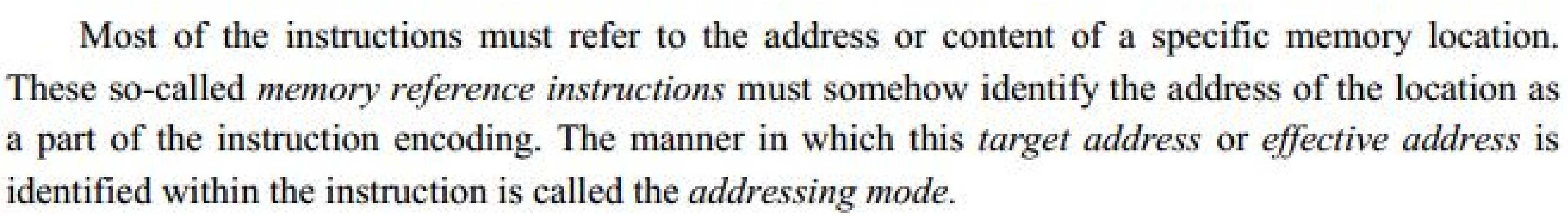
Addressing Modes

Introduction



Most of the instructions must refer to the address or content of a specific memory location. These so-called *memory reference instructions* must somehow identify the address of the location as a part of the instruction encoding. The manner in which this *target address* or *effective address* is identified within the instruction is called the *addressing mode*.

Addressing Modes

- Implied
- Immediate
- Direct
- Indirect
- Register
- Register Indirect
- Displacement (Indexed)
- Autoincrement
- Autodecrement
- Stack

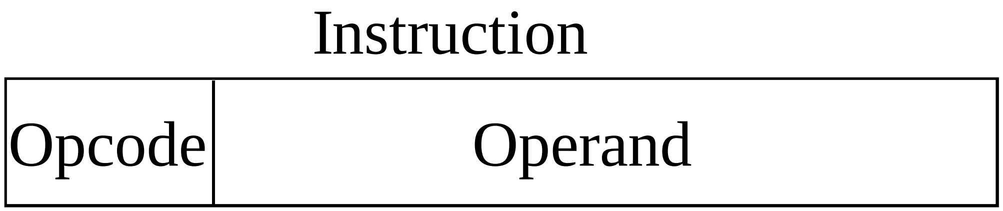
Implied Mode

- Operands are specified implicitly in definition of the instruction
- ***Examples***
  - **COM** : Complement Accumulator
    - Operand in AC is implied in the definition of the instruction.
    - All register reference instruction that use an accumulator are implied mode instruction.
  - **PUSH** : Stack push
    - Operand is implied to be on top of the stack.
    - Zero address instruction in stack are implied mode since the operands are implied on top of stack.

Immediate Addressing

- Operand is part of instruction
- Operand = address field
- e.g. ADD 5
  - Add 5 to contents of accumulator
  - 5 is operand
- No memory reference to fetch data
- Fast
  - Useful for initializing registers to a constant value
  - ***Example*** **: LD #NBR**

Immediate Addressing Diagram



Instruction

Opcode

Operand

Direct Addressing

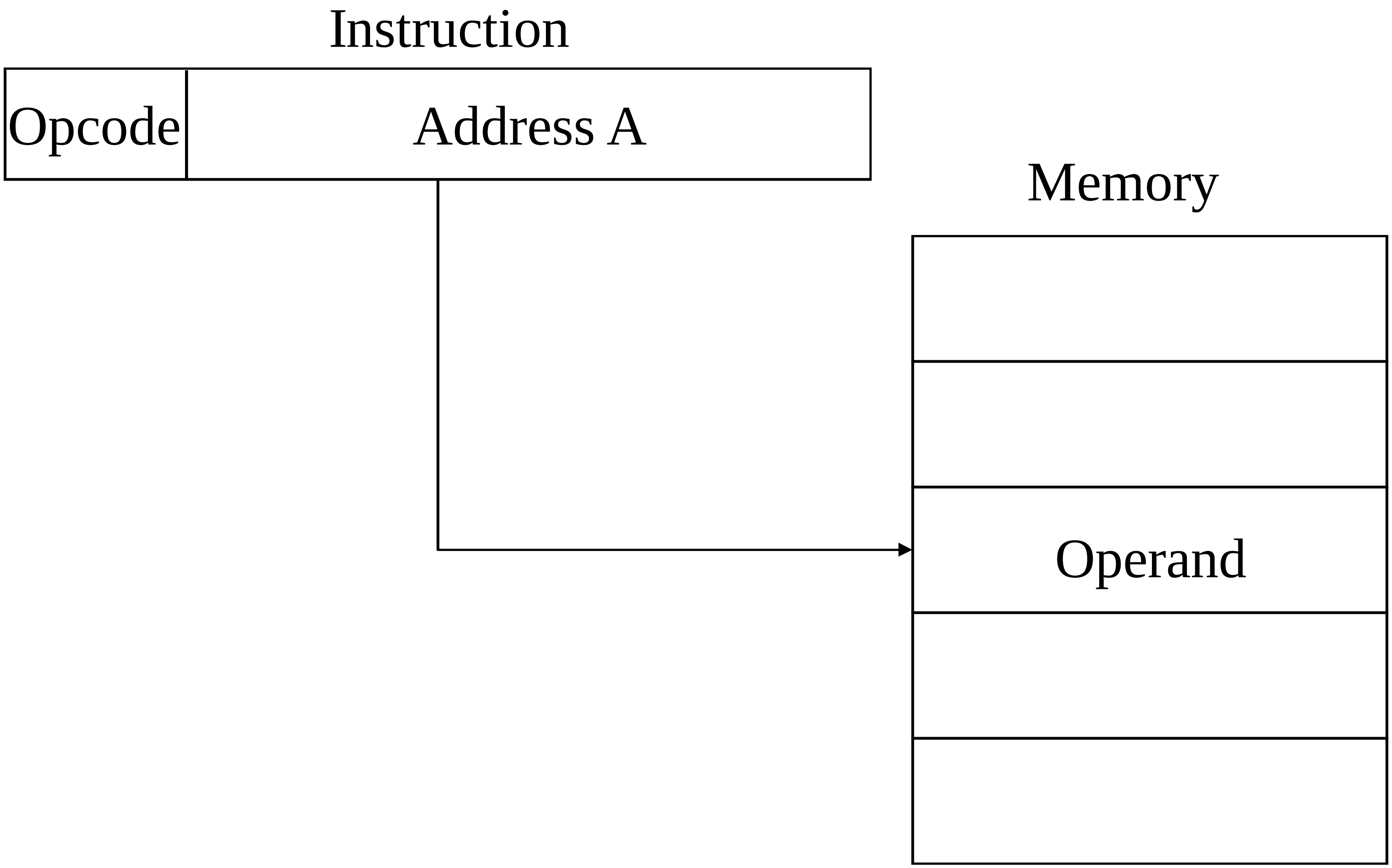
- Address field contains address of operand
- Effective address (EA) = address field (A)
- e.g. ADD A
    - Add contents of cell A to accumulator
    - Look in memory at address A for operand
- Single memory reference to access data
- No additional calculations to work out effective address

- ***Example*** : **LD ADR**

$AC \leftarrow M[ADR]$

ADR = Address part of Instruction

Direct Addressing Diagram



Indirect Addressing (1)

- Memory cell pointed to by address field contains the address of (pointer to) the operand
- EA = (A)
  - Look in A, find address (A) and look there for operand
- e.g. ADD (A)
  - Add contents of cell pointed to by contents of A to accumulator

Indirect Addressing (2)

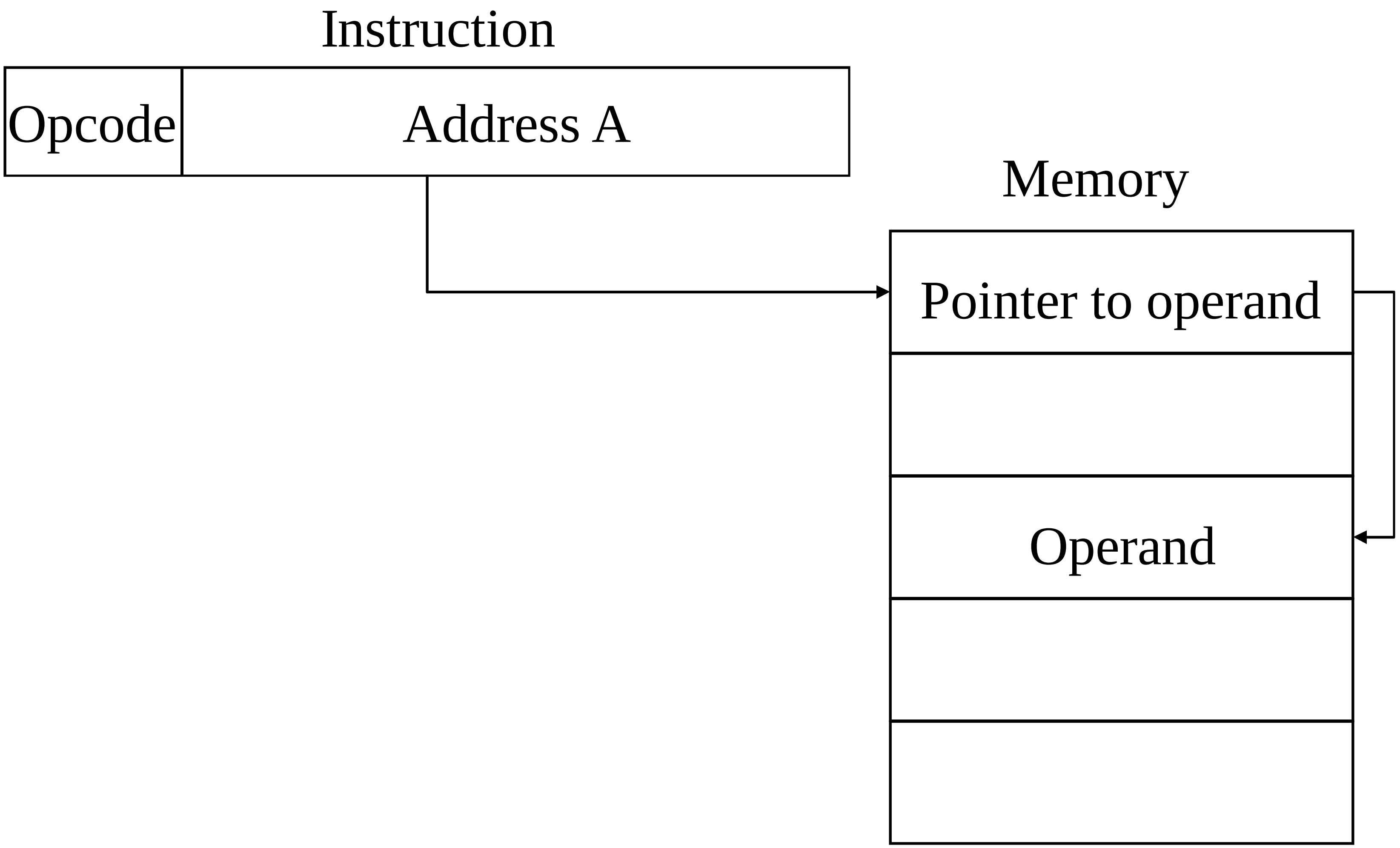
- Large address space
- 2^n where n = word length
- May be nested, multilevel, cascaded
  - e.g. EA = (((A)))

- Multiple memory accesses to find operand
- Hence slower

Example : **LD @ADR**

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

Indirect Addressing Diagram


Instruction

Opcode

Address A

Memory

Pointer to operand

Operand

Register Addressing (1)

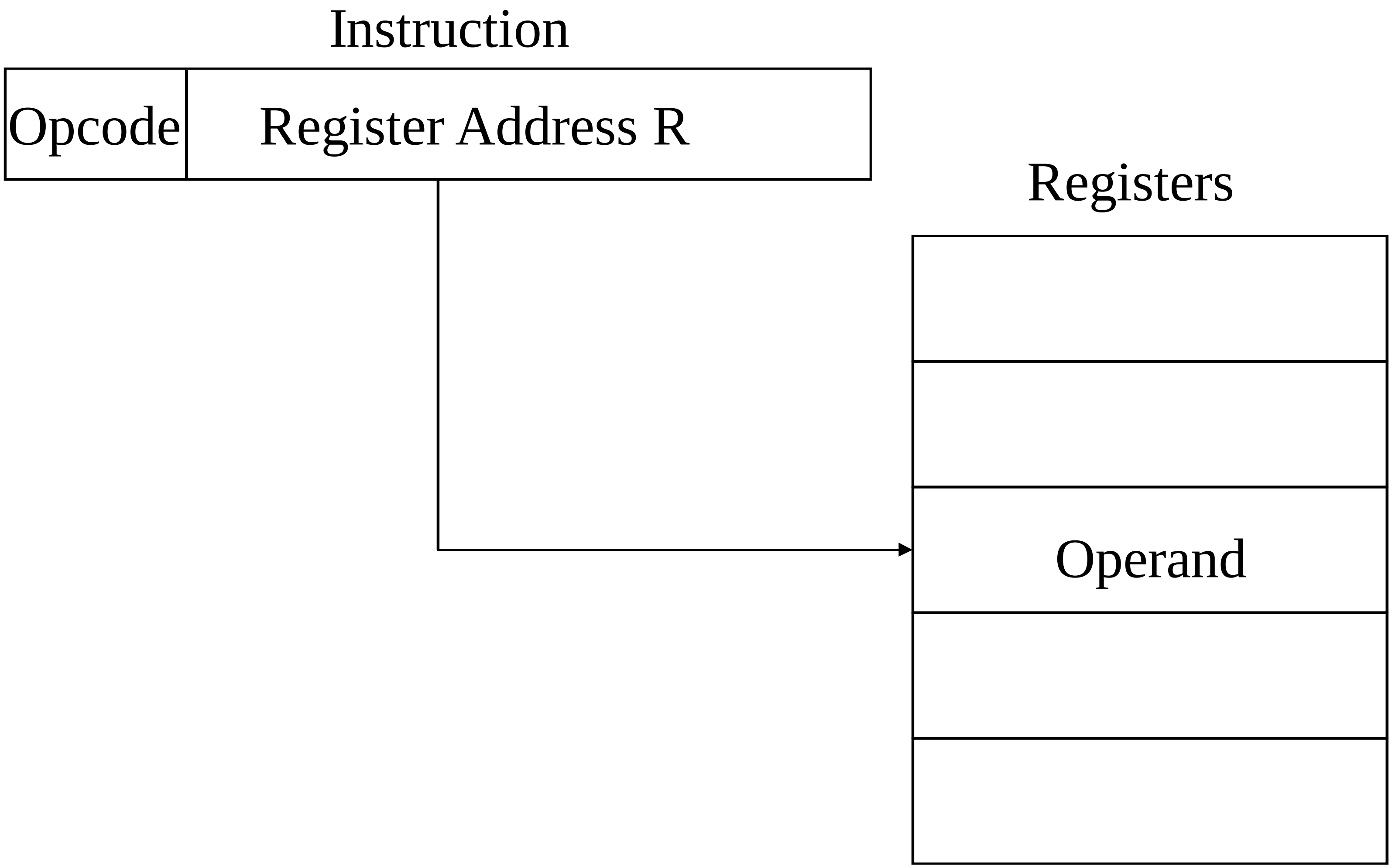
- Operand is held in register named in address filed
    - Register is selected from a register field in the instruction
        - k-bit register field can specify any one of 2^k registers
- EA = R
- Limited number of registers
- Very small address field needed
    - Shorter instructions
    - Faster instruction fetch

Register Addressing (2)

- No memory access
- Very fast execution
- Very limited address space
- Multiple registers helps performance
  - Requires good assembly programming or compiler writing
  - Ex. C programming
    - register int a;

Example : **LD R1** $AC \leftarrow R1$

Register Addressing Diagram

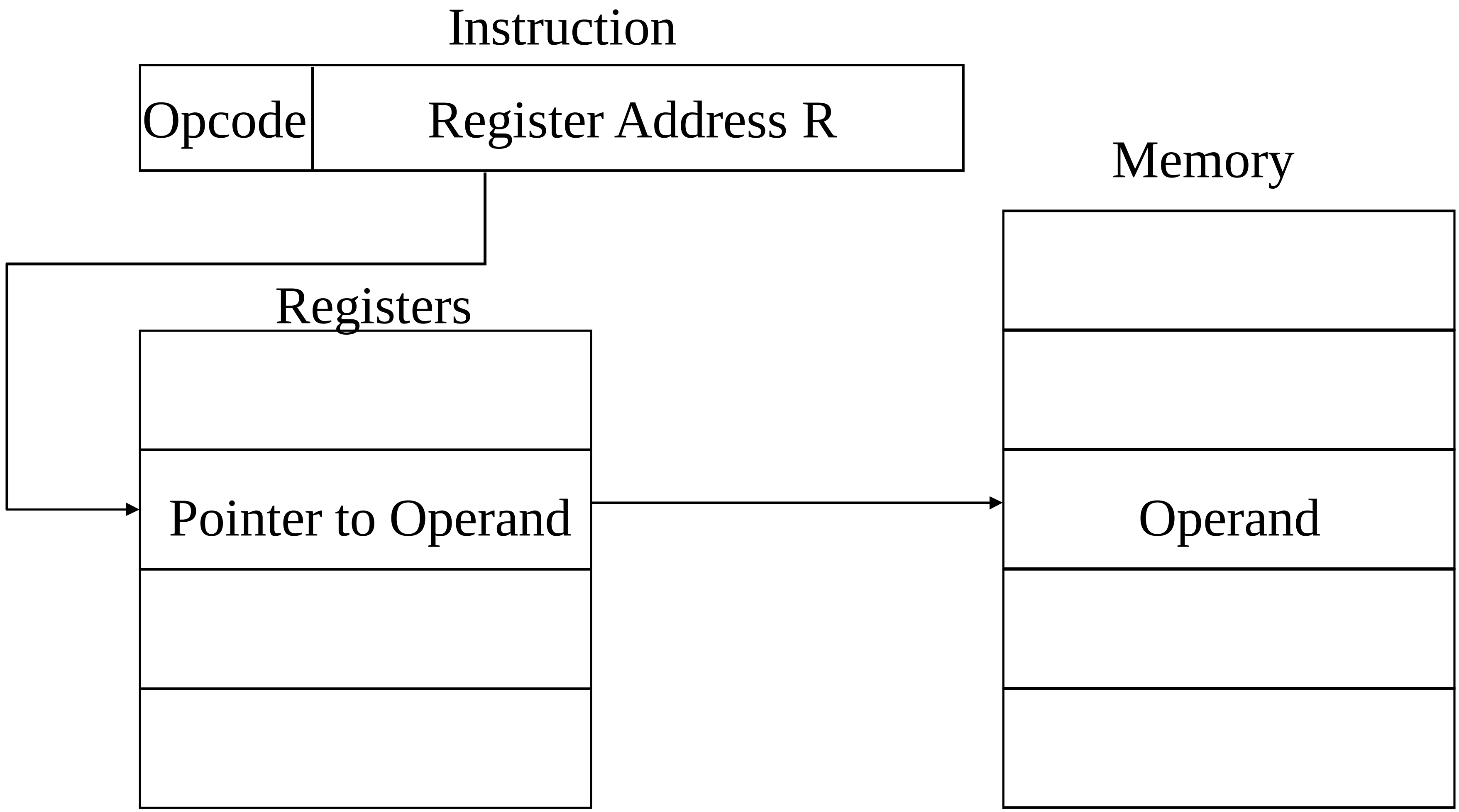


Register Indirect Addressing

- C.f. indirect addressing
- EA = (R)
- Operand is in memory cell pointed to by contents of register R
- Large address space (2^n)
- One fewer memory access than indirect addressing

●*Example* : **LD (R1)** $AC \leftarrow M[R1]$

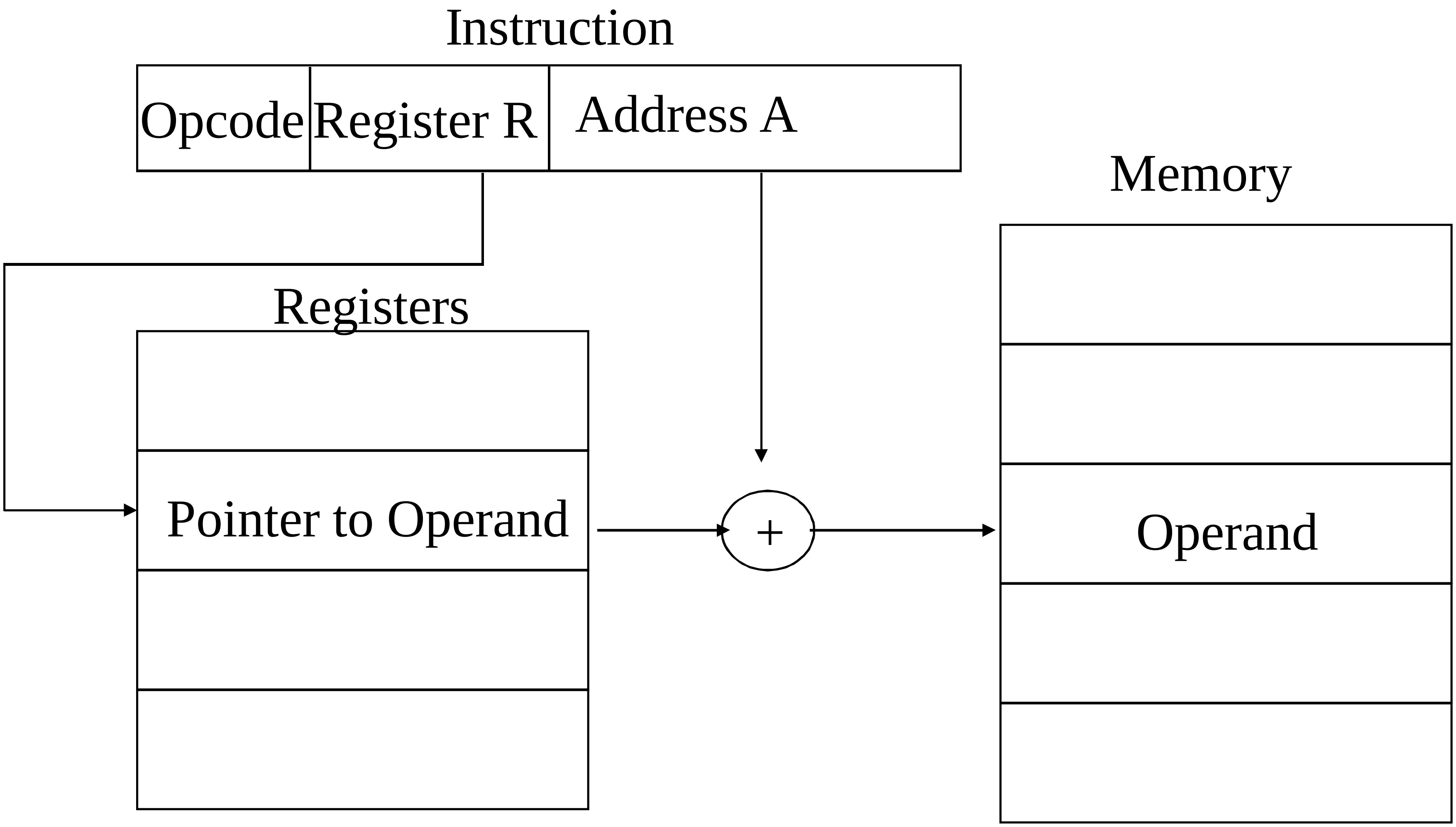
Register Indirect Addressing Diagram



Displacement Addressing

- EA = A + (R)
- Address field hold two values
  - A = base value
  - R = register that holds displacement
  - or vice versa

Displacement Addressing Diagram



Relative Addressing

- A version of displacement addressing
  - PC is added to the address part of the instruction to obtain the effective address
- R = Program counter, PC
- EA = A + (PC)
- i.e. get operand from A cells from current location pointed to by PC
- c.f locality of reference & cache usage

- ***Example*** : **LD $ADR** $AC \leftarrow M[PC + ADR]$

Indexed Addressing

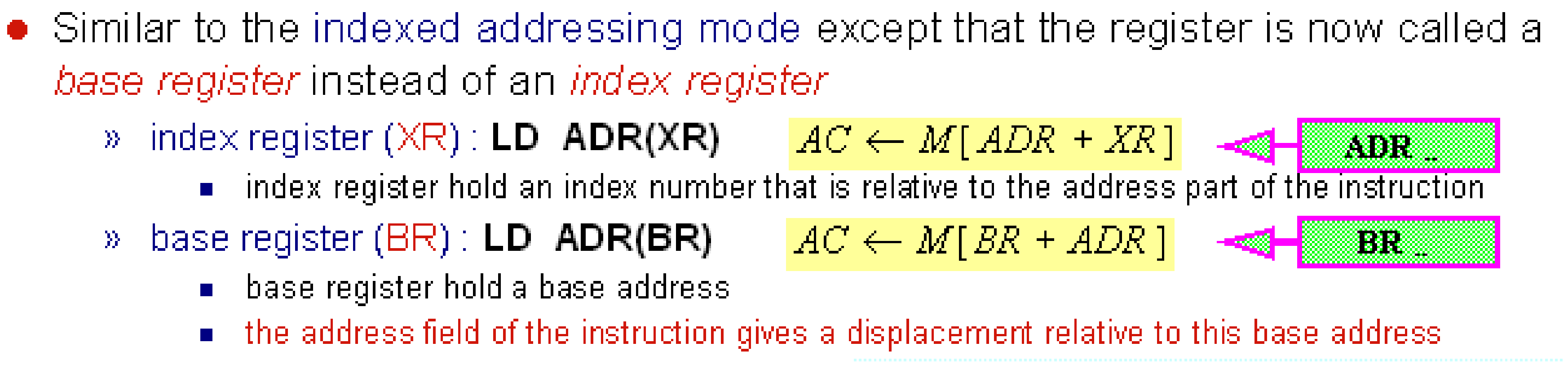
- XR (*Index register*) is added to the address part of the instruction to obtain the effective address
- ***Example*** : **LD ADR(XR)**

- A = base
- R = displacement
- EA = A + (R)

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

Base-Register Addressing

- the content of a base register is added to the address part of the instruction to obtain the effective address



- Similar to the indexed addressing mode except that the register is now called a *base register* instead of an *index register*
  - » index register (XR) : **LD ADR(XR)** $AC \leftarrow M[ADR + XR]$ ADR
    - index register hold an index number that is relative to the address part of the instruction
  - » base register (BR) : **LD ADR(BR)** $AC \leftarrow M[BR + ADR]$ BR
    - base register hold a base address
    - the address field of the instruction gives a displacement relative to this base address

- 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*) : **LD (R1)+**

$$AC \leftarrow M[R1],\ R1 \leftarrow R1 + 1$$

Stack Addressing

- Operand is (implicitly) on top of stack
- e.g.
  - ADD Pop top two items from stack and add

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

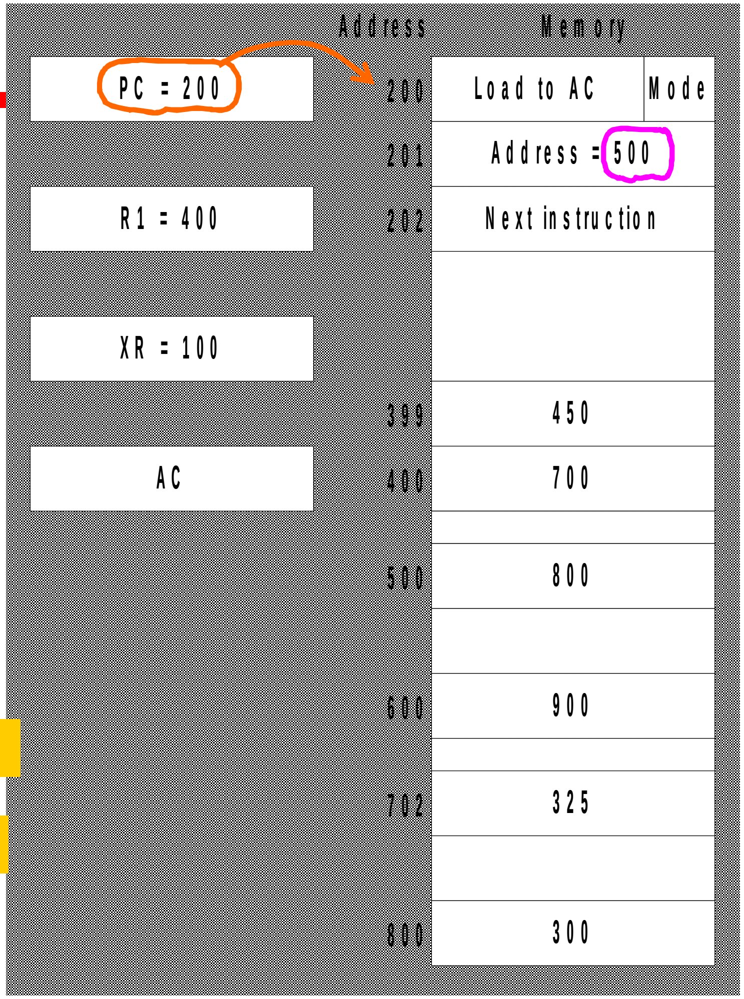
R1 = 400

500 + 202 (PC)

500 + 100 (XR)

R1 = 400 (after)

R1 = 400 -1 (prior)



Application of Addressing Modes

The 8085 has the following 5 different types of addressing.

1. **Immediate Addressing**
2. **Direct Addressing**
3. **Register Addressing**
4. **Register Indirect Addressing**
5. **Implied Addressing**

8086 Addressing Modes

Register addressing
Immediate addressing
Direct addressing
Register indirect addressing
Base-plus-index addressing
Register relative addressing
Base relative-plus-index addressing: