CS 223 Computer Architecture and Organization

Instruction Sets: Addressing Modes and Formats

J. K. Deka

Professor

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

Instructions

- Instruction Set
- Format of Instructions
  - Single Operand
  - Two Operands
  - Three Operands

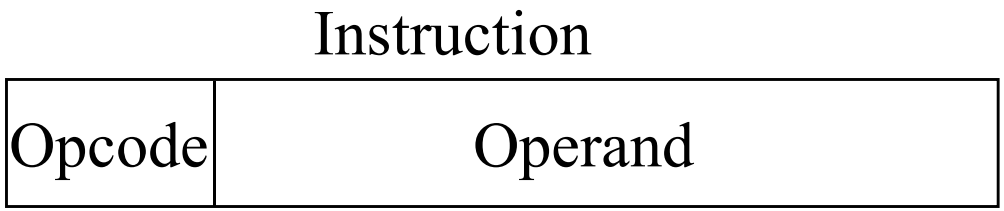
Addressing Modes

- 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

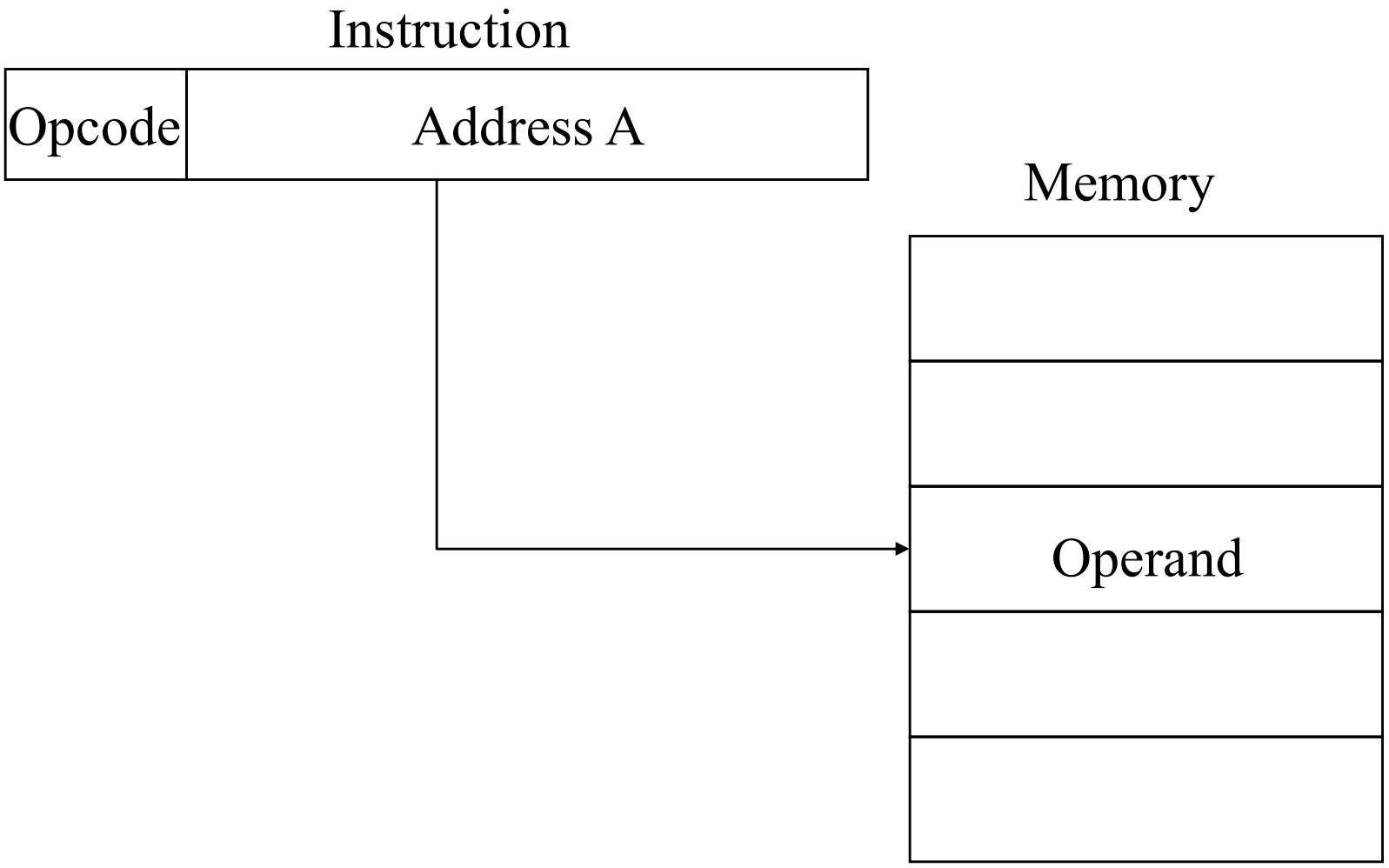
Immediate Addressing Diagram



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



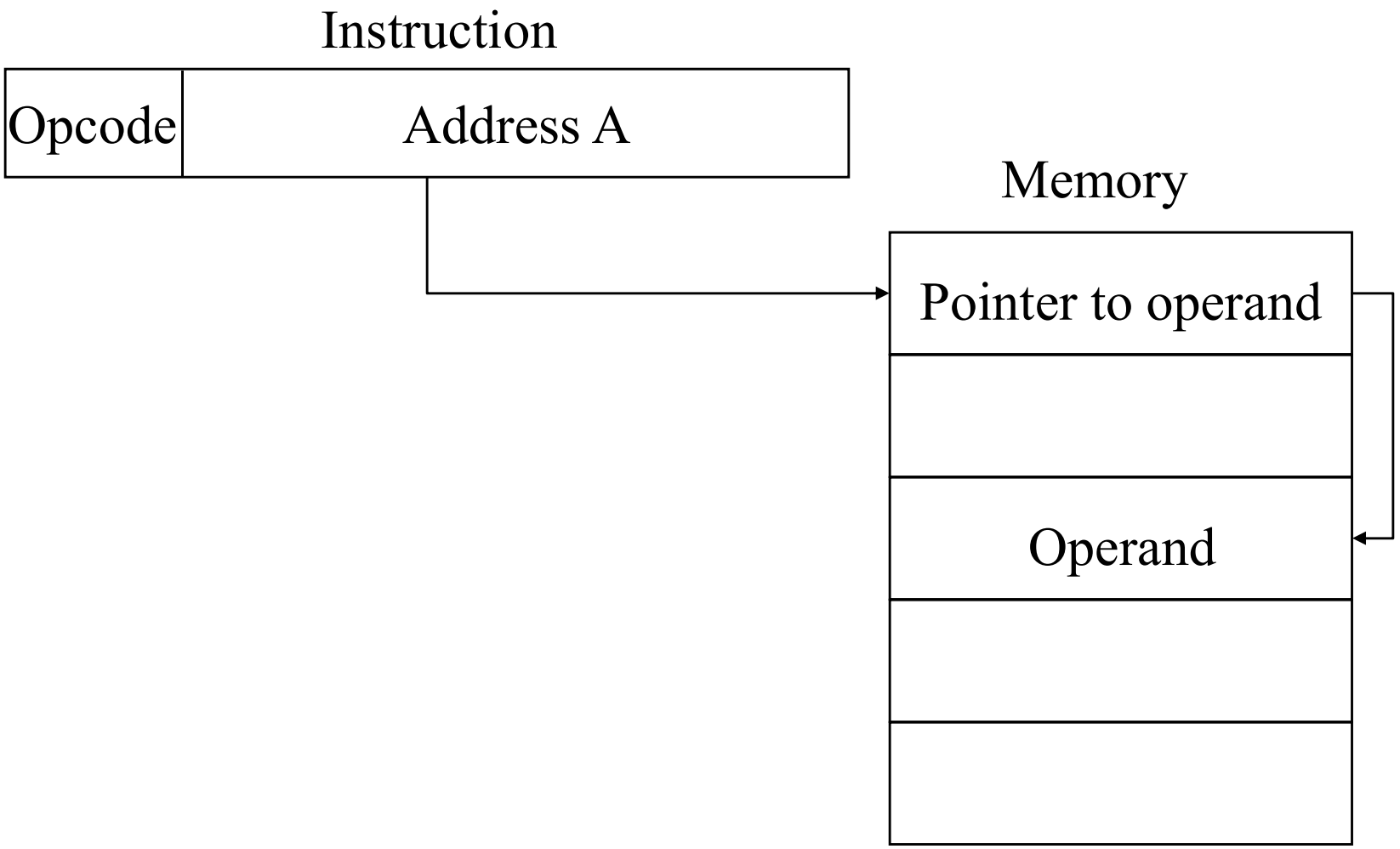
Indirect Addressing

- 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

- Large address space
- May be nested, multilevel, cascaded
  - e.g. EA = (((A)))
- Multiple memory accesses to find operand
- Hence slower

Indirect Addressing Diagram



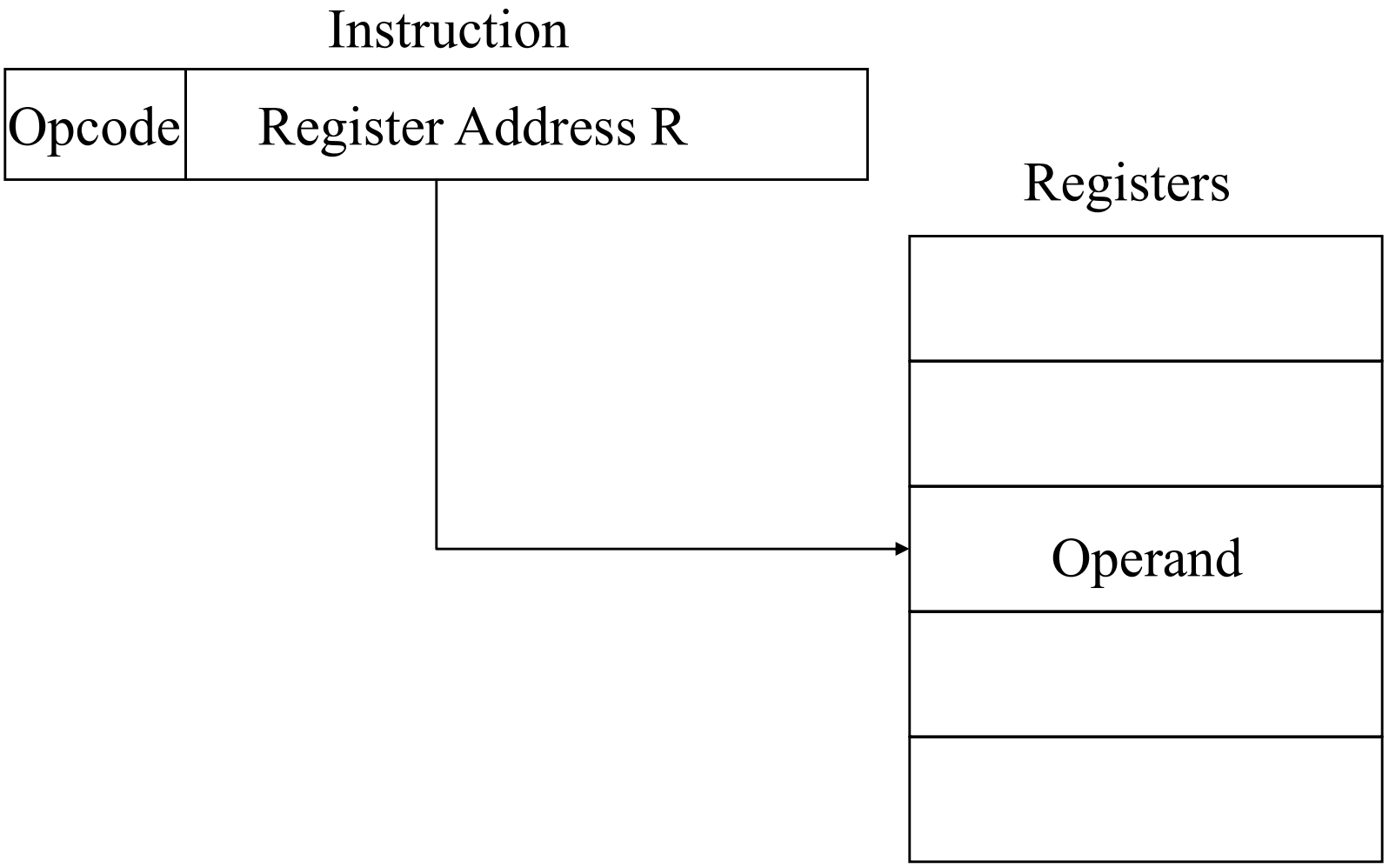
Register Addressing

- 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

Register Addressing

- No memory access
- Very fast execution
- Very limited address space
- c.f. Direct addressing

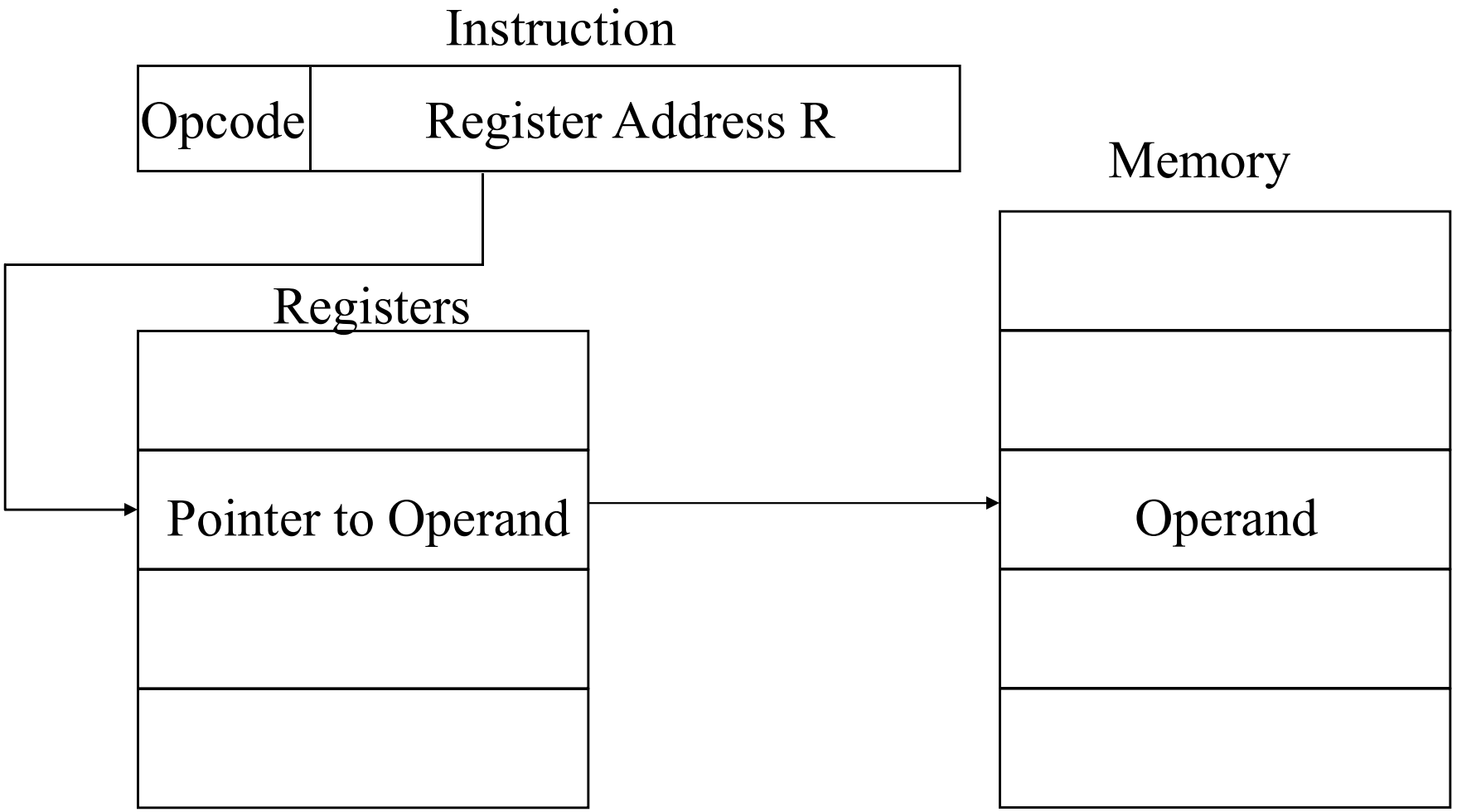
Register Addressing Diagram



Register Indirect Addressing

- Indirect addressing
- EA = (R)
- Operand is in memory cell pointed to by contents of register R
- One fewer memory access than indirect addressing

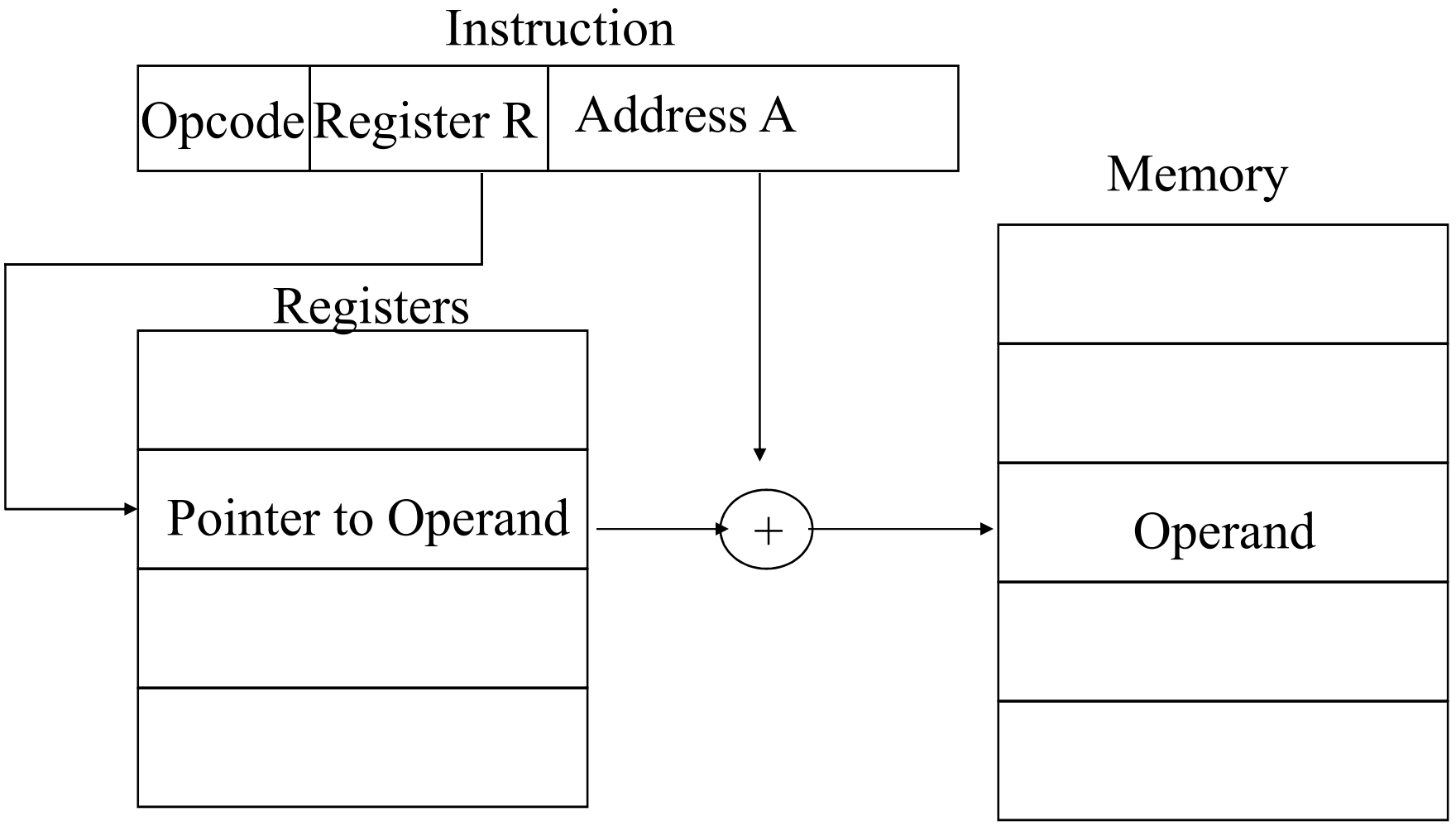
Register Indirect Addressing Diagram



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++

Stack Addressing

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

Instruction Formats

- Layout of bits in an instruction
- Includes opcode
- Includes (implicit or explicit) operand(s)
- Usually more than one instruction format in an instruction set

Instruction Length

- Affected by and affects:
  - Memory size
  - Memory organization
  - Bus structure

Allocation of Bits

- Number of addressing modes
- Number of operands
- Register versus memory
- Number of register sets

Reference

Computer Organization and Architecture –
Designing for Performance
William Stallings

Chapter 11: Page no. 386 – 398 (Seventh Edition)
Page No.: 401 – 408 (Eighth Edition)
413 – 421 (Eighth Edition)

- ADD R1, (R2) R1 <- R1 + (R2)
  - Add the content of a memory location whose address is in R2 to the content of register R1 and store the result in register R1

Fetch Phase:

t1: MAR <- PC, Read

t2: MBR <- Memory

PC <- PC + 1

t3: IR <- MBR

- ADD R1, (R2) R1 <- R1 + (R2)
  - Add the content of a memory location whose address is in R2 to the content of register R1 and store the result in register R1

  Execute Phase:

- ADD R1, M R1 <- R1 + M
  - Add the content of a memory location whose address is a part of the instruction to the content of register R1 and store the result in register R1

Fetch:

t1: MAR <- PC, Read

t2: MBR <- Memory

PC <- PC + 1

t3: IR <- MBR

- ADD R1, M R1 <- R1 + M
  - Add the content of a memory location whose address is a part of the instruction to the content of register R1 and store the result in register R1

  Execute Phase:

- ADD R1, M R1 <- R1 + M
  - Add the content of a memory location whose address is in the second word of the instruction to the content of register R1 and store the result in register R1

Fetch:

t1: MAR <- PC, Read

t2: MBR <- Memory

PC <- PC + 1

t3: IR <- MBR

- ADD R1, M R1 <- R1 + M
  - Add the content of a memory location whose address is in the second word of the instruction to the content of register R1 and store the result in register R1

  Execute Phase