Computer Organization

Instruction Set Characteristics, Instruction Formats, Addressing Modes, RTL & Micro-Operations, CISC, RISC.

Chapters (10 + 11 + Mano Ch.4 + 13)

Infix & Postfix Representations

- Infix notation
  - c = a + b
  - c = a - b
- Postfix notation
  - a b +
  - a b -

$a + (b \times c)$ becomes a b c $\times +$

$(a + b) \times c$ becomes a b $+$ c $\times$

Machine Instructions

- What is an Instruction?

- What are the instruction components?

Instruction Operands

- What is an Operand?
- Where are the Operands found?
- How the CPU Find the Operands?

Typical Instructions

Data Movement	Load (from memory) Store (to memory) memory-to-memory move register-to-register move input (from I/O device) output (to I/O device) push, pop (to/from stack)
Arithmetic	Data Types: (signed & unsigned) Integer (binary + decimal) (signed & unsigned) Floating Point Numbers Operations: Add, Subtract, Multiply, Divide
Logical	Not, and, or, set, clear
Shift	Arithmetic (& Logical) shift (left/right), rotate (left/right)
Control (Jump/Branch)	unconditional, conditional
Subroutine Linkage	call, return
Interrupt	trap, return
Synchronisation	test & set (atomic r-m-w)
String	search, compare, translate

Instruction Types

- Load, Store, Move, Input, Output
- Add, Sub, Mul, Div
- NOT, AND, OR, Set, Clear
- Shift Left (Logical/Arithmetic), Shift Right (Logical/Arithmetic).
- Jump, Branch
- Call, Return

Number of Addresses (Operands)

- 0 – Addresses (Operands)
- 1 – Address (Operands)
- 2 – Addresses (Operands)
- 3 – Addresses (Operands)

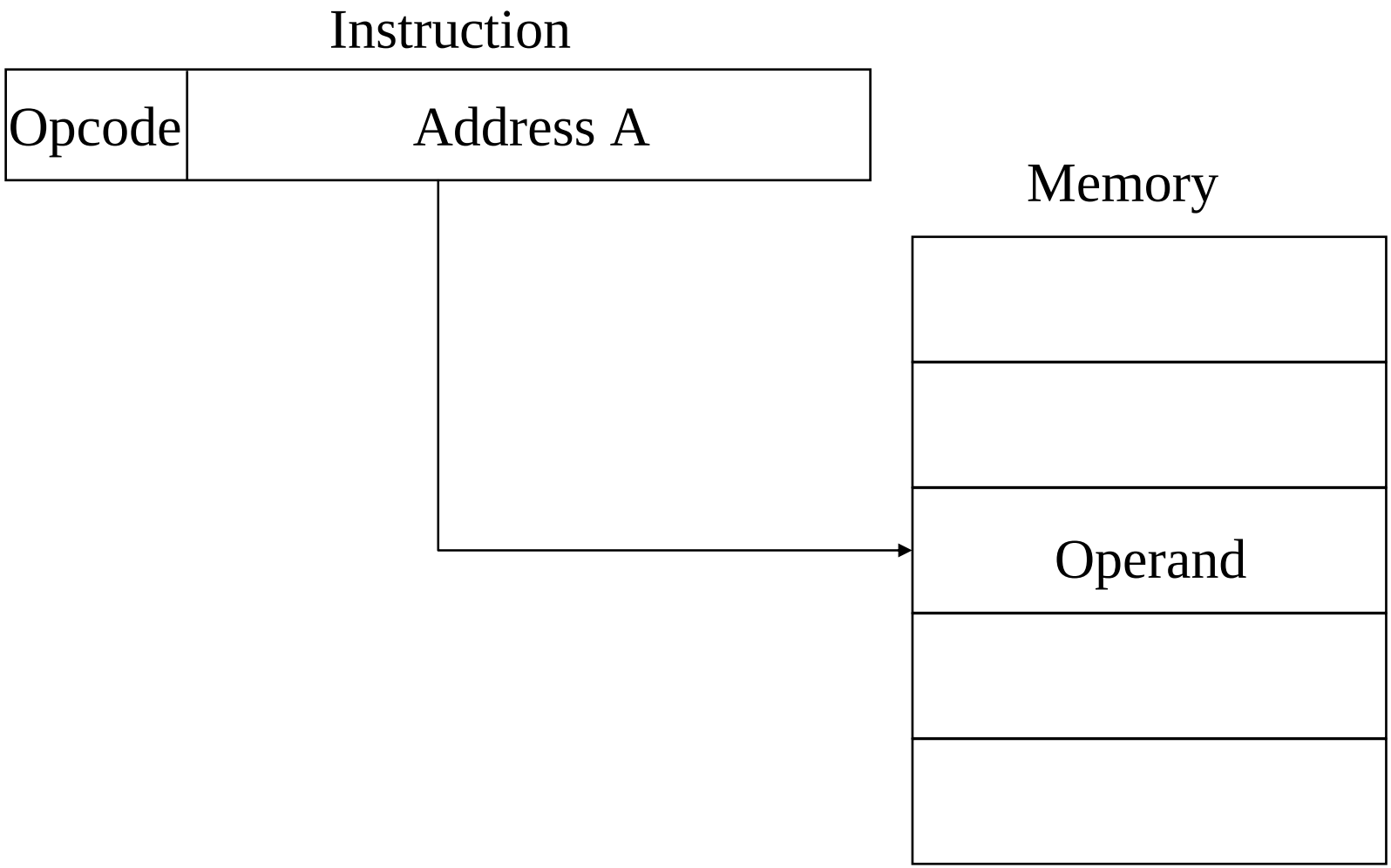
How many Operand (Address) an Instruction Need? (Maximum)

- Load, Store, Move, Input, Output
- Add, Sub, Mul, Div
- NOT, AND, OR, Set, Clear
- Shift Left (Logical/Arithmetic), Shift Right (Logical/Arithmetic).
- Jump, Branch
- Call, Return

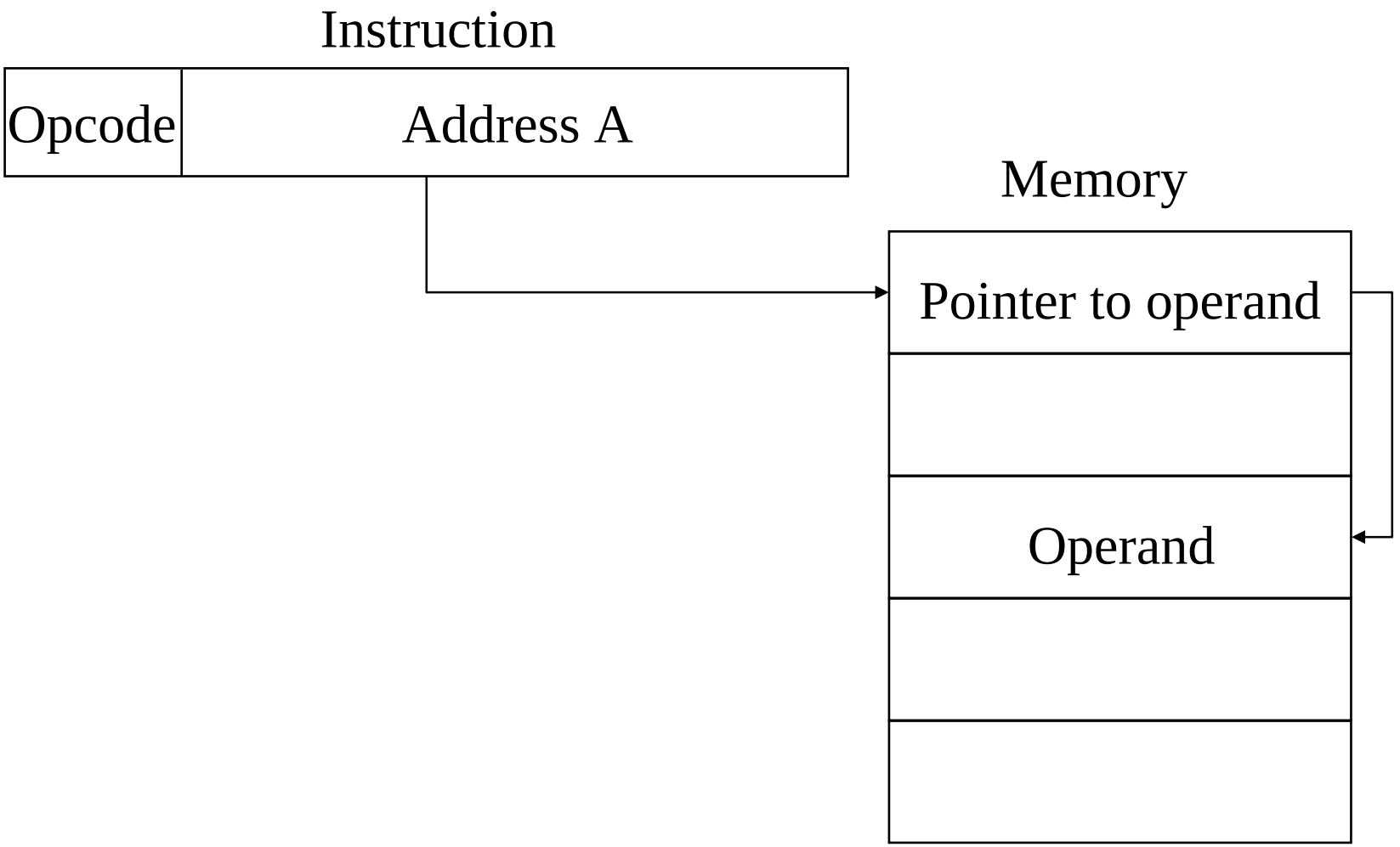
Addresses Mode

- An Address Tells Us Where the Data to be Processed Exists?
- So where it is possible to find the Data?
  - Given Directly: **Immediate Addressing Mode**
  - In CPU Register: **Register Addressing Mode**
  - In Memory:
    - **Direct Addressing Mode**
    - **Indirect Addressing Mode**
    - **Stack Addressing**
    - **Register Indirect Addressing Mode**
    - **Displacement Addressing Mode**

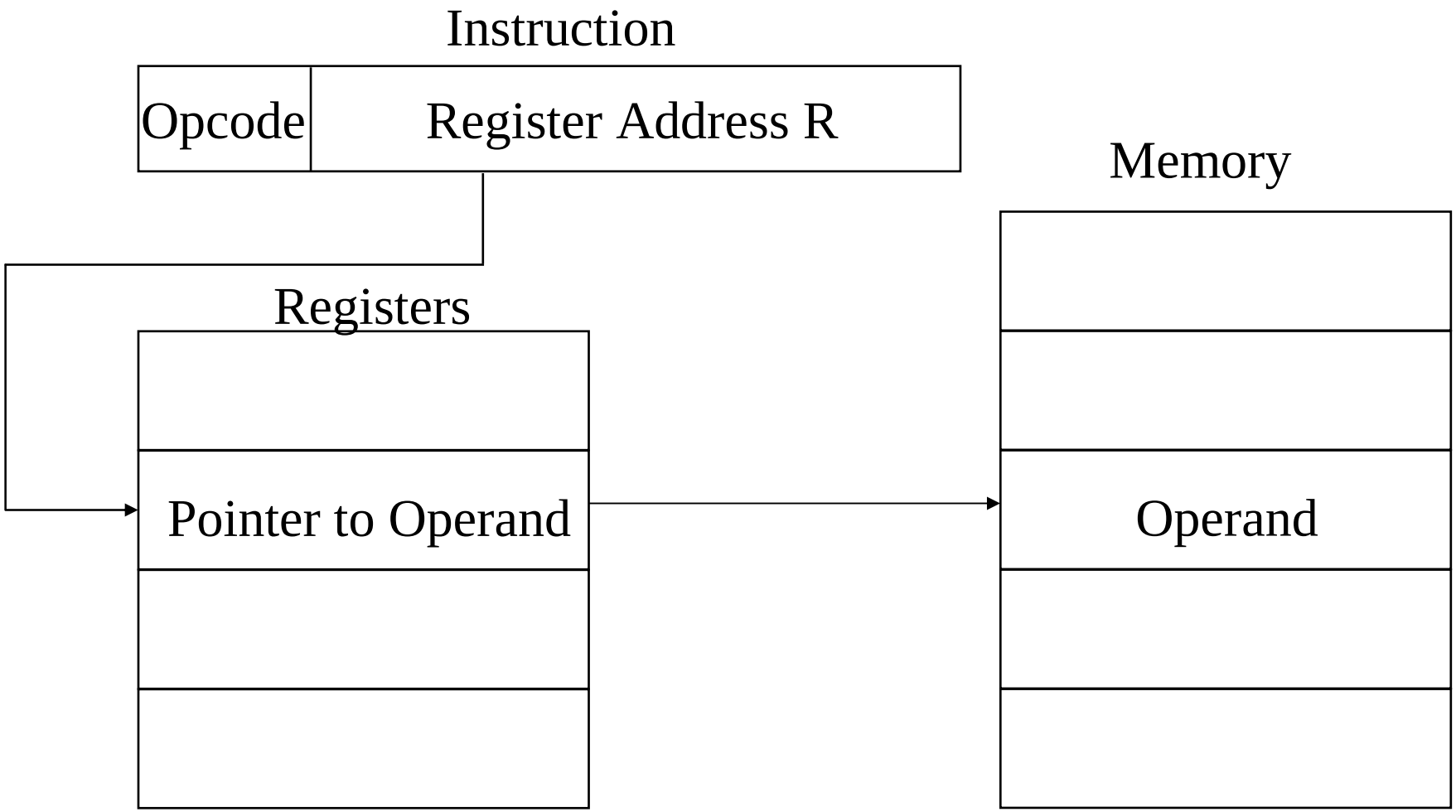
Direct Addressing Diagram



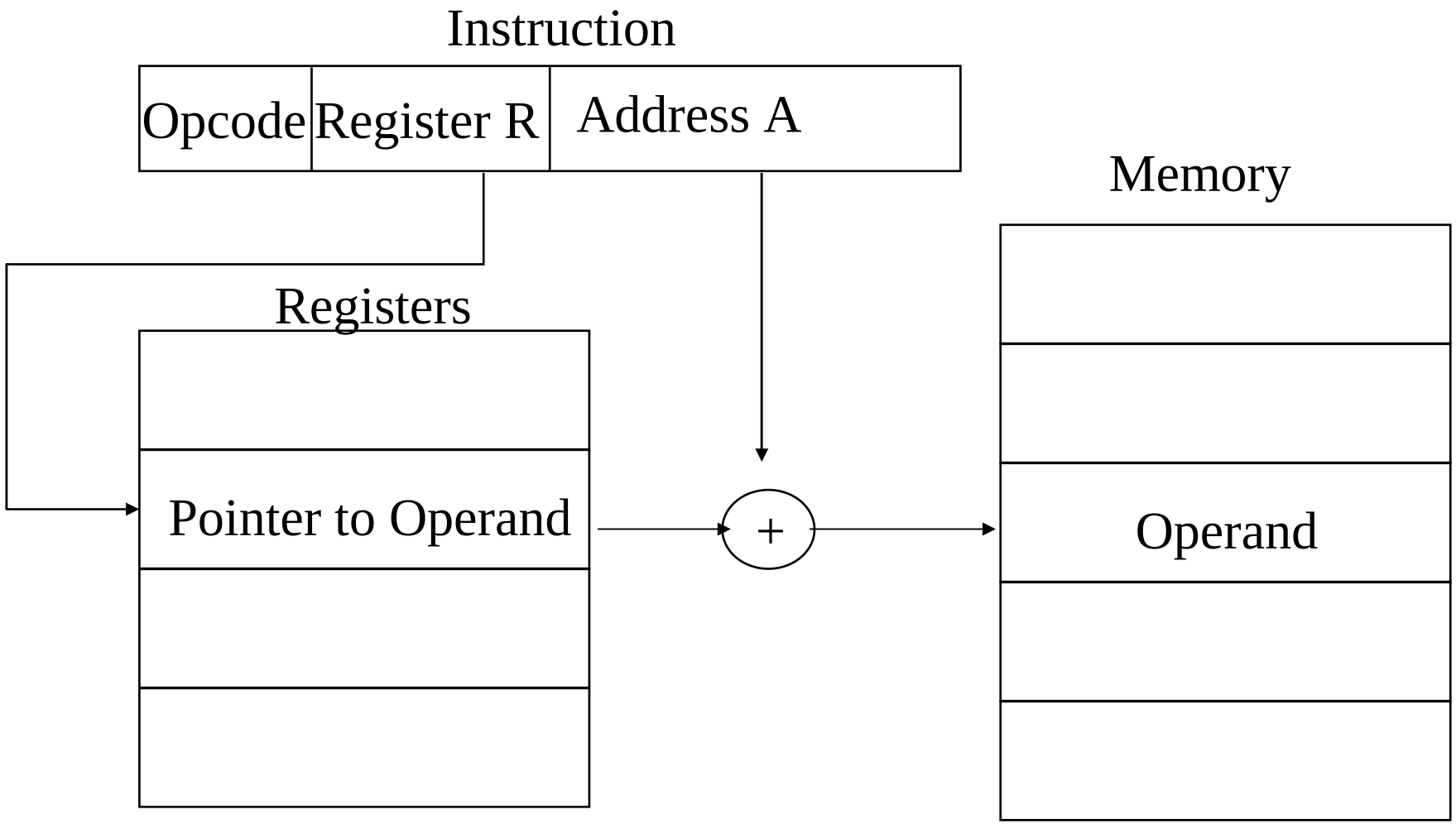
Indirect Addressing Diagram



Register Indirect Addressing Diagram



Displacement Addressing Diagram



Displacement Addressing Types

- Relative Addressing
- Base-Register Addressing
- Indexed Addressing

Combinations

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

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

```
        move r0, 10
        move r1, 0
        move r2, 0
 L1:
        BRE r1, r0, Exit
        shl r3, r1, 1
        add r2, r2, r3
        add r1, r1, 1
        BR L1
Exit:
        Load r4, (r2)
        store r4, 8(r2)
```

Homework

1. For the following data structures, draw the big-endian and little-endian layouts

a. struct {
    double i; //0x1112131415161718
};

b. struct {
    int i; //0x11121314
    int j; //0x15161718
};

c. struct {
    short i; //0x1112
    short j; //0x1314
    short k; //0x1516
    short l; //0x1718
};

Homework

- Convert the expression A + B - C to postfix notation.
- Show the steps involved. Is the result equivalent to (A + B) - C or A + (B – C) ?
- Does it matter?