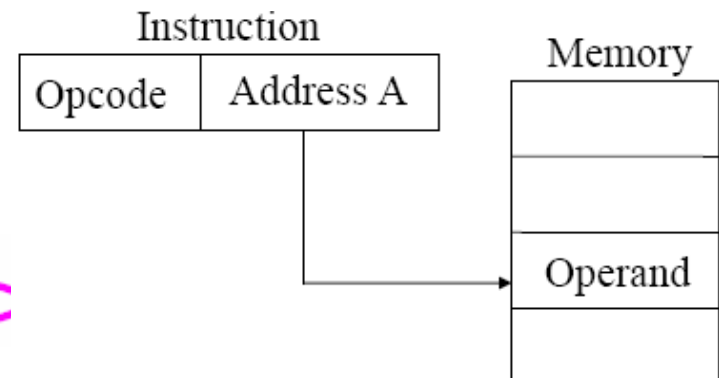
- Similar to the register indirect mode except that
 - the register is *incremented after* its value is used to access memory
 - the register is *decrement before* its value is used to access memory
- **Example (Autoincrement):**
 - **LOAD (R1)+**
$$AC \leftarrow M[R1], R1 \leftarrow R1 + 1$$
- **2 forms post and pre:**
 - **Mov R1,(R2)+** → post increment
 - **Mov R1,+(R2)** → pre increment
 - **Mov R1,(R2)-** → post decrement
 - **Mov R1,-(R2)** → pre decrement

6. Direct Addressing Mode

- Operand resides in memory and its address is given in the inst.
- In a branch-type inst. address field specifies the actual branch address.
- **Advantage:** Simple memory reference to access data, no additional calculations to work out effective address
- **Disadvantage:** Limited address space
- **Example:**
 - **LOAD ADR**

$AC \leftarrow M[ADR]$

ADR = Address part of Instruction



– **MOV R1,2000**

7. Indirect Addressing Mode

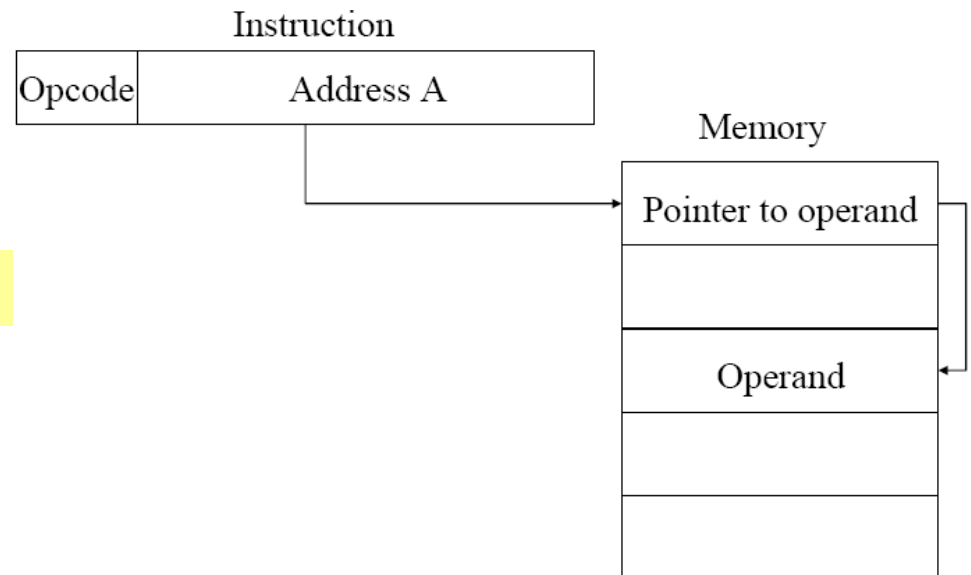
- Address field of inst. gives the address where the effective address is stored in memory
- **Advantage:** Large address space, may be nested, multilevel or cascaded
- **Disadvantage:** Multiple memory accesses to find the operand, hence slower

- **Example:**

- **LOAD @ADR**

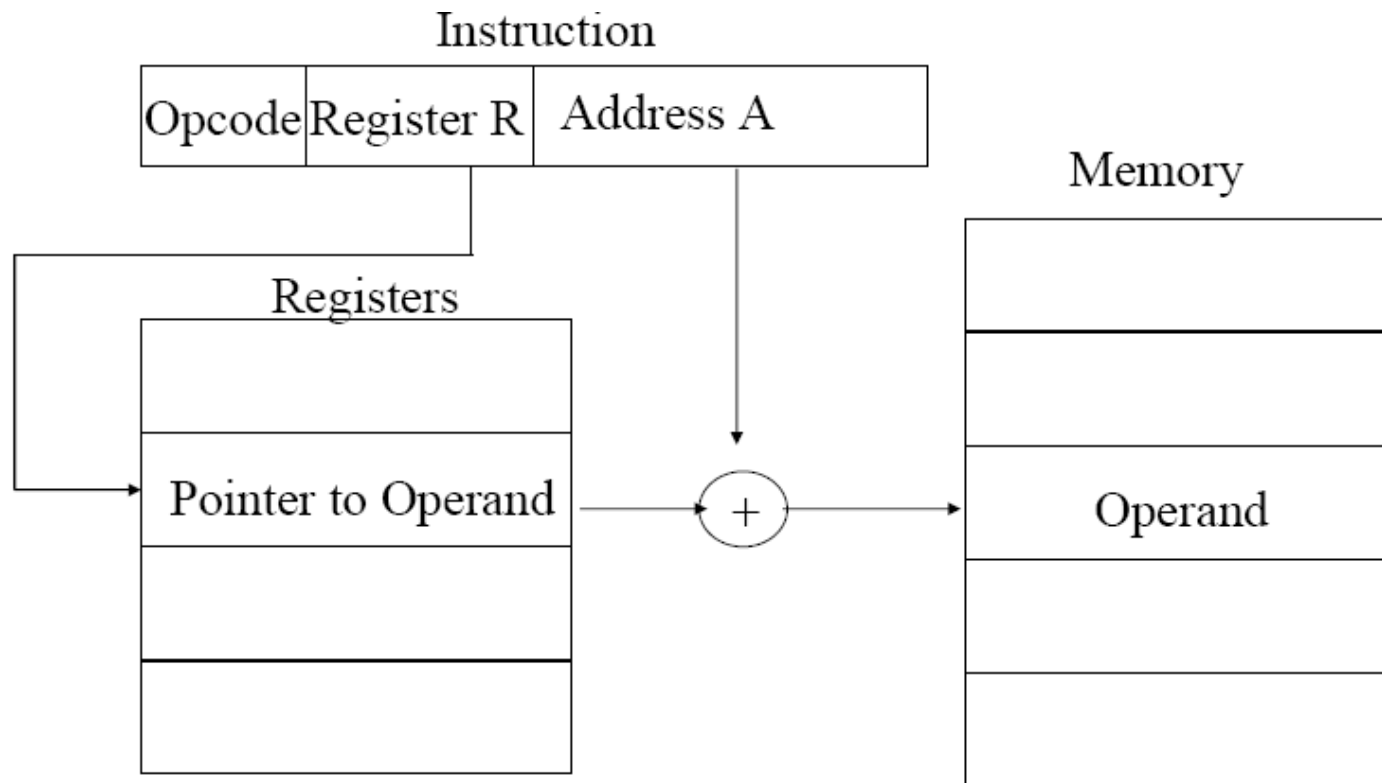
$AC \leftarrow M[M[ADR]]$

- **MOV R1,(2000)**



8. Displacement Addressing Mode

- $EA = A + (R)$
- Address field holds two values
 - A = Base value
 - R = register that holds displacement
 - Or vice-versa



(i) Relative Addressing Mode

- PC is added to the address part of the instruction to obtain the effective address
- $EA = [PC] + \text{address part of the inst.}$
- **Advantage:** Flexibility
- **Disadvantage:** Complexity
- *Example:*
 - **LOAD \$ADR**

$$AC \leftarrow M[PC + ADR]$$

(ii) Indexed Addressing Mode

- XR (*Index register*) is added to the address part of the instruction to obtain the effective address
- $EA = [\text{Index reg.}] + \text{address part of the inst.}$
- **Advantage:** Flexibility, good for accessing arrays
- **Disadvantage:** Complexity
- **Example:**
 - **LOAD ADR(XR)**


$$AC \leftarrow M[ADR + XR]$$

(iii) Base Register Addressing Mode

- EA=Content of a base register + address part of the inst.
- Similar to the indexed addressing mode except that the register is now called a *base register (BR)* instead of an *index register*.

- **Example:**

- **LOAD ADR(BR)**

$$AC \leftarrow M[BR + ADR]$$


- Base register hold a base address
 - The address field of the instruction gives a displacement relative to this base address

Basic addressing modes differences

Mode	Algorithm	Advantage	disadvantage
Immediate	Operand=A	No memory reference	Limited operand magnitude
Direct	EA=A	Simple	Limited address space
Indirect	EA=(A)	Large address space	Multiple memory references
Register	EA=R	No memory reference	Limited address space
Register indirect	EA=(R)	Large address space	Extra memory refernce
Displacement	EA=A+(R)	Flexibility	Complexity
Stack	EA=top of stack	No memory reference	Limited applicability

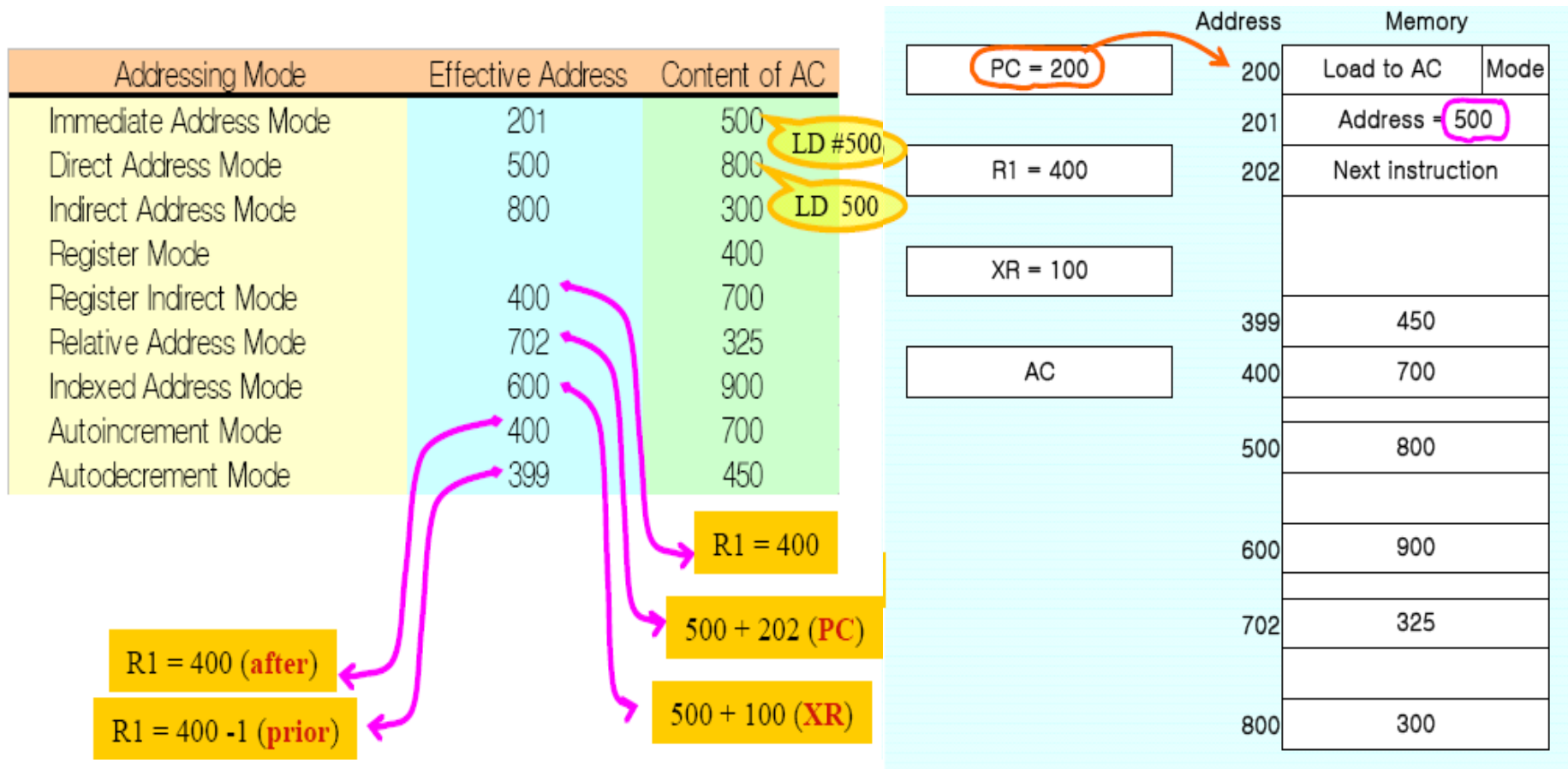
Problems

- Find the effective address and the content of AC for the given data.

Address		Memory	
PC = 200	200	Load to AC	Mode
	201	Address = 500	
	202	Next instruction	
R1 = 400			
XR = 100	399	450	
	400	700	
AC	500	800	
	600	900	
	702	325	
	800	300	

Addressing Mode	Effective Address		Content of AC
Direct Address	500	AC \leftarrow (500)	800
Immediate operand	201	AC \leftarrow 500	500
Indirect address	800	AC \leftarrow ((500))	300
Relative address	702	AC \leftarrow (PC + 500)	325
Indexed address	600	AC \leftarrow (XR + 500)	900
Register	-	AC \leftarrow R1	400
Register Indirect	400	AC \leftarrow (R1)	700
Autoincrement	400	AC \leftarrow (R1)+	700
Autodecrement	399	AC \leftarrow -(R1)	450

Numerical Example



Problems

- A two-word instruction is stored in memory at an address designated by the symbol W . The address field of the instruction (stored at $W + 1$) is designated by the symbol Y . The operand used during the execution of the instruction is stored at an address symbolized by Z . An index register contains the value X . State how Z is calculated from the other addresses if the addressing mode of the instruction is
 - Direct
 - Indirect
 - Relative
 - Indexed
- A relative mode branch type of instruction is stored in memory at an address equivalent to decimal 750. The branch is made to an address equivalent to decimal 500. What should be the value of the relative address field of the instruction (in decimal)?

- How many times does the control unit refer to memory when it fetches and executes an indirect addressing mode instruction if the instruction is (a) a computational type requiring an operand from memory; (b) a branch type.
- What must the address field of an indexed addressing mode instruction be to make it the same as a register indirect mode instruction?
- An instruction is stored at location 300 with its address field at location 301. The address field has the value 400. A processor register R1 contains the number 200. Evaluate the effective address if the addressing mode of the instruction is (a) direct; (b) immediate (c) relative (d) register indirect; (e) index with R1 as the index register.

- Assume that in a certain byte-addressed machine all instructions are 32 bits long. Assume the following state of affairs for the machine: Fill in the following table:

Address	Value
PC	100
R0	200
R1	300
100	200
104	300
108	400
200	500
300	600
500	700

Instruction	Addressing mode	Value in R0
Load r0, #200	Immediate	
Load r0, 200	Direct	
Load r0, (200)	Indirect	
Load r0,r1	Register	
Load r0, [r1]	Register Indirect	
Load r0, -100[r1]	Based	
Load r0, 200[PC]	Relative	

- Given the following memory values and a one-address machine with an accumulator, what values do the following instructions load into the accumulator?
 - Word 20 contains 40
 - Word 30 contains 50
 - Word 40 contains 60
 - Word 50 contains 70
 - Load immediate 20
 - Load direct 20
 - Load indirect 20
 - Load immediate 30
 - Load direct 30
 - Load indirect 30
- Let the address stored in the program counter be designated by the symbol X1. The instruction stored in X1 has the address part (operand reference) X2. The operand needed to execute the instruction is stored in the memory word with address X3. An index register contains the value X4. What is the relationship between these various quantities if the addressing mode of the instruction is (a) direct (b) indirect (c) PC relative (d) indexed?

- An address field in an instruction contains decimal value 14. where is the corresponding operand located for:
 - Immediate addressing?
 - Direct addressing?
 - Indirect addressing?
 - Register addressing?
 - Register indirect addressing?
- A PC-relative mode branch instruction is stored in memory at address 620_{10} . The branch is made to location 530_{10} . The address field in the instruction is 10 bits long. What is the binary value in the instruction?