EE321

Computer Architecture

Chap. 06 : Instruction Set and Addressing Modes

Dr. Abdelhakim Khouas

Email : akhouas@hotmail.com

ab.khouas@univ-boumerdes.dz

IGEE (ex. INELEC)

University M'hamed Bougara of Boumerdes

Course chapters

1. Review of Digital Design
2. Top Level of Computer
3. Central Processing Unit (CPU)
4. Control Unit
5. Memory
6. Instruction Set and Addressing Modes

Lecture Objectives

In this chapter, we will focus on characteristics and elements of instruction set and how to specify the operations and operands of instructions. Specifically, we will examine:

- Types of operands and operations that may be specified by machine instructions
- Addressing modes

Readings

Textbook

Computer Organization and Structure, Designing for Performance, By William Stallings, 8th edition

Sections

- Chapter 10, sections: 10.1, 10.2 and 10.4
- Chapter 11, sections: 11.1 and 11.5

Lecture Outline

1. Processor Programming Overview
2. Instruction Set
3. Types of Operands
4. Addressing Modes
5. Types of Operations
6. Assembly Language Elements

1. Processor Programming Overview

Computer programs and applications are in High Level Language (C, C++, Java, ...etc.), but CPU can only execute simple Low-Level Machine Instruction

- It is easier to write a program in High Level Language
- The CPU understands only Low Level Instruction

1. Processor Programming Overview

To go from complex program to low level instructions, many interpreters are used:

- Pre-processor: expands all the macros definitions and include statements
- Compiler: effectively translates preprocessed code into assembly code
- Assembler: translates the assembly code into machine code; the resulting file is called an object file
- Linker: links object file with other object files and libraries to produce a program
  - assigns absolute memory locations to everything and resolve any unresolved references

2. Instruction Set

What is Instruction Set ?

Instruction Set is the complete collection of instructions (Machine Codes) that are understood (can be executed) by a CPU

- Instructions are just collection of bits (binary)

Instruction Set is the interface between the CPU designer and the CPU user

2.1 Elements of an Instruction

Each instruction must contain the information required by the processor for execution. Instruction must contain the following elements:

1. Operation Code (Opcode): Specifies the operation to be performed (e.g., ADD), the operation is specified by a binary code

2.1 Elements of an Instruction

2. Source operand reference: one or more source operands, operands are inputs for the operation
3. Result operand reference: The operation may produce a result.
4. Next instruction reference: This tells the processor where to fetch the next instruction after the execution of this instruction is complete

2.1 Elements of an Instruction

Source and result operands can be in one of four areas:

1. CPU register: a processor contains one or more registers that may be referenced by machine instructions.
   - If only one register exists reference to it may be implicit.
   - If more than one register exists, then each register is assigned a unique name or number

2.1 Elements of an Instruction

Source and result operands can be in one of four areas:

2. Main or virtual memory: the main or virtual memory address must be supplied.
3. Immediate: The value of the operand is contained in a field in the instruction being executed.
4. I/O device: The instruction must specify the I/O module and device for the operation. If memory-mapped I/O is used, this is just another main or virtual memory address.

2.2 Instruction representation

Within the computer, each instruction is represented by a sequence of bits

- The instruction is divided into fields, corresponding to the constituent elements of the instruction

Example:

- Simple example of an instruction format

2.2 Instruction representation

Symbolic Representation

It is difficult for both the programmer and the reader to deal with binary representations of machine instructions, thus, it has become common practice to use a *symbolic representation of machine instructions*

- Opcodes are represented by abbreviations, called *mnemonics (e.g. ADD, LOAD, ...etc.)*
- *Operands are also represented symbolically (e.g. ADD A,mem for A=A+[mem])*

2.3 Instruction Types

we can categorize instruction types as follows:

- Data processing: Arithmetic and logic instructions
- Data storage: Movement of data into or out of register and or memory locations
- Data movement: I/O instructions
- Control: Test and branch instructions

2.4 Number of addresses

In most CPU architectures, most instructions have one, two, or three operand addresses

- The number of addresses per instruction is a basic design decision

2.4 Number of addresses

How many addresses?

- More addresses: More complex (powerful?) instructions, more registers, and fewer instructions per program
- Fewer addresses: less complex (powerful?) instructions, more instructions per program, and faster fetch/execution of instructions

Most current CPU employ a mixture of two- and three-address instructions

2.5. Instruction Set Design

One of the most analyzed aspects of computer design is instruction set design

- The design of an instruction set is very complex because it affects so many aspects of the computer system
- The instruction set defines many of the functions performed by the processor and thus has a significant effect on the implementation of the processor

2.5. Instruction Set Design

Most important fundamental design issues are:

- Operation repertoire
  - How many and how complex are operations?
  - What can they do?
- Data types
  - The various types of data upon which operations are performed
- Instruction formats
  - Length of opcode field
  - Number of addresses

2.5. Instruction Set Design

- Registers
  - Number of CPU registers available
  - Which operations can be performed on which registers?
- Addressing modes
  - The mode or modes by which the address of an operand is specified (see next chapter)
- RISC vs CISC
  - Reduced vs. Complex Instruction Set Computer

These issues are highly interrelated and must be considered together in designing an instruction set

3. Types of Operands

Machine instructions operate on data, the most important general categories of data are:

- Logical data
- Characters
- Numbers (two's complement representation)
- Addresses
  - Different addressing modes

3. Types of Operands

Numbers

- All machine languages include numeric data types
- Numbers in computers are limited in the magnitude and resolution for floating-point
- Three types of numerical data are common in computers:
  - Binary integer or binary fixed point
  - Binary floating point
  - Decimal BCD (Binary Coded Decimal)
- Example: $01000011 = (43)_{BCD} = (67)_{@2} = (4.1875)_{Q4}$

4. Addressing Modes

Addressing modes refers to the way to identify the data (location of the data) in an instruction

- The address field or fields in a typical instruction format are relatively small
- Addressing modes allow the CPU to reference a large range of locations in memory
- Addressing modes involve some trade-off between address range and/or addressing flexibility, and the number of memory references and/or the complexity of address calculation

4. Addressing Modes

All processors provide more than one addressing mode. The addressing mode used in a given instruction can be specified by:

- Opcode
- Mode field in the instruction

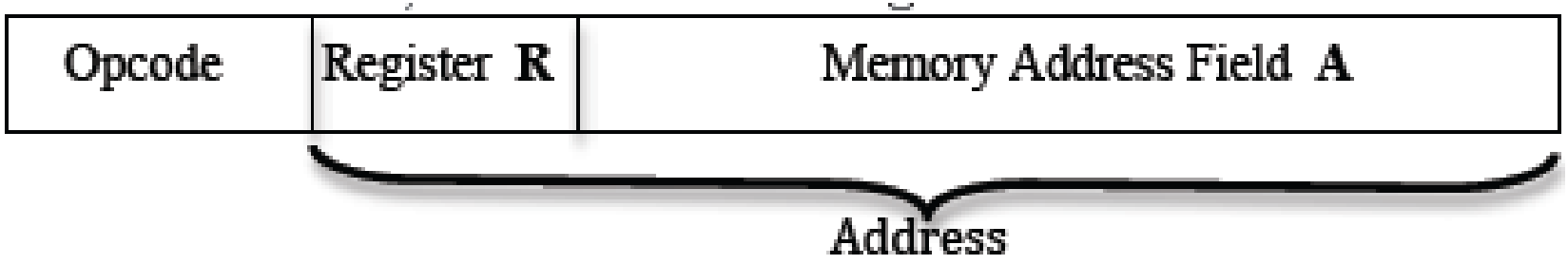
4. Addressing Modes

The number of addressing modes differs from one processor to another

- RISC processors have a few numbers of addressing modes while CISC processors can have a dozen of addressing modes
- Naming the different addressing modes differs from processors manufacturers

4. Addressing Modes

Instruction Format



- A : content of an address field
- R : Address field that refers to a register
- EA : Effective (actual) Address of the location containing the operand. EA is either a memory address or a register
- (X) : contents of memory location X

4. Addressing Modes

The most common addressing techniques are:

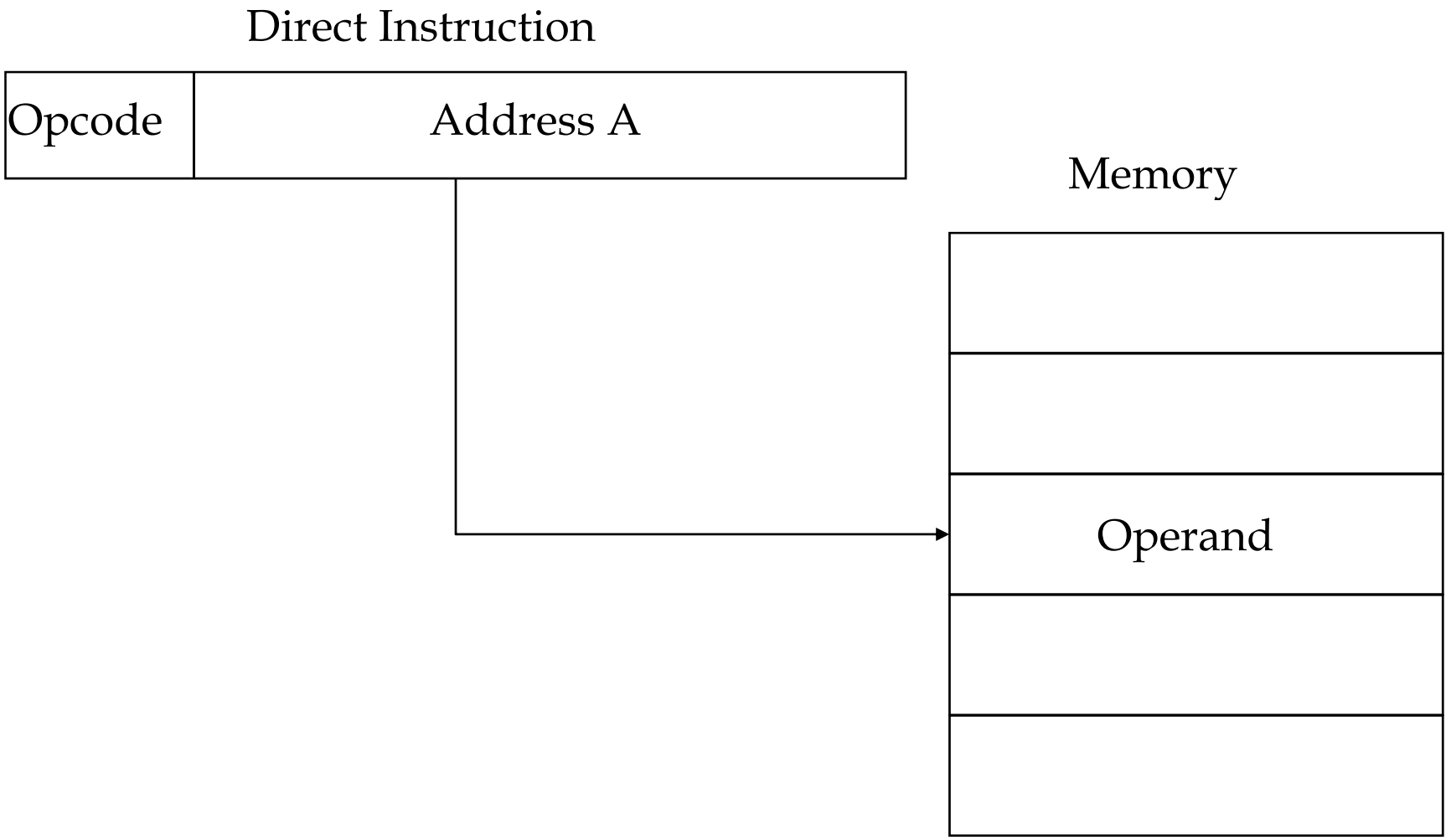
- Immediate
- Direct
- Indirect
- Register
- Register indirect
- Displacement
- Stack

4. Addressing Modes

Immediate: Operand is part of instruction

- Simplest form of addressing
- Example: ADI AC,F5H or ADD AC, #F5H
- Advantages:
  - No memory reference
  - fast instruction
- Disadvantage:
  - restricted size of the number because of the size of the operand field

4. Addressing Modes

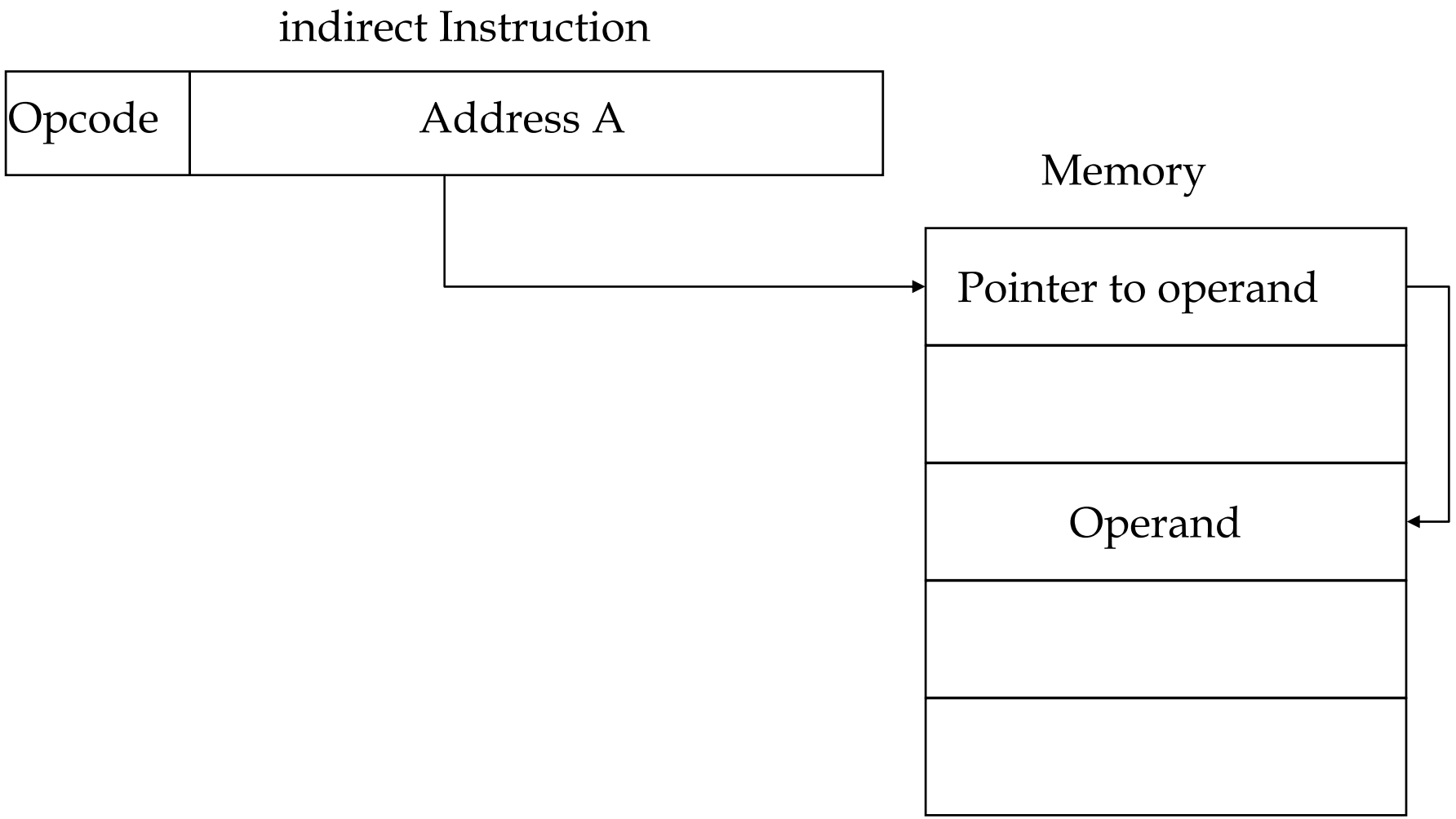


4. Addressing Modes

Direct: Address field contains address of operand

- EA= A (i.e. data = (A))
- Example: ADD AC, F5H
  - Add contents of cell F5H to accumulator
  - Look in memory at address F5H for operand
- Advantages:
  - Single memory reference to access data
  - No additional calculations to work out effective address
- Disadvantage:
  - Limited address space because of the size of the address field (limited to 2^k, k is length of the address field)

4. Addressing Modes

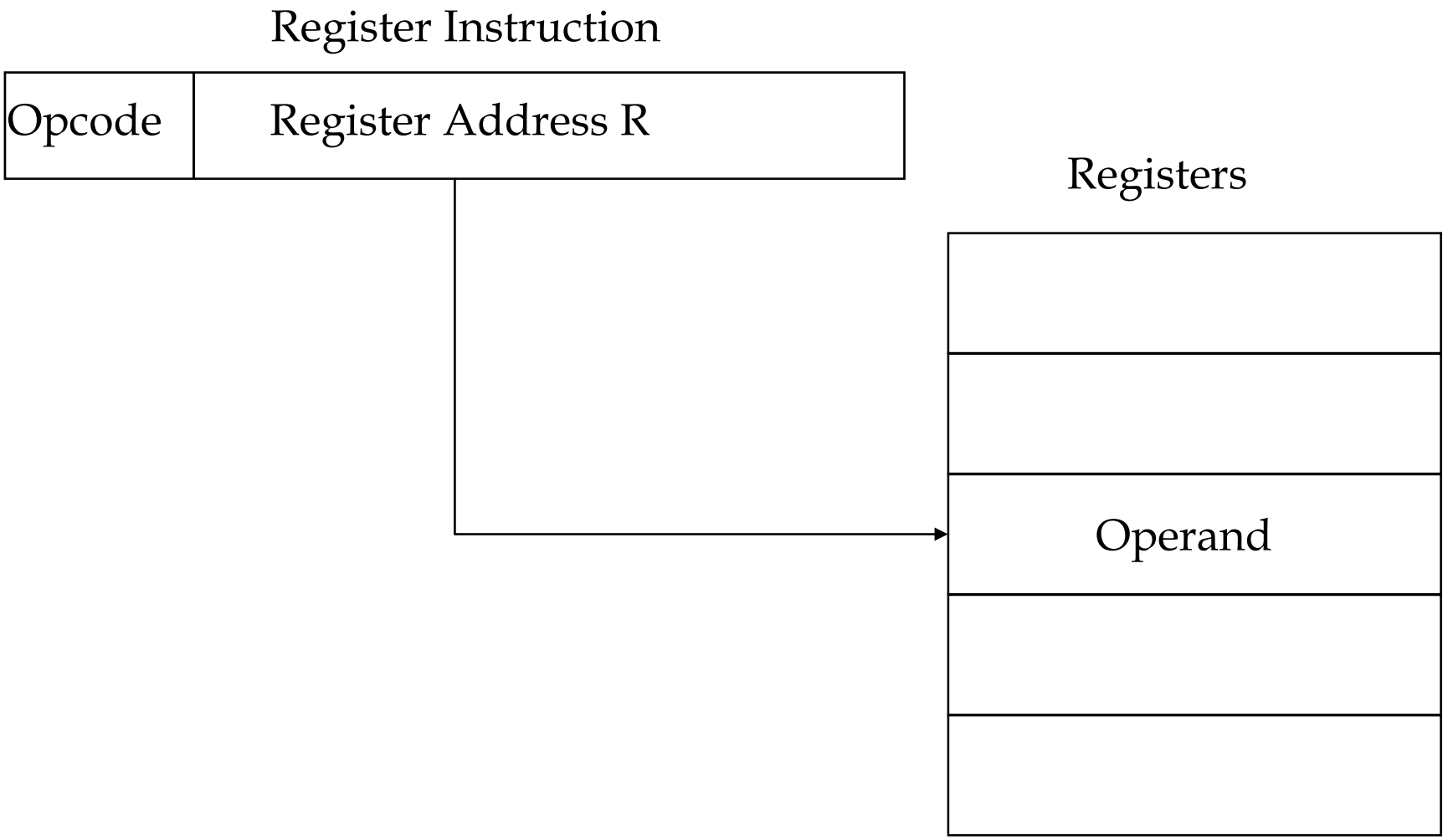


4. Addressing Modes

Indirect: Address field contains the address of the pointer to the operand

- EA = (A) (i.e. data = ((A))
  - Look in A, find address (A) and look there for operand
- Advantages:
  - Large address space (2^n, n is word length)
- Disadvantage:
  - Multiple memory accesses to find operand
  - Slower

4. Addressing Modes

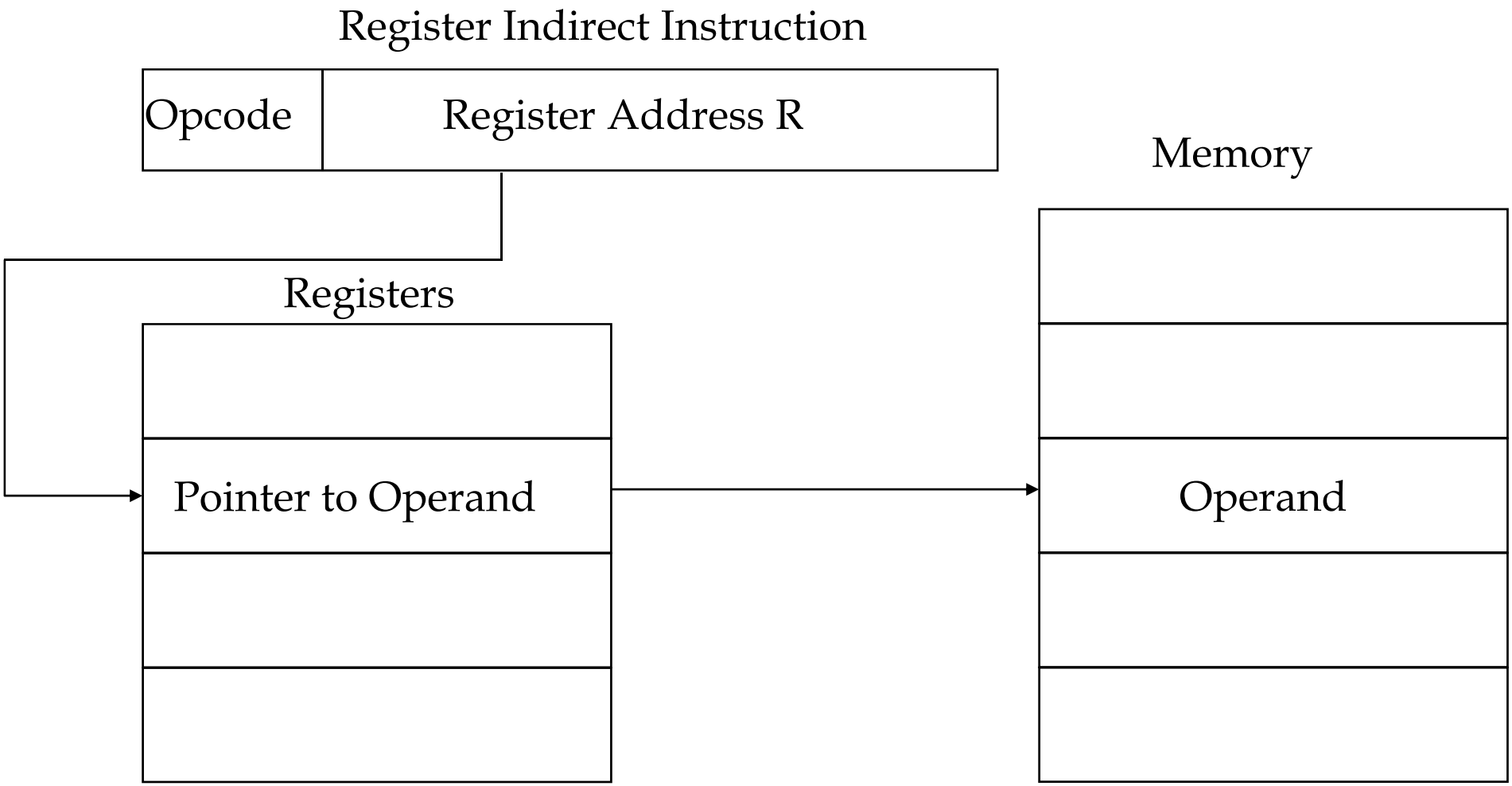


4. Addressing Modes

Register: Address field contains a register containing the operand

- Data = content of register R (i.e. data=(R))
- Disadvantage:
  - Limited number of registers (limited address space)
- Advantages:
  - Very small address field needed (for 32 registers, only 5 bits)
  - Shorter instructions and faster instruction fetch
  - No memory access
  - Very fast execution

4. Addressing Modes

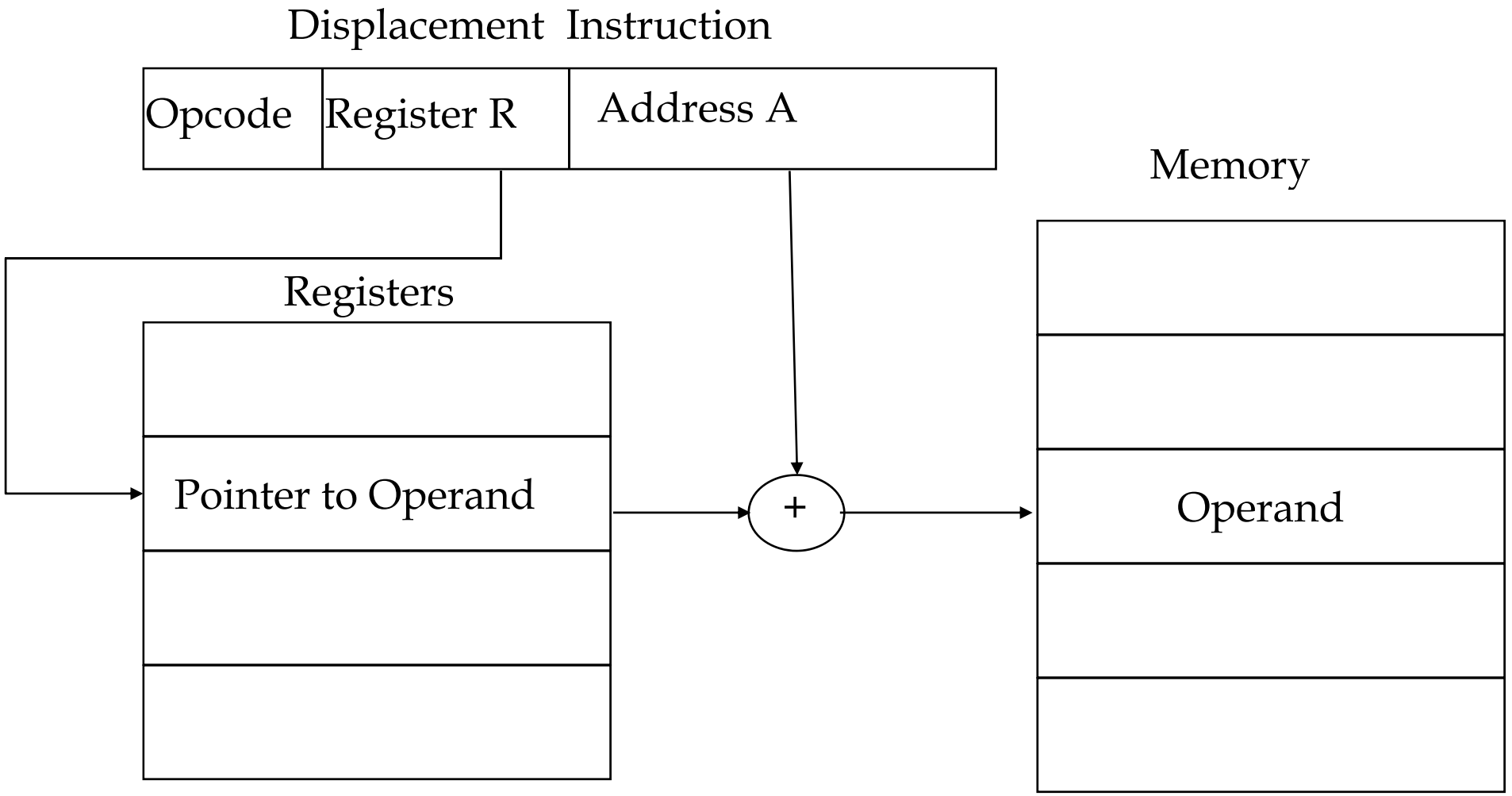


4. Addressing Modes

Reg.-Ind.: Address field contains a register containing the address of the operand

- EA = R (i.e. data = ((R)))
- Operand is in memory cell pointed to by the content of register R
- Advantages:
  - Large address space (2^n)
  - One fewer memory access than indirect addressing

7. Displacement (Indexed) Addressing



4. Addressing Modes

Displacement: Address field contains two values: base and displacement

- At least one field is explicit
- EA = A + R (i.e. data = (A+(R))
  - A = base value and R = register that holds displacement or vice versa
- A very powerful mode of addressing
  - Combines the capabilities of direct addressing and register indirect addressing,

4. Addressing Modes

Displacement: Relative Addressing Mode

- A version of displacement addressing
- Implicit resister is Program counter (i.e. R=PC)
- EA= A + (PC)
- Get operand from A cells from current location pointed to by PC
- Advantage:
  - Relative addressing saves address bits in the instruction

4. Addressing Modes

Displacement: Indexed Addressing Mode

- A version of displacement addressing
- A = base
- R = displacement
- EA = A + (R)
- Good for accessing arrays (table manipulation
  - EA = A + (R)
  - (R)++

4. Addressing Modes

Displacement: Base-Register Addressing Mode

- A= displacement
- R = holds pointer to the base address
- Register may be explicit or implicit
- Example:
  - Segment registers in x86 processor family
  - Segment registers are implicit

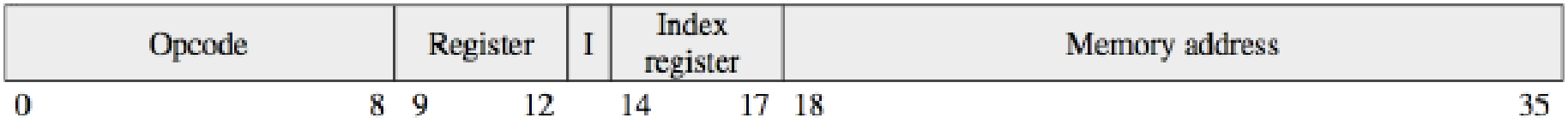
4. Addressing Modes

Stack Addressing: Operand is (implicitly) on top of stack

- Example : ADD
  - Pop top two data from stack and add them

4. Addressing Modes

Example: PDP-10 Instruction Format



- 36-bit word length and a 36-bit instruction length
- Most instructions have two addresses, one of which is one of 16 general-purpose registers (4 bits register filed). The other operand reference starts with an 18-bit memory address field. This can be used in both immediate and direct addressing

4. Addressing Modes

Example: PDP-10 Instruction Format

- For memory sizes greater than 2^{18}, indirect addressing is provided (I bit)
- Displacement addressing (indexing) is also provided for table manipulation and iterative programs, where the contents of the index register is added to the memory address field
- The same general-purpose registers are also used as index (reference) registers

4. Addressing Modes

Consider a CPU with the following instruction format:

- What is word length for this processor?
- How many instructions can this processor support?
- How many registers it can contain?
- How many memory locations can this processor address?
- What is the range of 2's complement constants that can be specified using Immediate Addressing?
- How many memory locations can this processor address using Direct Addressing?
- How many memory locations can this processor address using Indirect Addressing?

5. Types of Operation

The number of different opcodes varies widely from machine to machine, same general types of operations are found on all machines:

- Data transfer
- Arithmetic
- Logical (NOT, AND, OR, XOR, SHIFT and ROTATE)
- Conversion (e.g decimal<->binary)
- I/O
- Transfer of control

5. Types of Operation

Transfer of control (Branching) Operation

- for all of the operation types discussed, the next instruction to be performed is the one that immediately follows the current instruction (in memory)
- However, several instructions in any program needs to change the sequence of execution
  - for these instructions, the CPU updates the PC to contain the address of the target instruction

5. Types of Operation

Transfer of control (Branching) Operation

The transfer-of-control operations are required in the following cases:

1. Loops: to be able to execute code more than once
2. Conditional statements: to do one thing if one condition holds, and another thing if another condition holds (if statement)
3. Procedures or functions: mechanisms for breaking the task up into smaller pieces that can be worked on one at a time

5. Types of Operation

Transfer of control (Branching) Operation

The most common transfer-of-control operations found in instruction sets are:

1. Branch Instruction
2. Skip Instruction
3. Procedure Call Instruction

5. Types of Operation

Branch Instructions

- Also called a Jump Instruction
- It has as one of its operands the address of the next instruction to be executed (target address)
- Unconditional Branch
  - PC= new address
- Conditional Branch
  - If condition satisfied, PC=new address else PC=PC+1

5. Types of Operation

Conditional Branch Instructions

There are two common ways of generating the condition to be tested in a conditional branch instruction

1. Using 1-bit or multiple-bit condition code (flag) that is set as the result of last operation
   - BRP X Branch to location X if result is positive
   - BRN X Branch to location X if result is negative
   - BRZ X Branch to location X if result is zero
   - BRC X Branch to location X if carry occurs
   - BRO X Branch to location X if overflow occurs

5. Types of Operation

Conditional Branch Instructions

2. Using three-address instruction format to perform a comparison and specify a branch in the same instruction
   - Example: BRE R1, R2, X Branch to X if contents of R1 = contents of R2

5. Types of Operation

Conditional Branch Example:

Z80	x86	Comment
JP X	JMP X	jump to X unconditionally
JR X		relative jump
JP Z, X	JZ X	branch/jump to X on zero
JR Z, X		
JP PE, X	JO X	branch/jump to X on overflow
JR PE, X		

5. Types of Operation

Skip Instruction

- The skip implies that one instruction be skipped, thus, the implied address equals the address of the next instruction
- Because the skip instruction does not require a destination address field, it is free to do other things
  - Example: the increment-and-skip-if-zero (ISZ) instruction

5. Types of Operation

Procedure Call Instruction

- A procedure is a piece of a program incorporated into a larger program
- At any point in the program, the procedure may be called
- The CPU executes the procedure and return to the calling program
- Advantages:
  - Economy: same code used many times (less storage)
  - Modularity: large tasks subdivided into smaller units

5. Types of Operation

Procedure Call Instruction

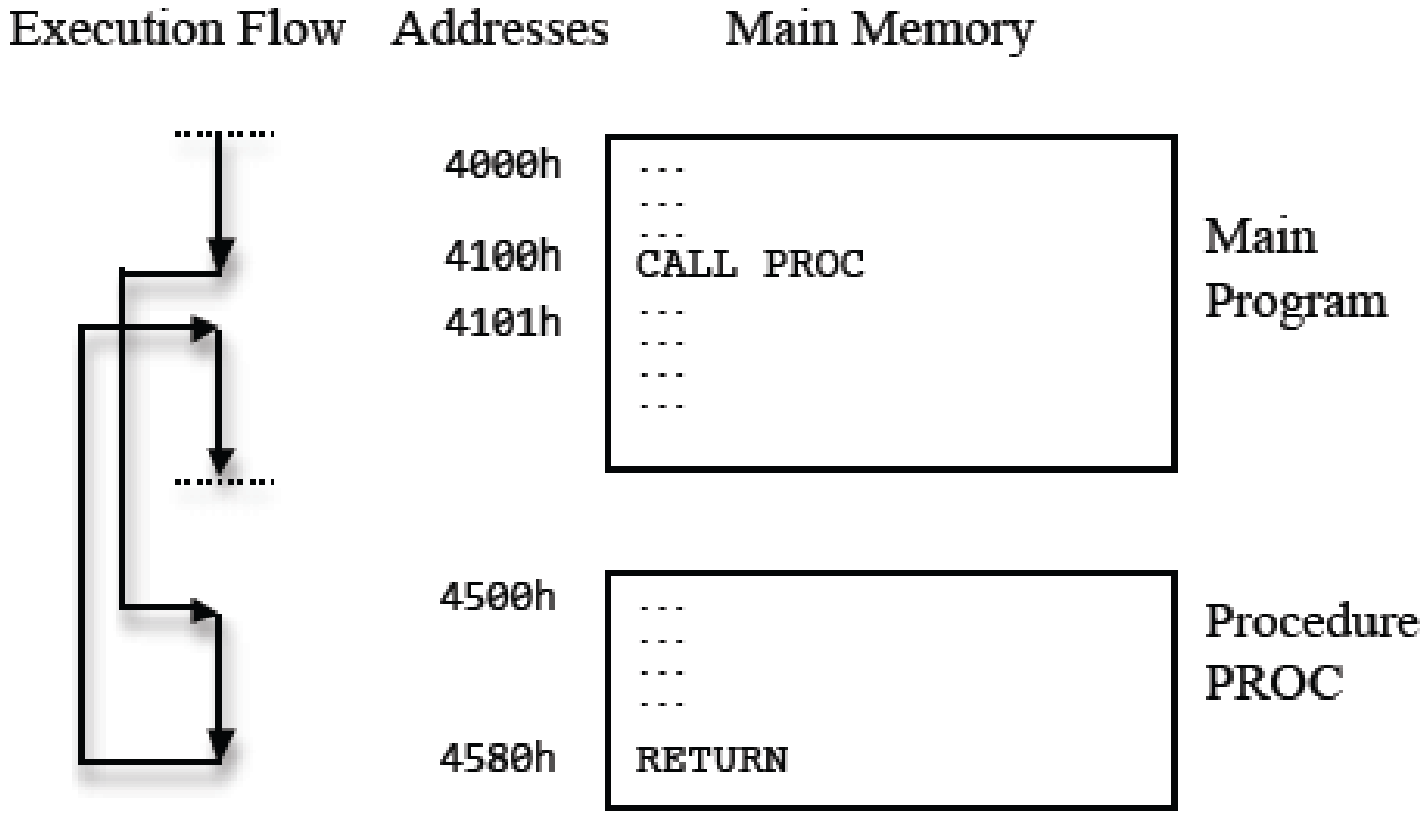
The procedure mechanism involves two basic instructions:

1. A call instruction that branches from the present location to the procedure
2. A return instruction that returns from the procedure to the place from which it was called

Both of these instructions are forms of branching instructions

5. Types of Operation

Procedure Call Instruction



5. Types of Operation

Procedure Call Instruction

To be able to call a procedure from a variety of points, the CPU must save the return address. There are three common places for storing the return address:

- Register
- Start of called procedure
- Top of stack (more powerful approach)
  - Used in Z80 processor

6. Assembly Language Elements

A statement in a typical assembly language has the form shown bellow:

Example:

```
Mem. Add.   Label        Instruction    ;Comment
400H        LOOP MOVE A, B ; Copy contents of B to A
401H             INC  A    ; Increment content of A
402H             BR LOOP   ; Branch to location LOOP=400H
```

Summary

The essential elements of a computer instruction set are:

- Opcode, which specifies the operation to be performed;
- Source and destination operand references, which specify the input and output locations for the operation;
- Addressing mode of operand reference and can be:
- Immediate value,
- Direct (operand address is in address field),
- Indirect (address field is location of the operand address),
- Register,
- Register indirect, and various forms of displacement