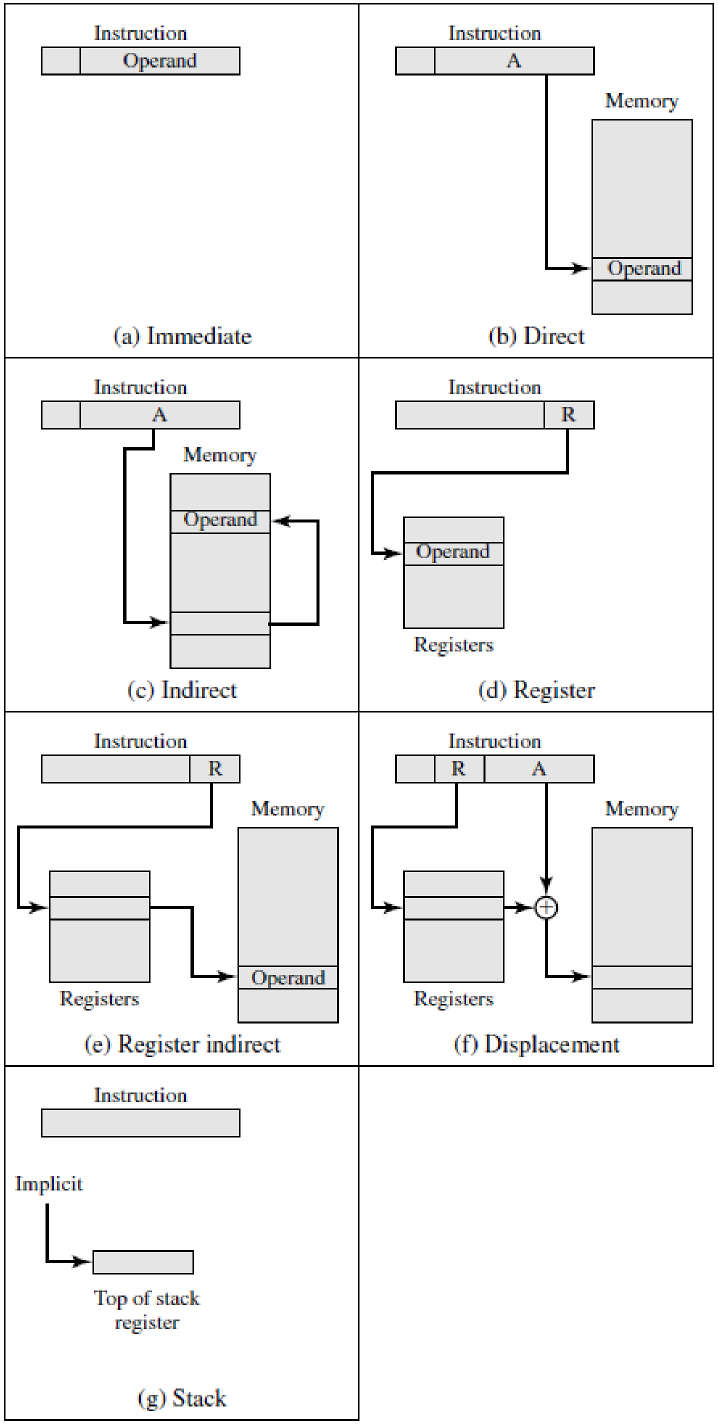
Addressing Modes

- The way of specifying data to be operated by an instruction is known as **addressing modes.** In the other word addressing modes refer to the different methods of addressing the operands. Depending upon the data types used in the instruction and the memory addressing modes, any instruction may belong to one or more addressing modes.
- **Instructions** are operations performed by the CPU. An instruction is a statement that is executed at runtime.
- Label (optional)
- Instruction (required)
- Operands (instruction specific)
- Comment (optional)

[Label:] Instruction Operands [; Comment]

Addressing Modes

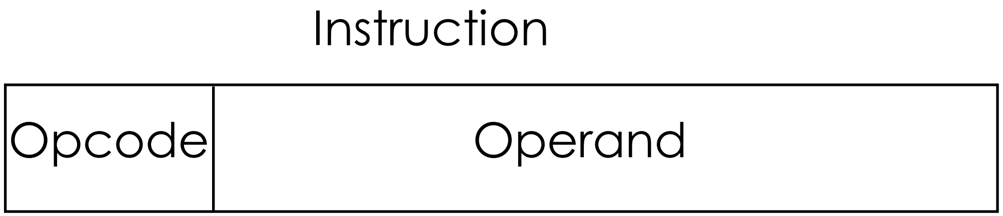
- Immediate
- Direct
- Indirect
- Register
- Register Indirect
- Displacement (Indexed)
- Stack



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
- Limited range

Immediate Addressing Diagram



Instruction

Opcode

Operand

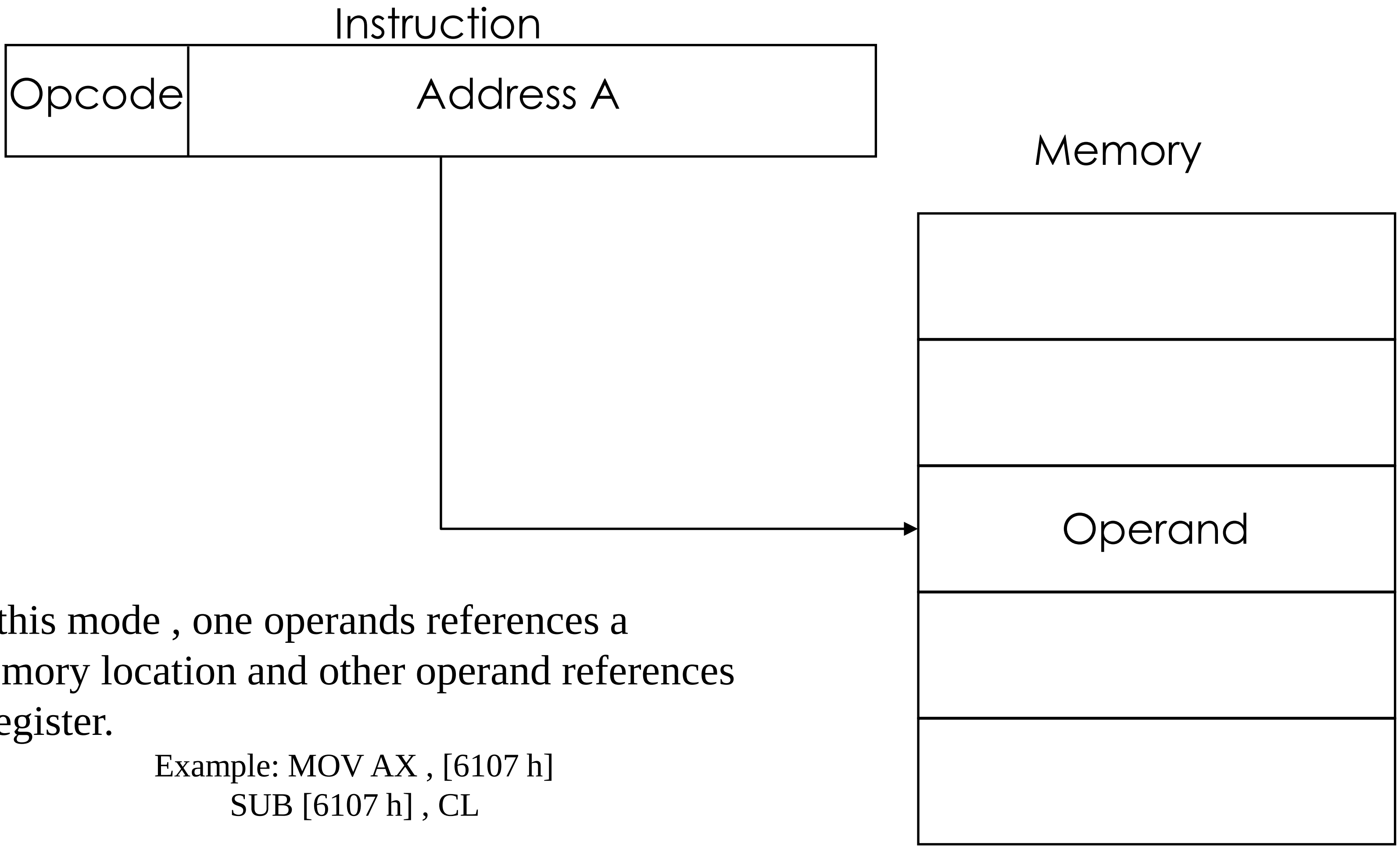
The destination field in the first operand defines the length of the data and may be a register or a memory location (the first operand is never an immediate value), and the second operand is the data itself (appears in the form of successive byte or bytes).

Example: MOV A , 8705 h

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
- Limited address space

Direct Addressing Diagram



In this mode , one operands references a memory location and other operand references a register.

Example: MOV AX , [6107 h]
SUB [6107 h] , CL

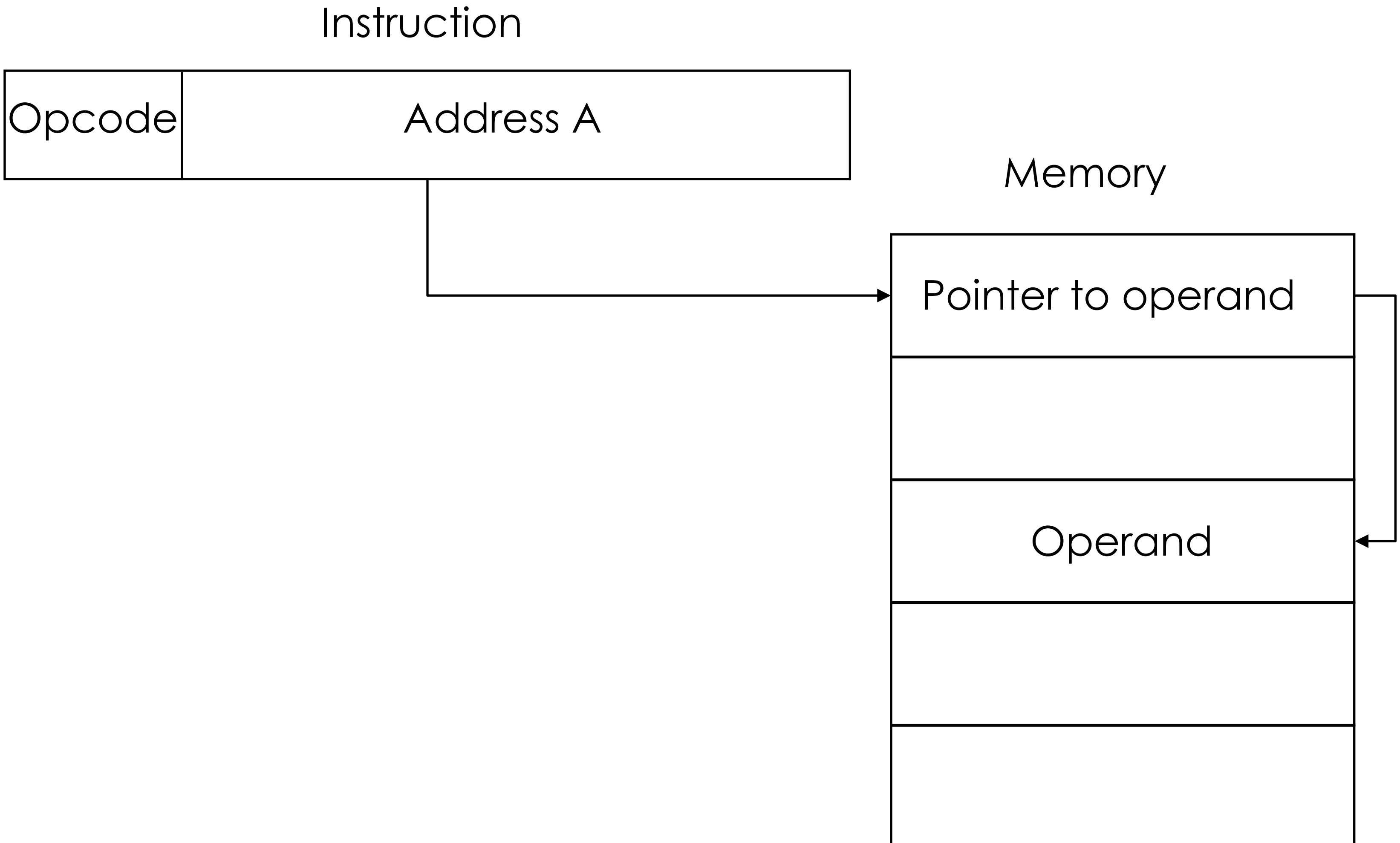
Indirect Addressing (1)

- 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)))
        - Draw the diagram yourself
- Multiple memory accesses to find operand
- Hence slower

Indirect Addressing Diagram



Register Addressing (1)

- Operand is held in register named in address filed
- EA = R
- Limited number of registers
- Very small address field needed
  - Shorter instructions
  - Faster instruction fetch

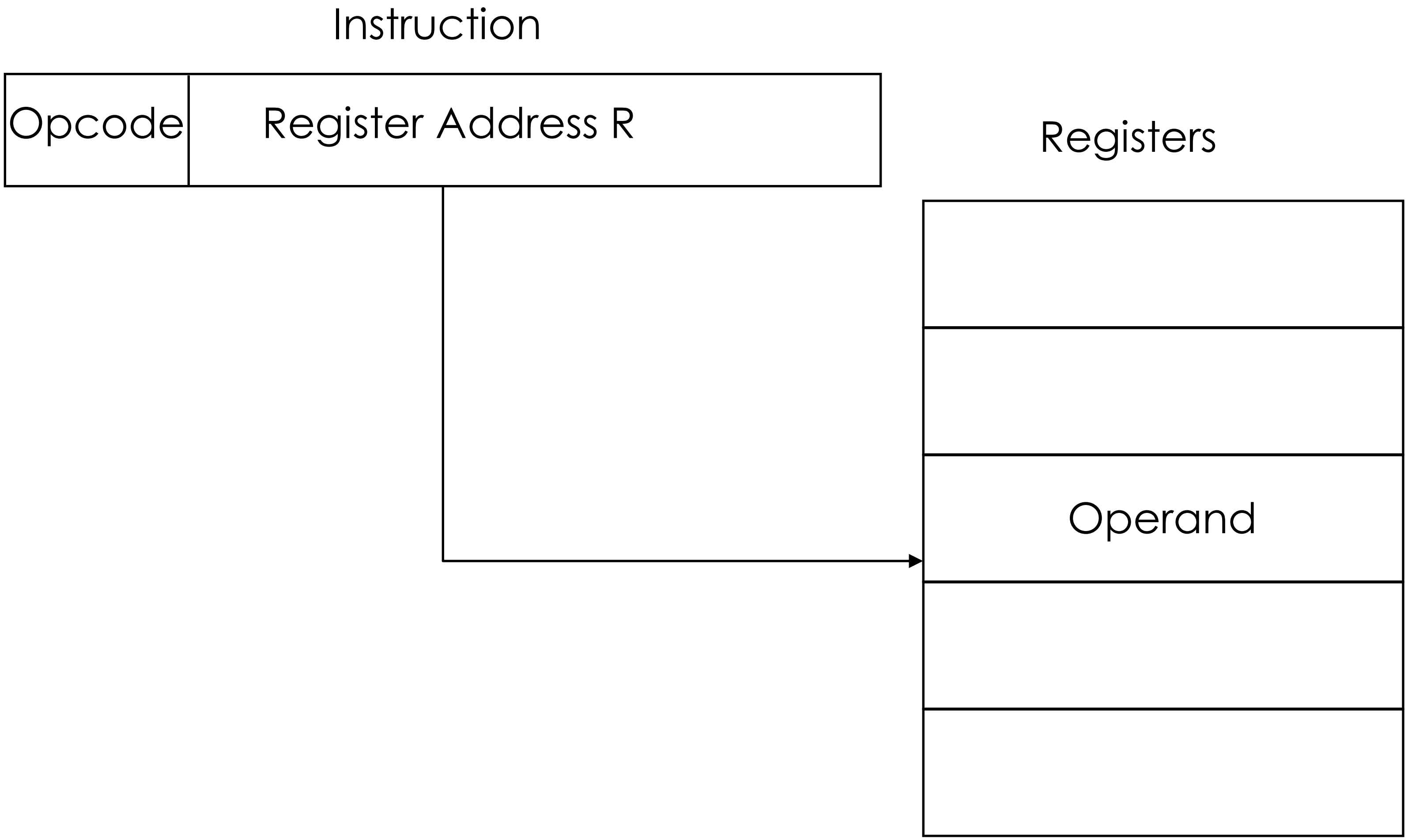
In the register addressing mode, the data is stored in a register and it is referred using the particular register (the operands are registers). All the registers, except IP, may be used in this mode.

Example: ADD BL , AL
MOV SI , CX

Register Addressing (2)

- No memory access
- Very fast execution
- Very limited address space
- Multiple registers helps performance
  - Requires good assembly programming or compiler writing
  - N.B. C programming
    - register int a;
- c.f. Direct addressing

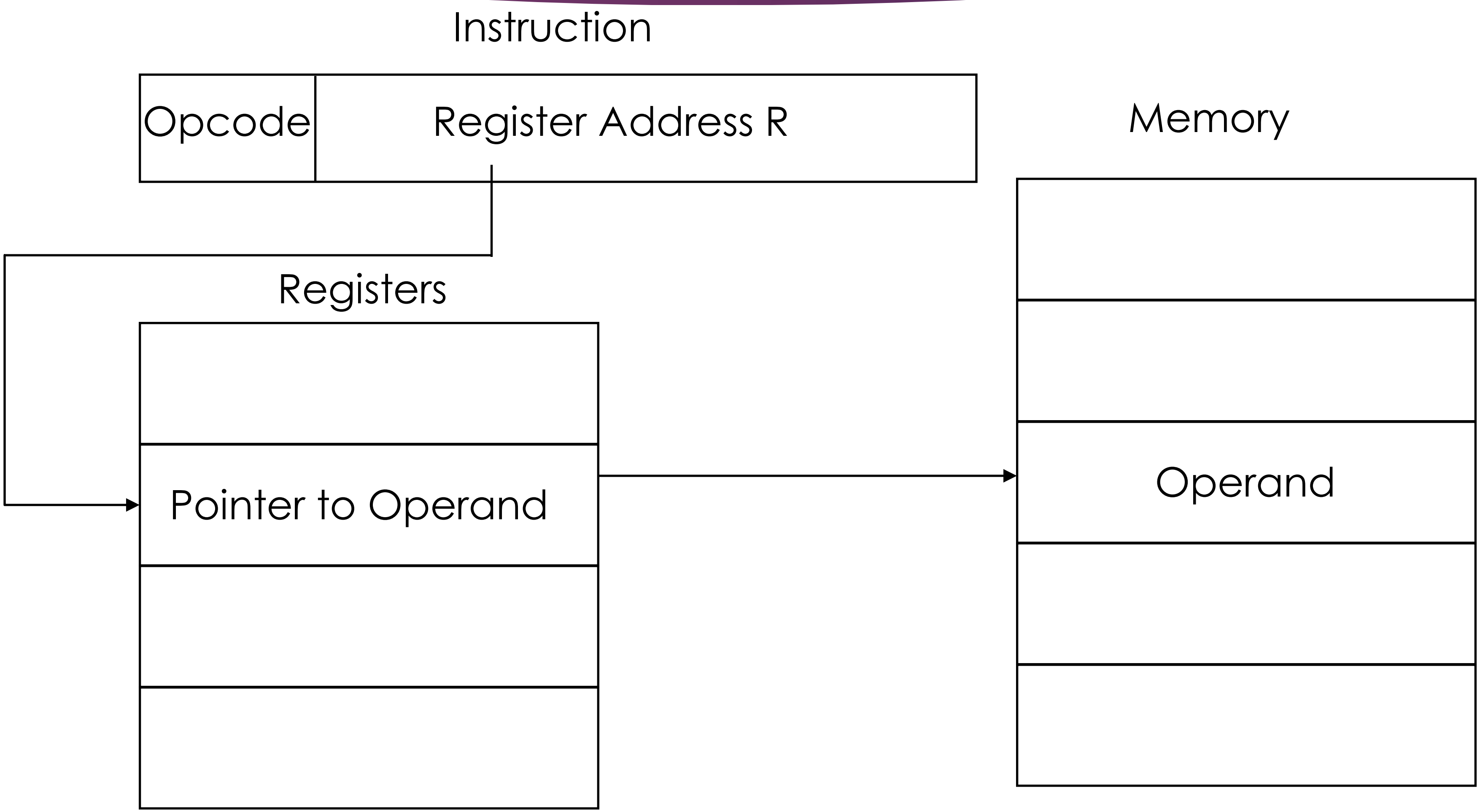
Register Addressing Diagram



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

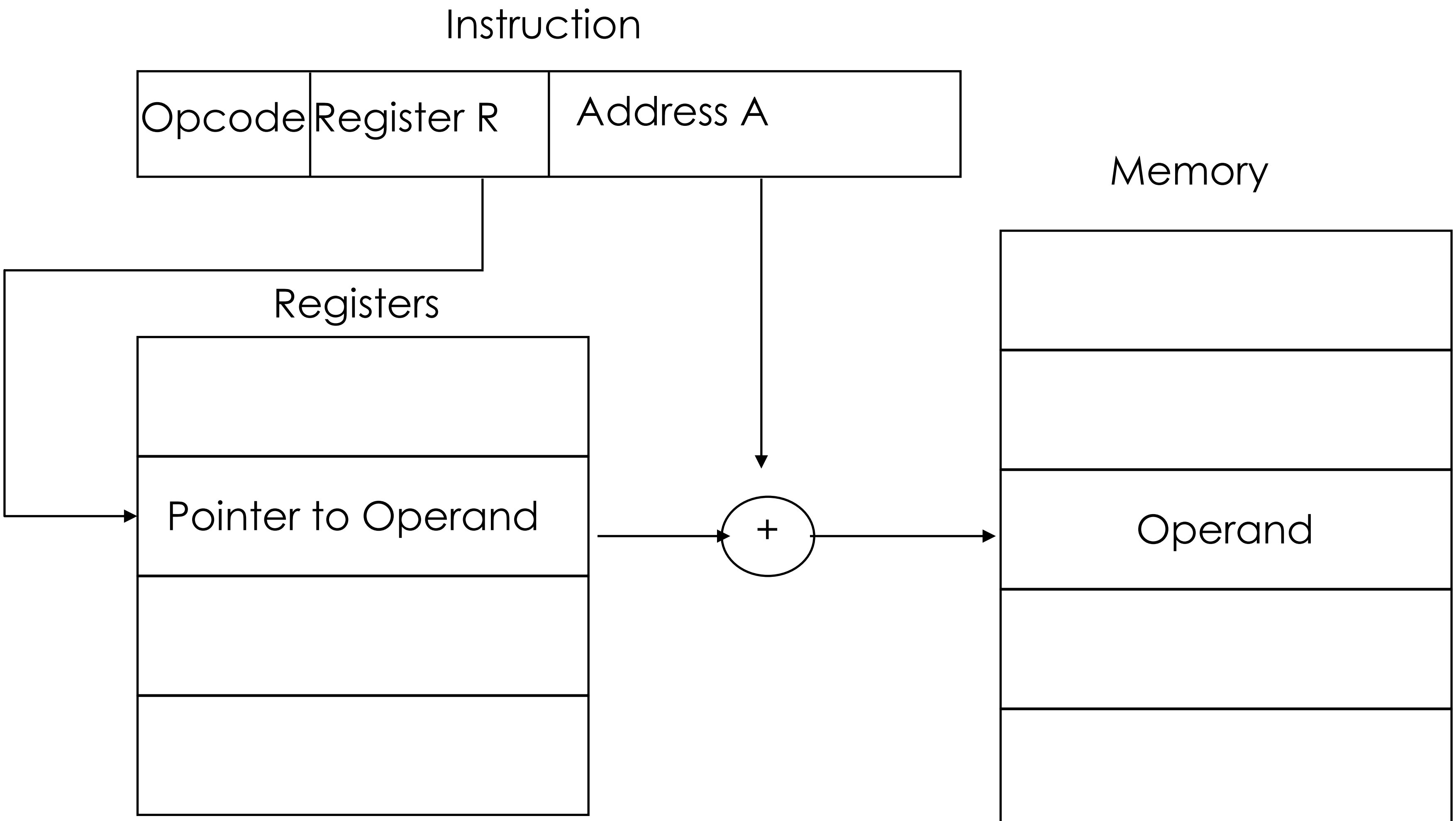
Register Indirect Addressing Diagram



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
- 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

Base-Register Addressing

- A holds displacement
- R holds pointer to base address
- R may be explicit or implicit
- e.g. segment registers in 80x86

Indexed Addressing

- A = base
- R = displacement
- EA = A + R
- Good for accessing arrays
  - EA = A + R
  - R++

Combinations

- Postindex
- EA = (A) + (R)

- Preindex
- EA = (A+(R))

- (Draw the diagrams)

Stack Addressing

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