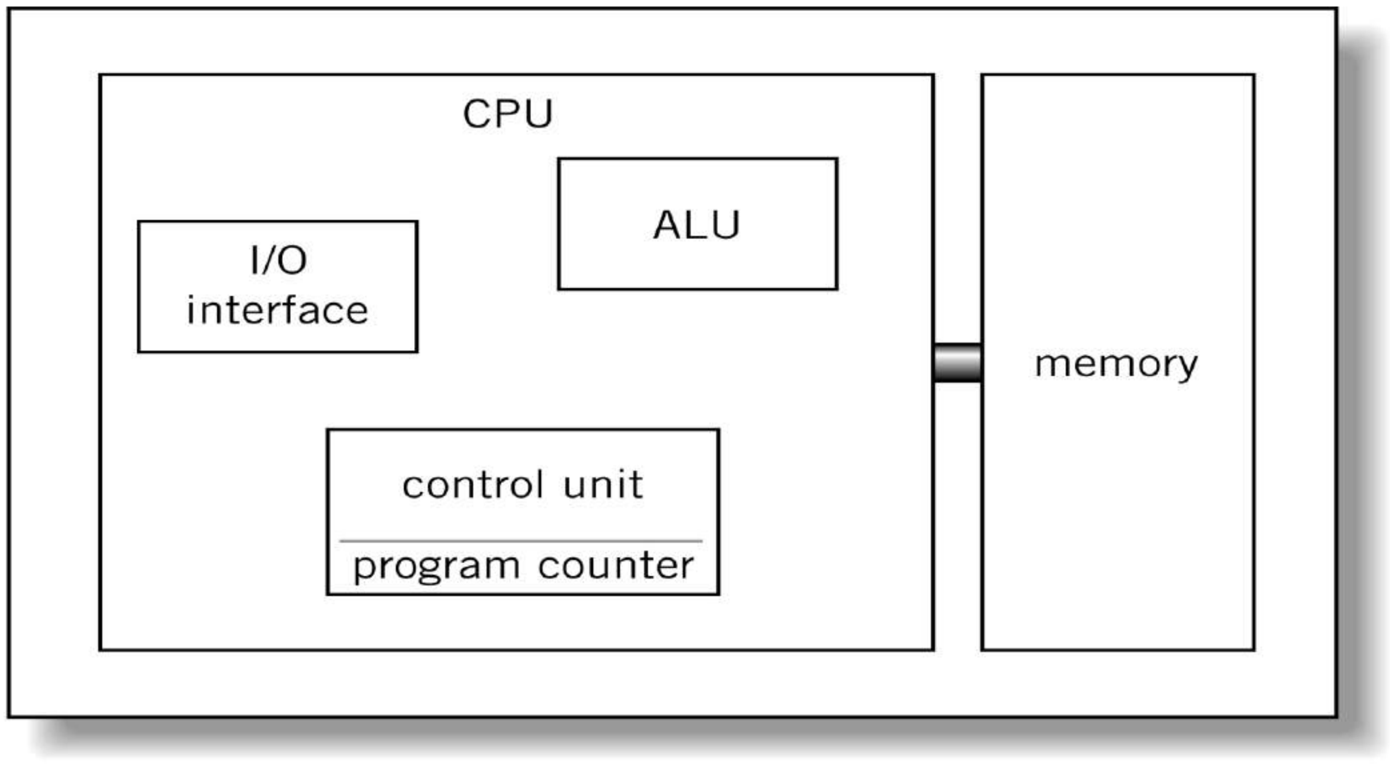
Chapter 7

Memory - Registers Instruction Sets

CPU Components



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-01

Concept of Registers

- Definition:

  Small, permanent storage location within the CPU used (wired) for a specifically defined purpose
- Manipulated directly by the CPU
- Limited in size to bits or bytes (not MB like internal memory)
- Typically configured to hold data, an address, an instruction, a status flag, etc.

General Purpose Registers

- Sometimes considered part of the ALU
- Similar to Accumulator in LMC
- Hold data used in arithmetic/logical operations as well as the results
- Used to transfer data to and from memory
- Directly accessible by programmers

General Purpose Register Operations

- Load data from memory or another register
- Data in another register or memory can be added to or subtracted from the register with result remaining in register
- Data in register can be rotated or shifted left or right
- Special - Set to 0, invert bits, +/- 1

Program Counter Register

- Holds address of current instruction being executed
- Same as LC - Location Counter in LMC

Instruction Register (IR)

- Holds instruction currently being executed
- Not part of LMC - the Moron’s brain seemed to serve this purpose in LMC

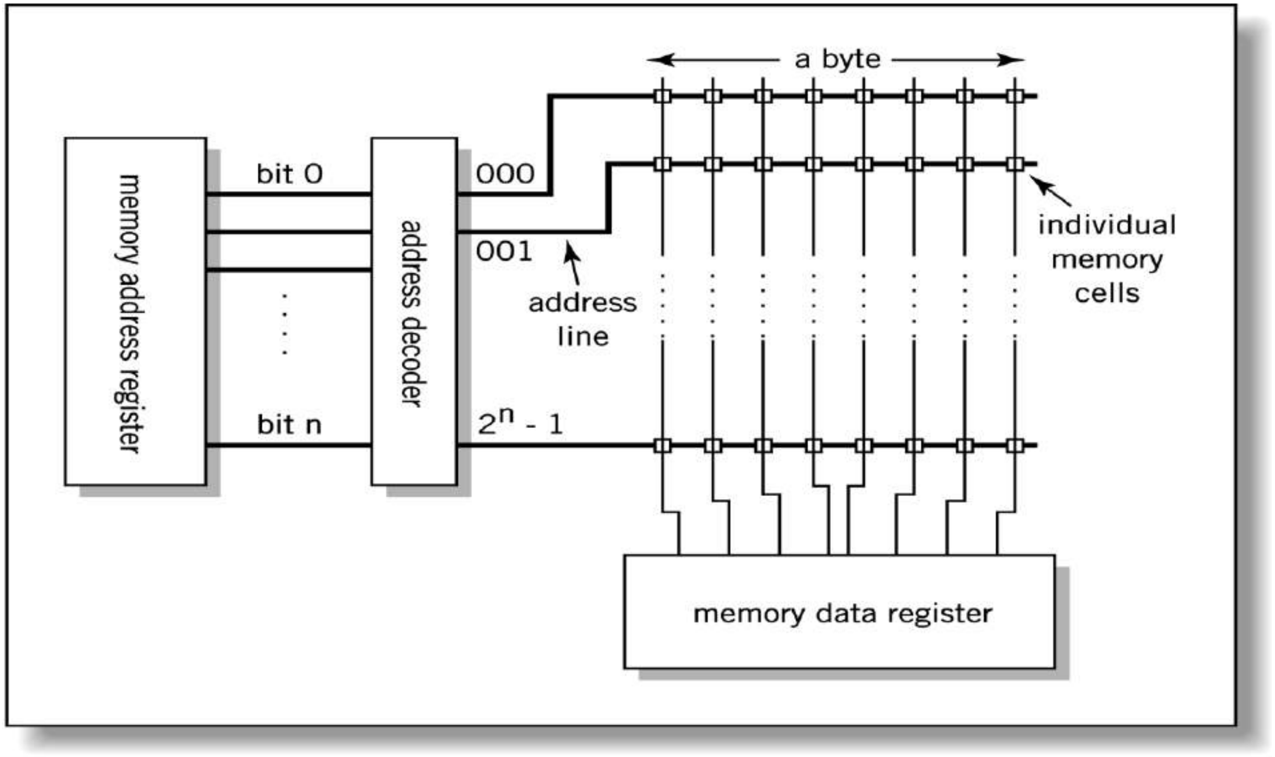
Flag/Status Registers

- Flags (1-bit) registers signal exceptional/special conditions - arithmetic overflow, carry out, comparison result, …
- Status Register - multiple flags in single register
- CPU sets status register as a result of conditions that arise during execution of instructions

MAR & MDR

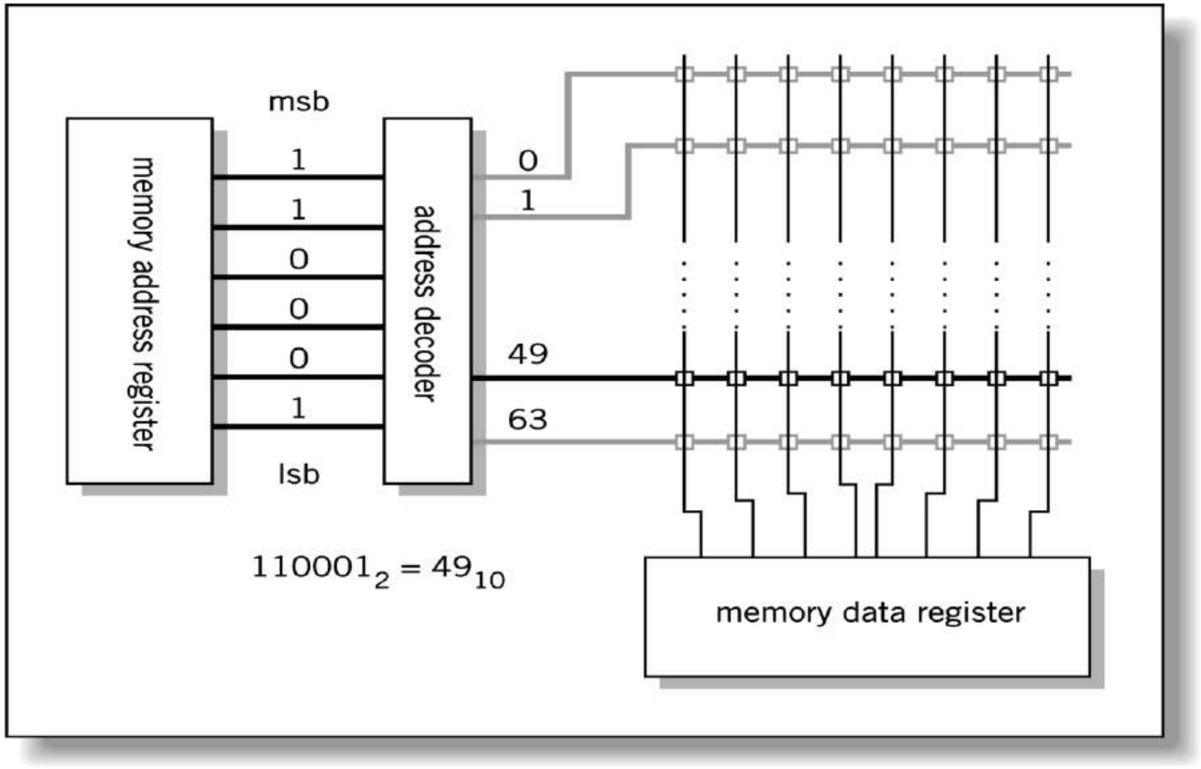
- Memory Address Register (MAR) - holds the address of a memory location
- Memory Data (buffer) Register (MAD) - holds a data value being stored or retrieved from the memory location currently addressed by MAR

MDR, MAR, Memory



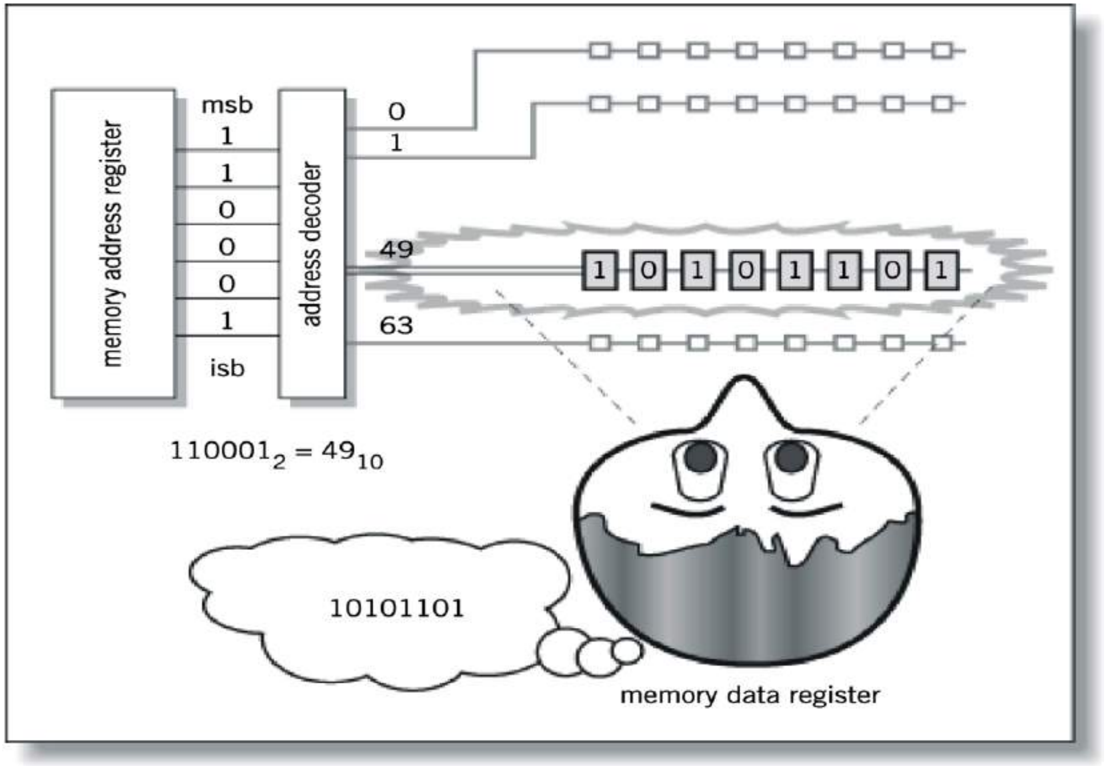
Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-04

MAR-MDR Example



Englander: The Architecture of Computer
Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-05

Memory Analogy

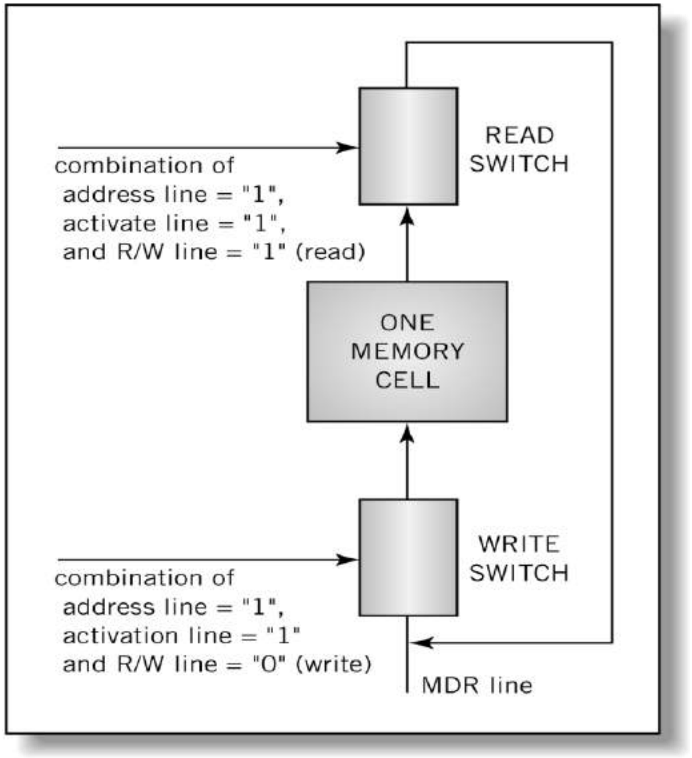


Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition Chapter 7, Figure 07-06

Individual Memory Cell

Operation:

- CPU retrieves address and stores in MAR
- CPU simultaneously sets read or write line reflecting load or store operation
- CPU turns on switch via activation line
- Data is transferred between MDR and Memory



READ SWITCH
combination of address line = "1", activate line = "1", and R/W line = "1" (read)
ONE MEMORY CELL
WRITE SWITCH
combination of address line = "1", activation line = "1" and R/W line = "0" (write)
MDR line

Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition Chapter 7, Figure 07-07

Memory Capacity

- Two factors determine the capacity of memory
  - number of bits in the MAR
    - K bits allow 2^k memory locations
  - size of the address portion of the instruction
    - 5 bits allows 32 locations
    - 8 bits allows 256 locations
    - 24 bits allows 16 MBytes
    - 32 bits allows 4GBytes

CPU Fetch-Execute Cycle Load Instruction

PC -> MAR	**Transfer the address from the PC to the MAR**
MDR -> IR	**Transfer the instruction to the IR**
IR(address) -> MAR	**Address portion of the instruction loaded in MAR**
MDR -> A	**Actual data copied into the accumulator**
PC + 1 -> PC	**PC incremented**

Elements of Machine Instructions

- **OPCODE**
- **Source OPERAND (address)**
- **Result OPERAND (address)**

OPCODE	Source Address	Result Operand

Addressing

Addressing may reflect a reference to a memory location or to a register.

- **Explicit address -** ***the address (or its components) is present in the instruction***
- **Implicit address -** ***the address is assumed, it is not in the instruction***

Most address references in today's computers are explicit.

What is the advantage of an IMPLICIT address?

Source/Destination Addresses

- Computer instructions generally specify at least two operands: a data source location and a data destination location
- In general, at least one of the addresses is a register address

- 1 address is required for in-place instructions (such as invert): 1 source/destination
- 2 addresses are required for move instructions (such as load): 1 source and 1 destination
- 3 addresses are required for arithmetic operations (such as add): 2 sources and 1 destination

3-Address Instructions

$$Y = (A - B) / (C + D + E)$$

Instruction	Comment
SUB Y, A, B	$Y \leftarrow A - B$
ADD T, D, E	$T \leftarrow D + E$
ADD T, T, C	$T \leftarrow T + C$
DIV Y, Y, T	$Y \leftarrow Y / T$

2-Address Instructions

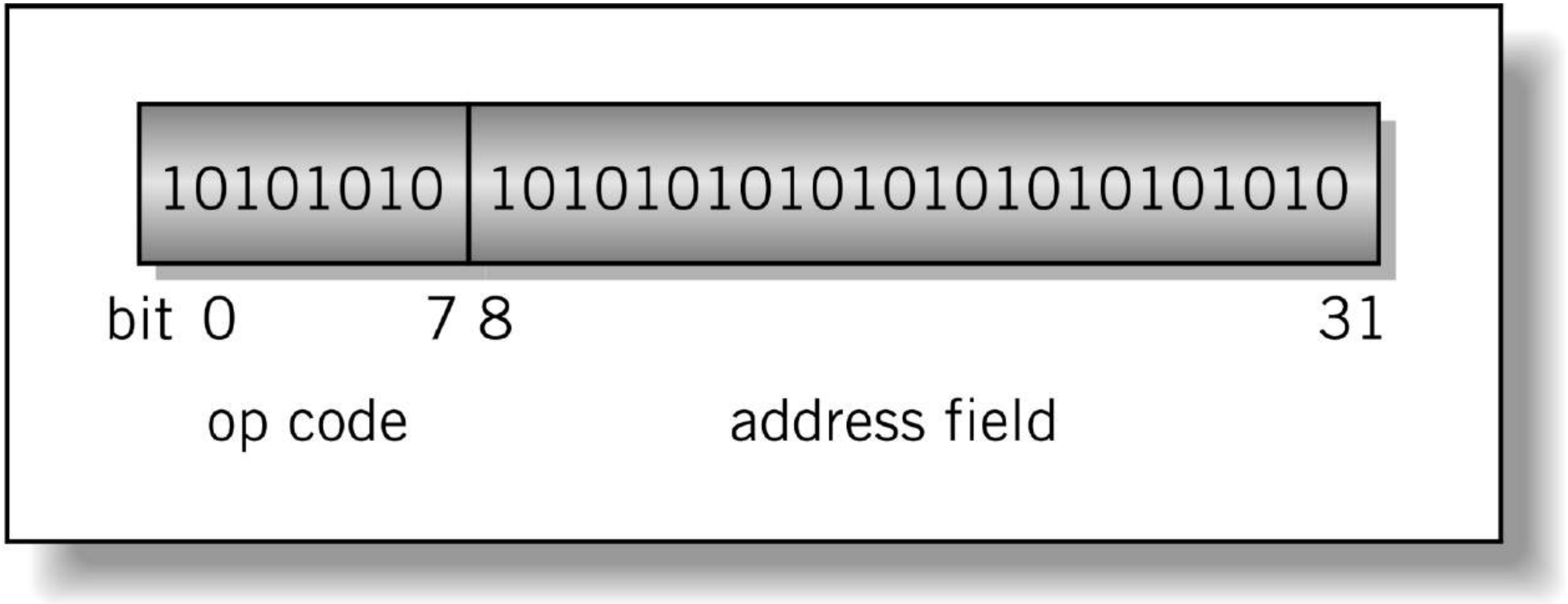
$$Y = (A - B) / (C + D + E)$$

Instruction	Comment
MOV Y, A	$Y \leftarrow A$
SUB Y, B	$Y \leftarrow Y - B$
MOV T, D	$T \leftarrow D$
ADD T, E	$T \leftarrow T + E$
ADD T, C	$T \leftarrow T + C$
DIV Y, T	$Y \leftarrow Y / T$

Instruction Word Length

- Considerations
  - Number of instructions - op code size to support reasonable set of instructions
  - Number of addresses - address size to accommodate increasing amounts of memory

Simple 32-bit Instruction



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition Chapter 7, Figure 07-14

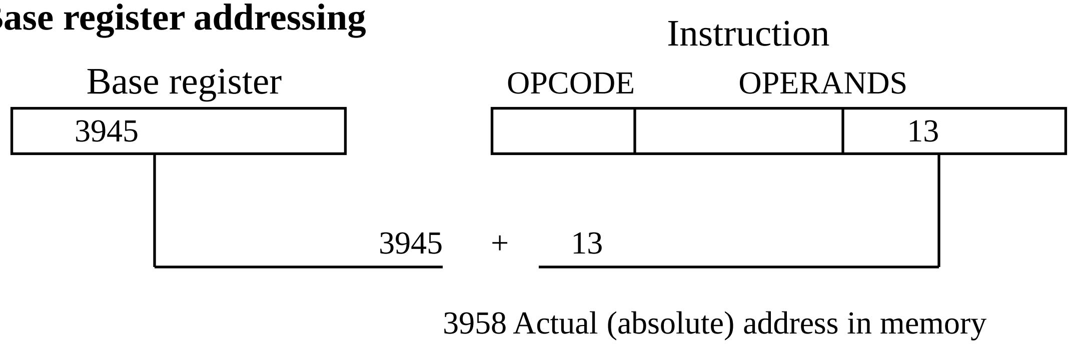
Amount of addressable memory is extremely limited.
Today’s computers support 32-bit or 64-bit addresses.

Alternatives to Absolute Addressing

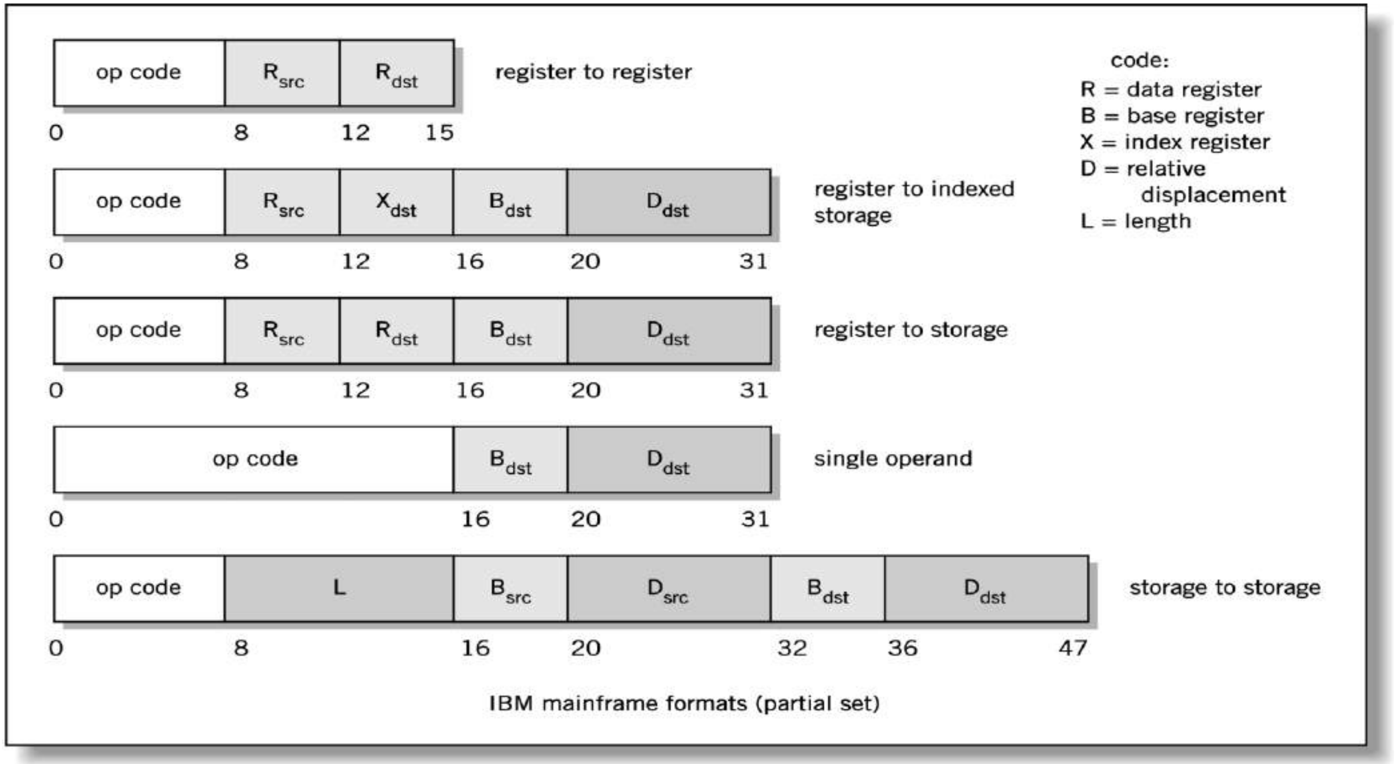
Objective :

to allow the addressing of large amounts of memory while maintaining a reasonably sized instruction word address field

Base register addressing



Instruction Formats
Variable Length Instructions

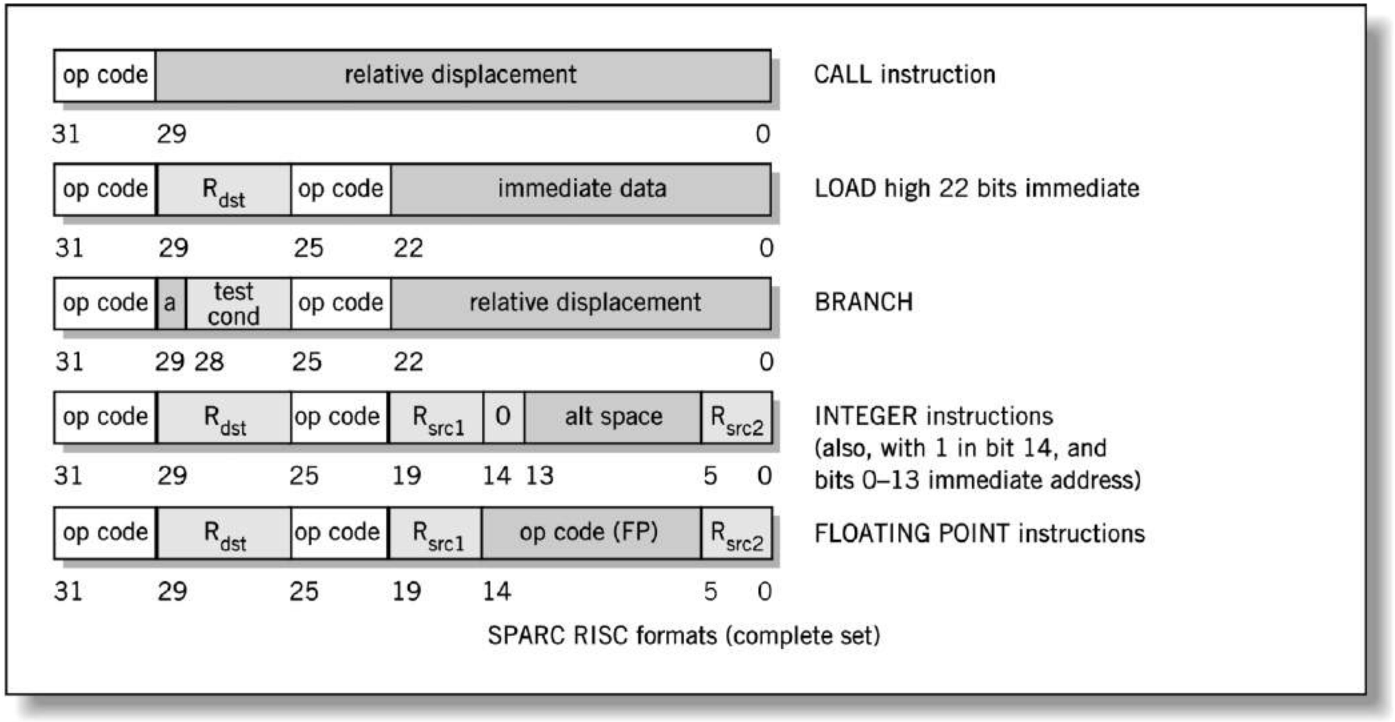


IBM mainframe formats (partial set)

(Figure continues on next slide)

Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-15

Instruction Formats
Fixed Length Instructions



SPARC RISC formats (complete set)

Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-15 continued

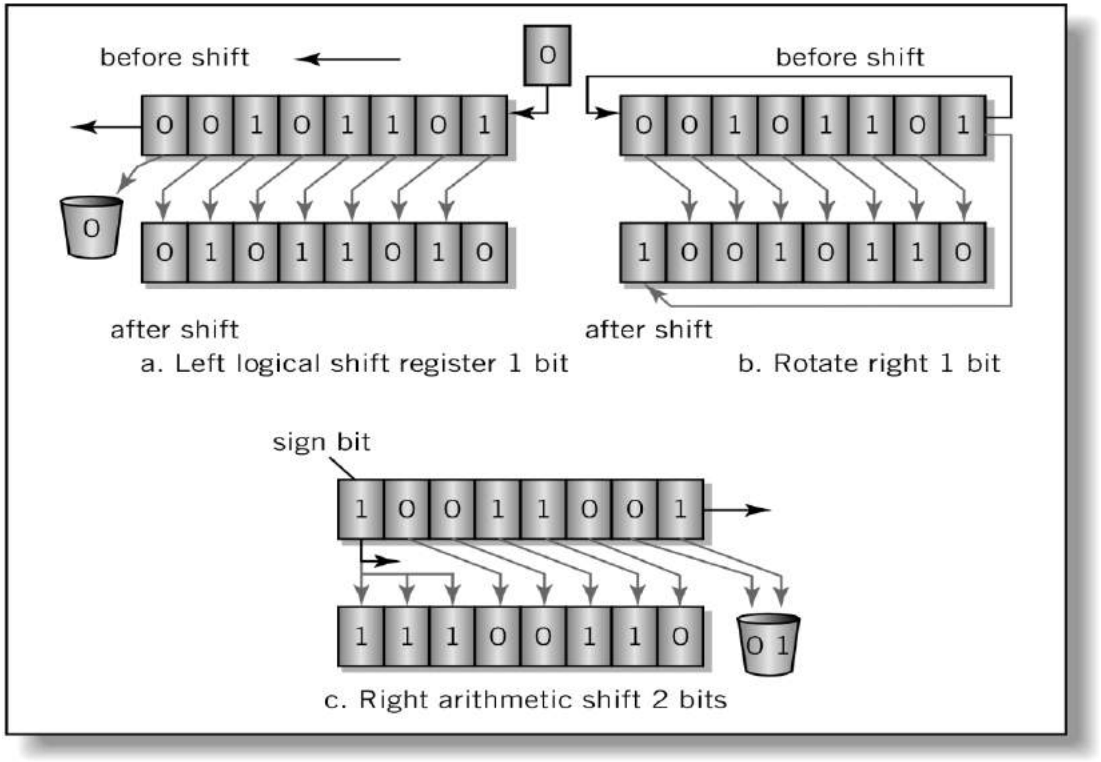
Instruction Types

- Data Transfer (Load, Store, Move, ...)
  - Register <-> Register
  - Register <-> Memory
  - Memory <-> Memory
  - Load Byte
  - Load Half-Word
  - Load Double Word

Instruction Types

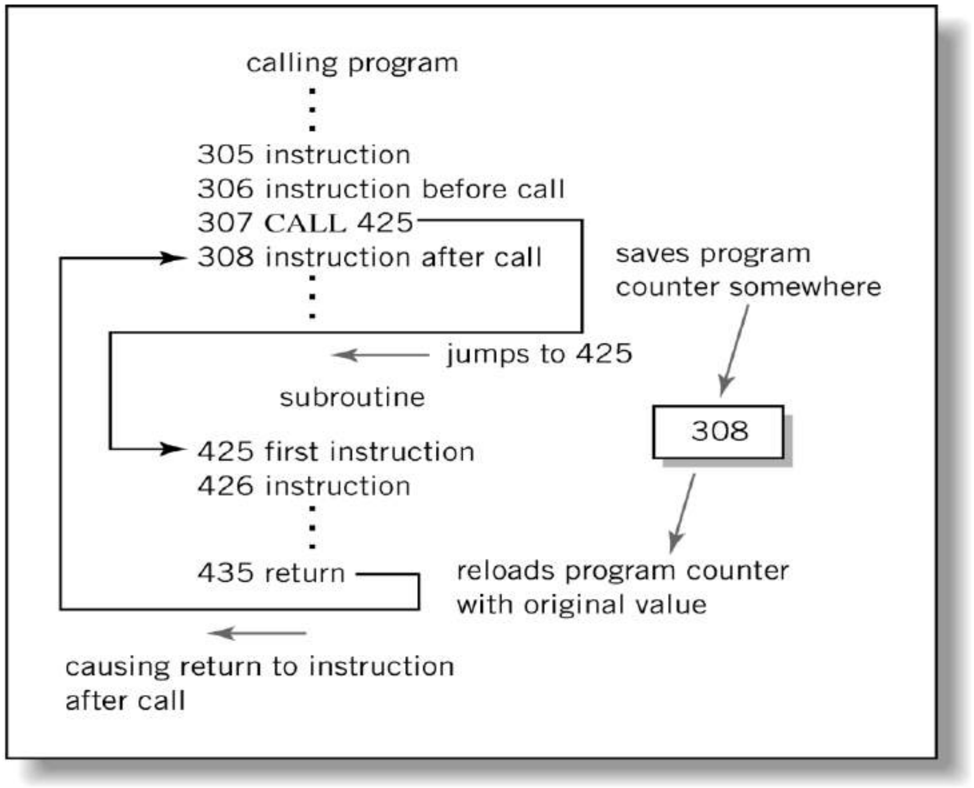
- Arithmetic
  - Integer Add/Subtract/Multiply/Divide
  - Floating Pt Add/Subtract/…
- Logical
  - NOT, AND, OR
- Bit Manipulation Instructions
  - Setting and Testing bits in a data word
- Program Control
  - Branch on 0, …
- Single operand manipulation instructions
  - Incrementing/Decrementing by 1
  - Negating
  - Setting to zero

Instruction Types
Shift and Rotate



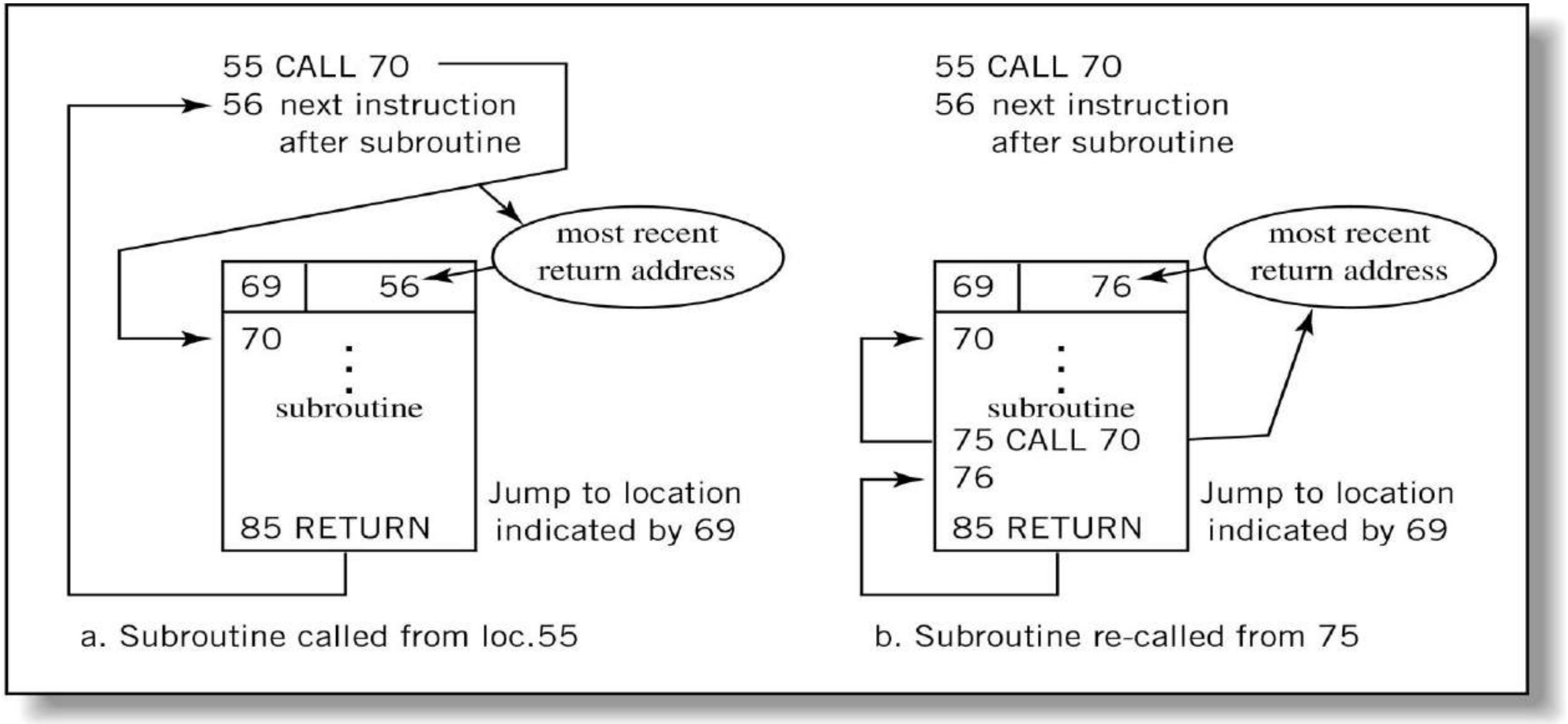
Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-17

Instruction Types
Jump Subroutine and Return


calling program
305 instruction
306 instruction before call
307 CALL 425
308 instruction after call
saves program counter somewhere
jumps to 425
subroutine
308
425 first instruction
426 instruction
435 return
reloads program counter with original value
causing return to instruction after call

Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-18

Subroutine
Recursive Calls



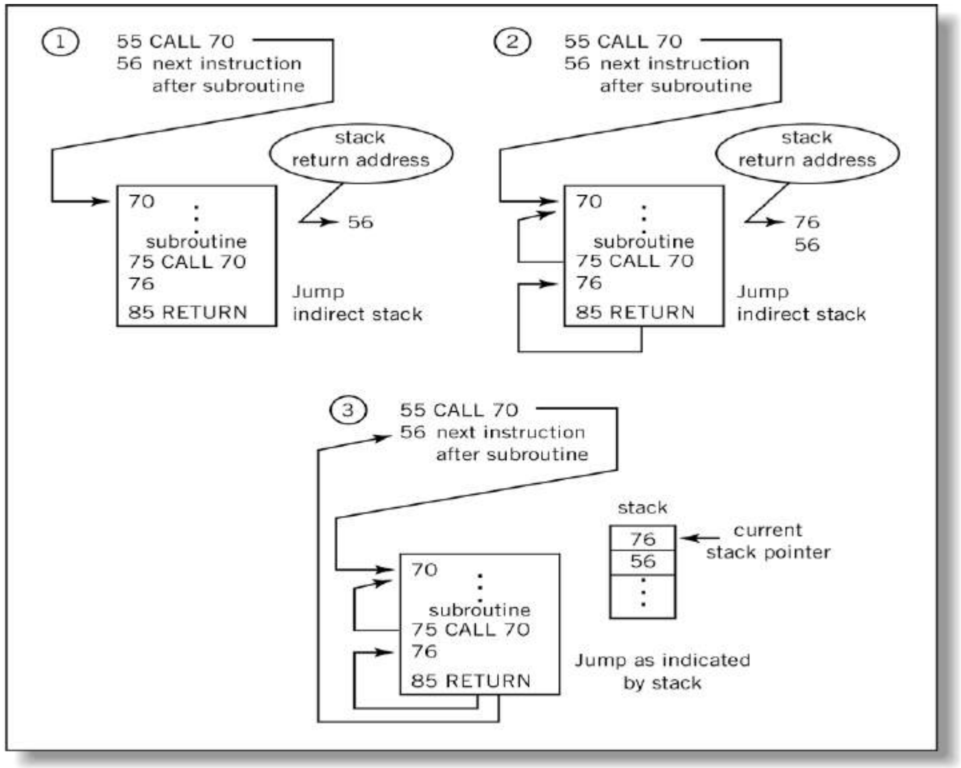
55 CALL 70
56 next instruction after subroutine
most recent return address
69
56
70
subroutine
85 RETURN
Jump to location indicated by 69
55 CALL 70
56 next instruction after subroutine
most recent return address
69
76
70
subroutine
75 CALL 70
76
85 RETURN
Jump to location indicated by 69

a. Subroutine called from loc.55

b. Subroutine re-called from 75

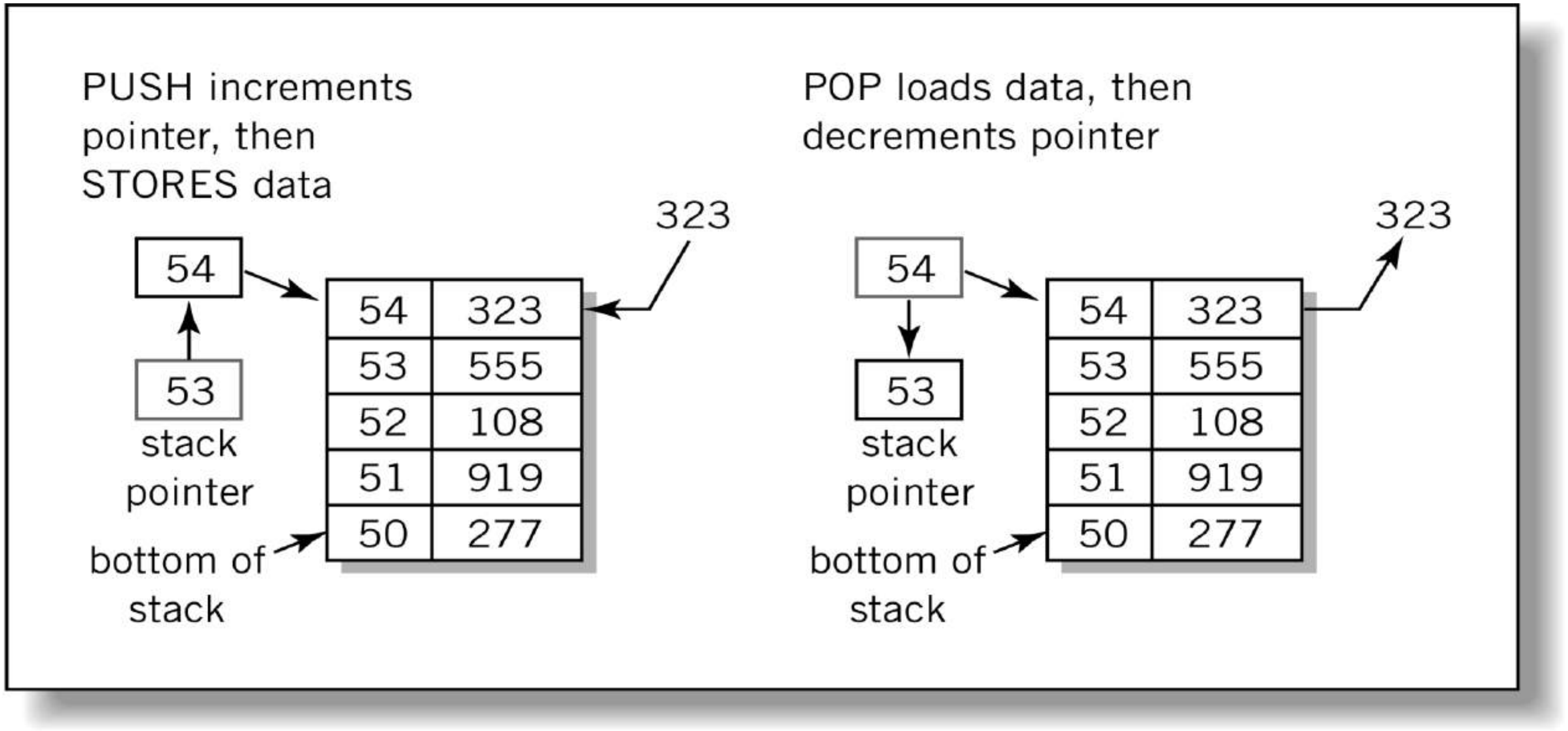
Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-20

Recursive Calls Stack Implementation



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-21

Stack Implementation



Englander: The Architecture of Computer Hardware and Systems Software, 2nd edition
Chapter 7, Figure 07-22