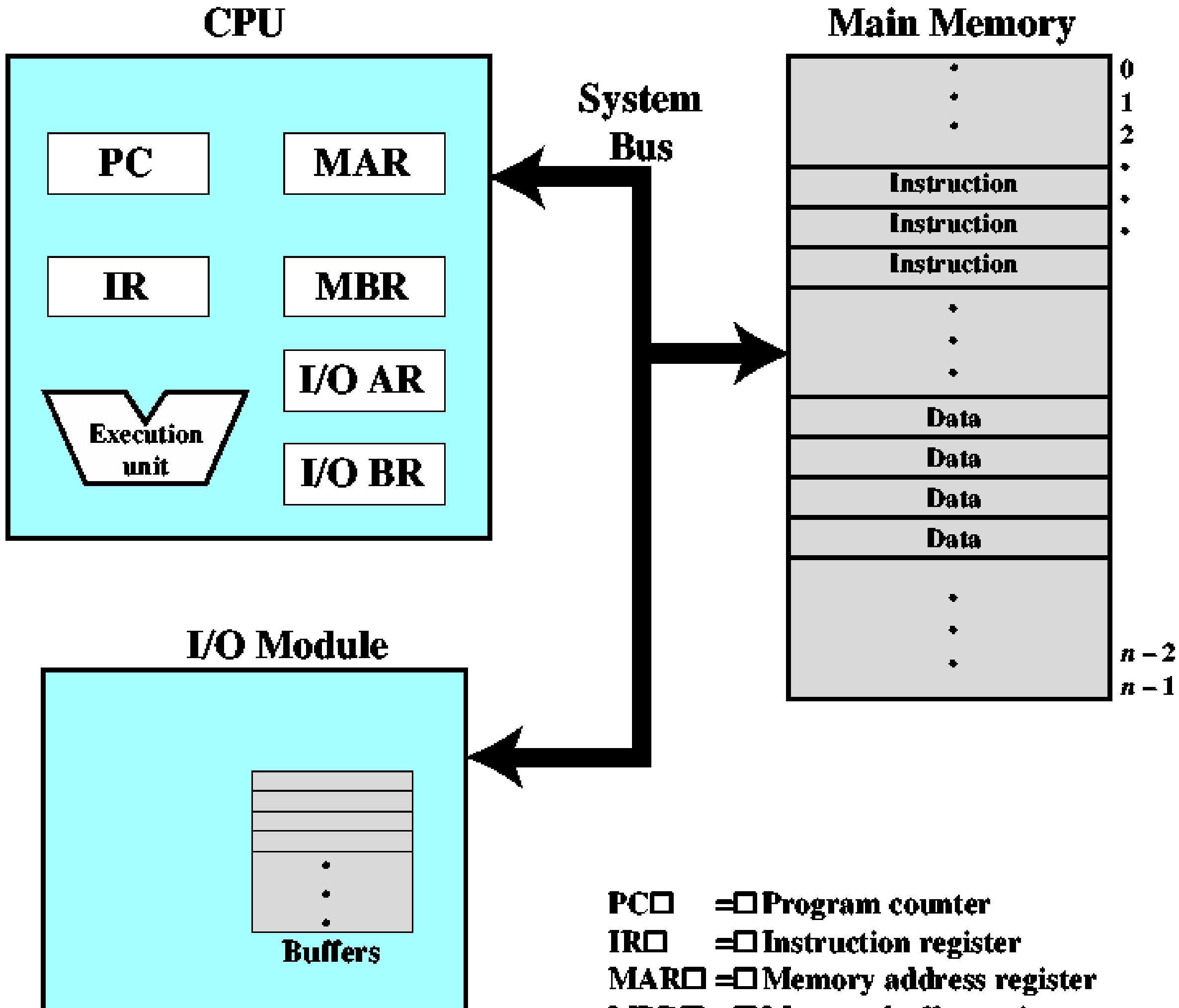
computer systems overview

slides tratte e adattate da
W. Stalling – Operating Systems: Internals and Design Principles

concetti fondamentali

- architettura di un clacolatore
  - architettura di un processore
  - linguaggio macchina
  - esecuzione di una istruzione
- chiamate di procedura e ritorno
  - uso dello stack
- interrupts
- gerarchie di memoria

Basic Elements

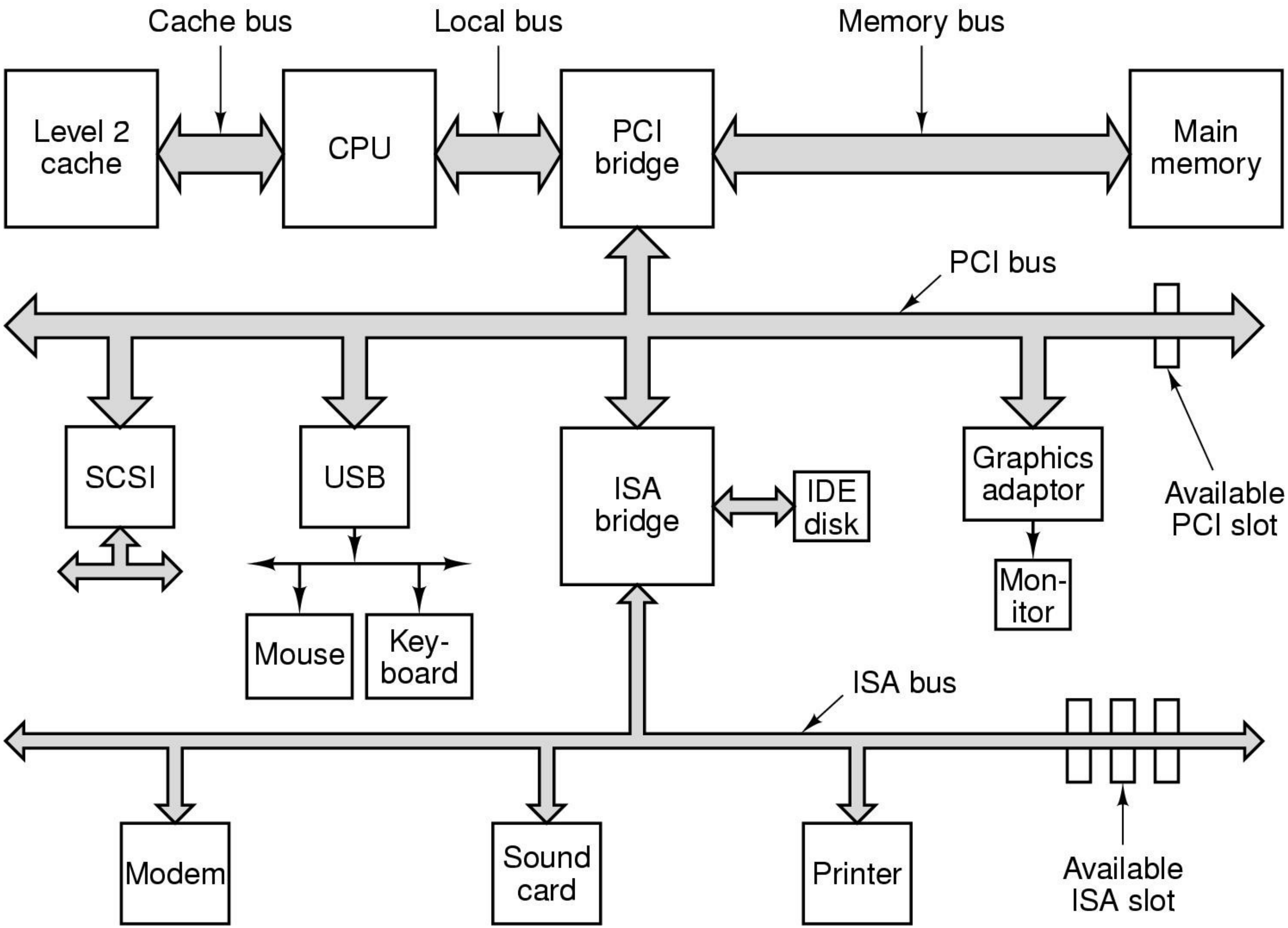


PC = Program counter
IR = Instruction register
MAR = Memory address register
MBR = Memory buffer register
I/O AR = Input/output address register
I/O BR = Input/output buffer register

not on the book

A (Simplified) Large Pentium System

da A.S. Tannenbaum



Instruction Execution

- Two steps
  - Processor reads instructions from memory
    - Fetches
  - Processor executes each instruction

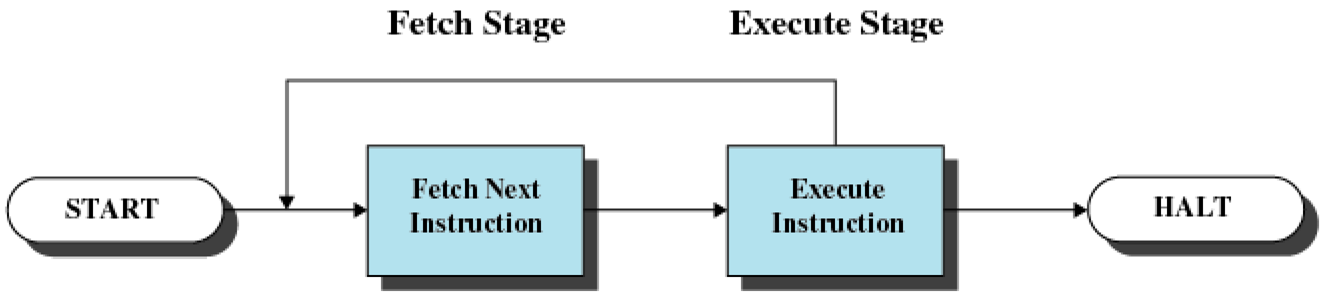


Figure 1.2 Basic Instruction Cycle

Instruction Categories

- Processor-memory
  - Transfer data between processor and memory
- Processor-I/O
  - Data transferred to or from a peripheral device
- Data processing
  - Arithmetic or logic operation on data
- Control
  - Alter sequence of execution

Characteristics of a Hypothetical Machine

0 – 3	4 – 15
Opcode	Address

(a) Instruction format

0	1 – 15
S	Magnitude

(b) Integer format

Program Counter (PC) = Address of instruction
Instruction Register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from Memory
0010 = Store AC to Memory
0101 = Add to AC from Memory

(d) Partial list of opcodes

Figure 1.3 Characteristics of a Hypothetical Machine

Example of Program Execution

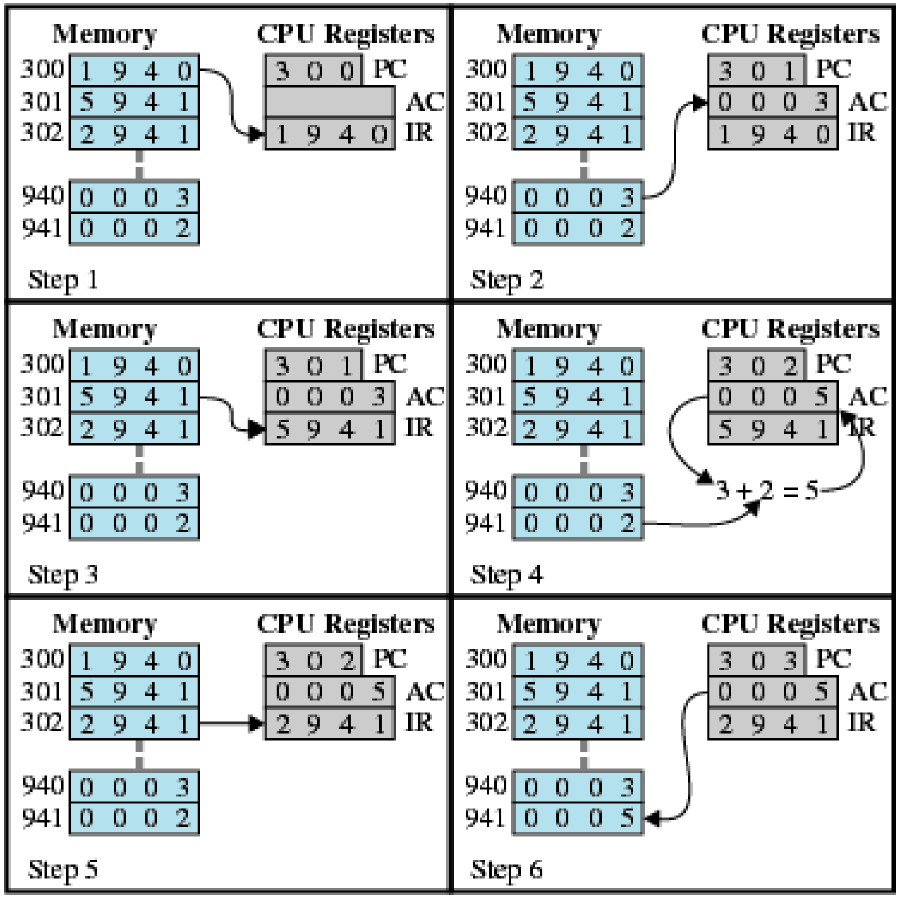


Figure 1.4 Example of Program Execution
(contents of memory and registers in hexadecimal)

Procedure Calls

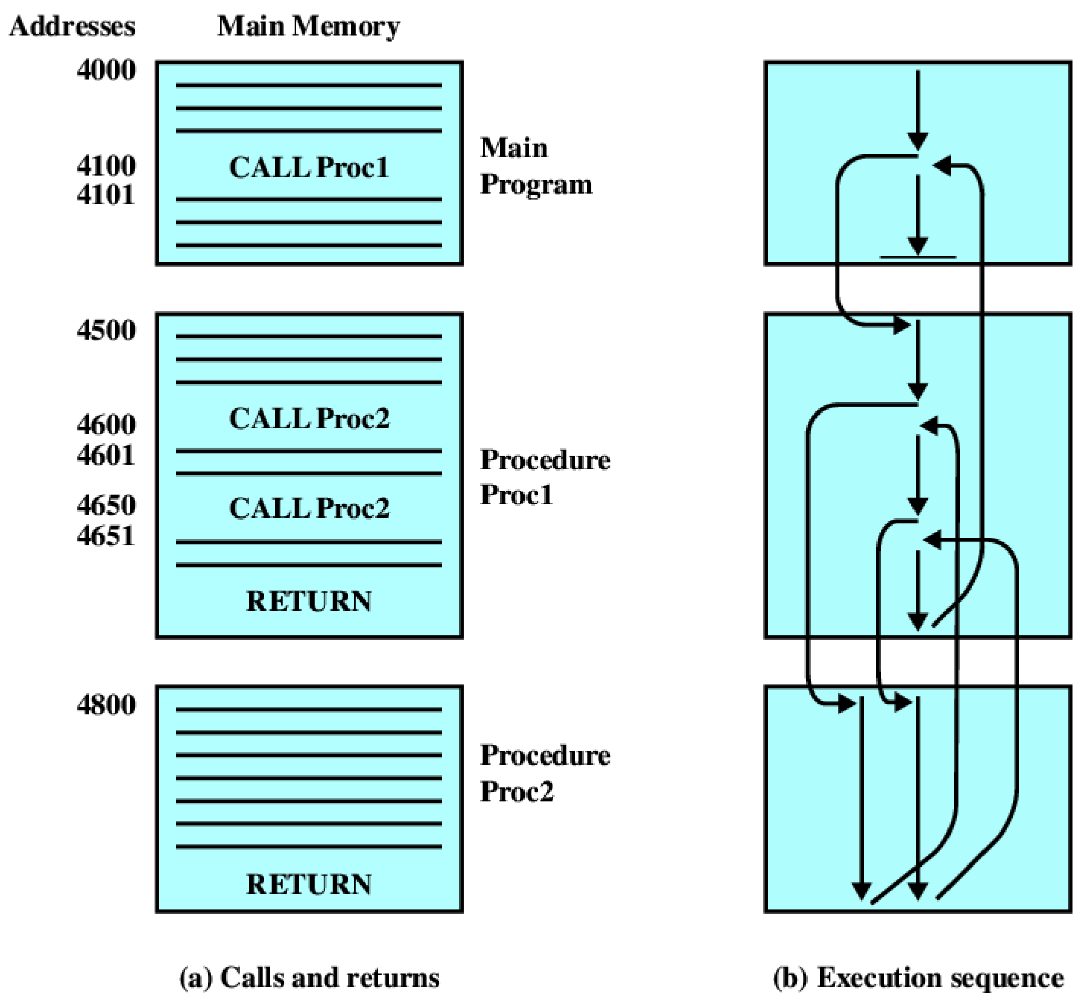
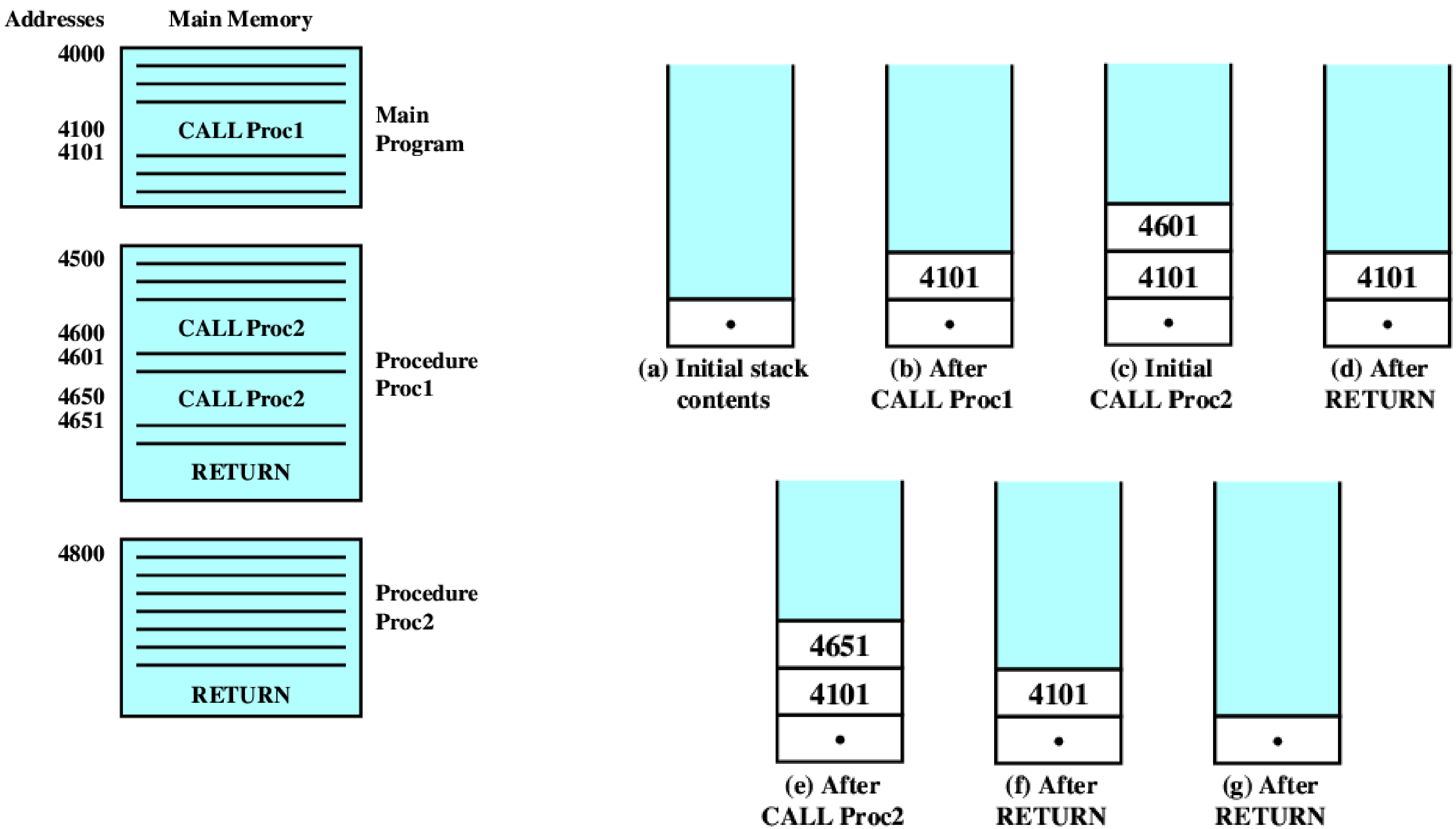


Figure 1.26 Nested Procedures

The Call Stack



CPU Registers for the Stack

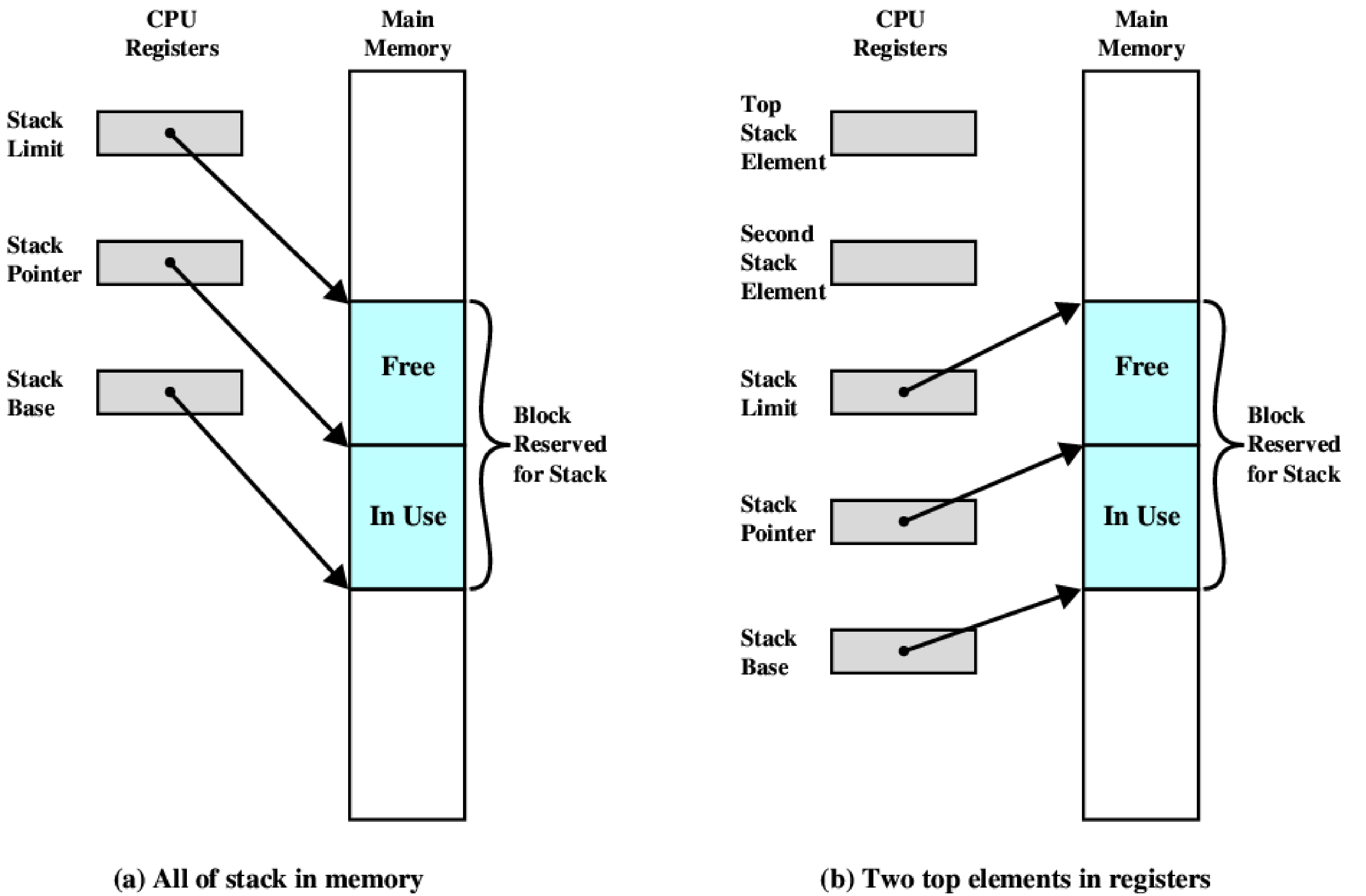
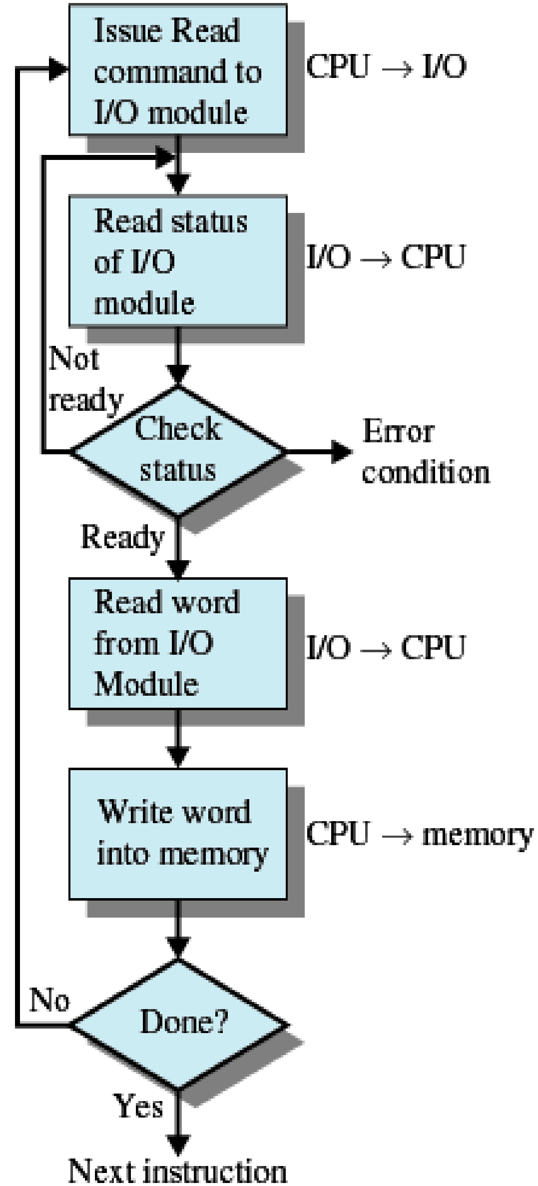


Figure 1.25 Typical Stack Organization

Programmed I/O

- processor drives I/O hw modules to perform I/O
- Sets appropriate bits in the I/O hw module status register
- Processor recursively checks status until operation is complete
- also known as “**busy waiting**”
- simple but inefficient
- used only for very very fast I/O devices
  – e.g. graphic



(a) Programmed I/O

interrupts

- most I/O devices are much slower than the processor
  - processor must pause would wait for device
  - better doing something else and being interrupted
- interrupts are **asynchronous** precedure calls
- interrupts are, usually, started by events that are
  - not related to the program that is currently executed by the processor
  - related to the program but not foreseen by the programmer

Classes of Interrupts

Table 1.1 Classes of Interrupts

Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.
I/O	Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.
Hardware failure	Generated by a failure, such as power failure or memory parity error.

Interrupt Handler

- the **interrupt handler** is the procedure called when an interrupt occours
- after that the processor execution flow and state are restored

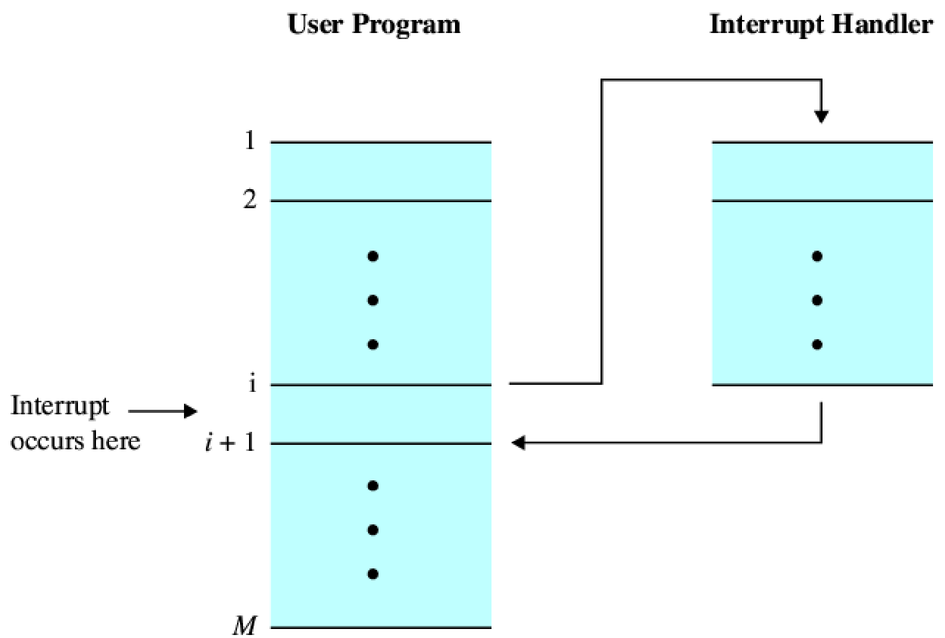


Figure 1.6 Transfer of Control via Interrupts

Interrupt Handler

- generally part of the operating system
- it knows or discovers the reason for the interrupt
- it handle the situation that caused the interrupt

Interrupt Cycle

- interrupts never interrupt the execution of a machine instruction
  - i.e., checks for pending interrupts is performed by the processor after each machine instruction

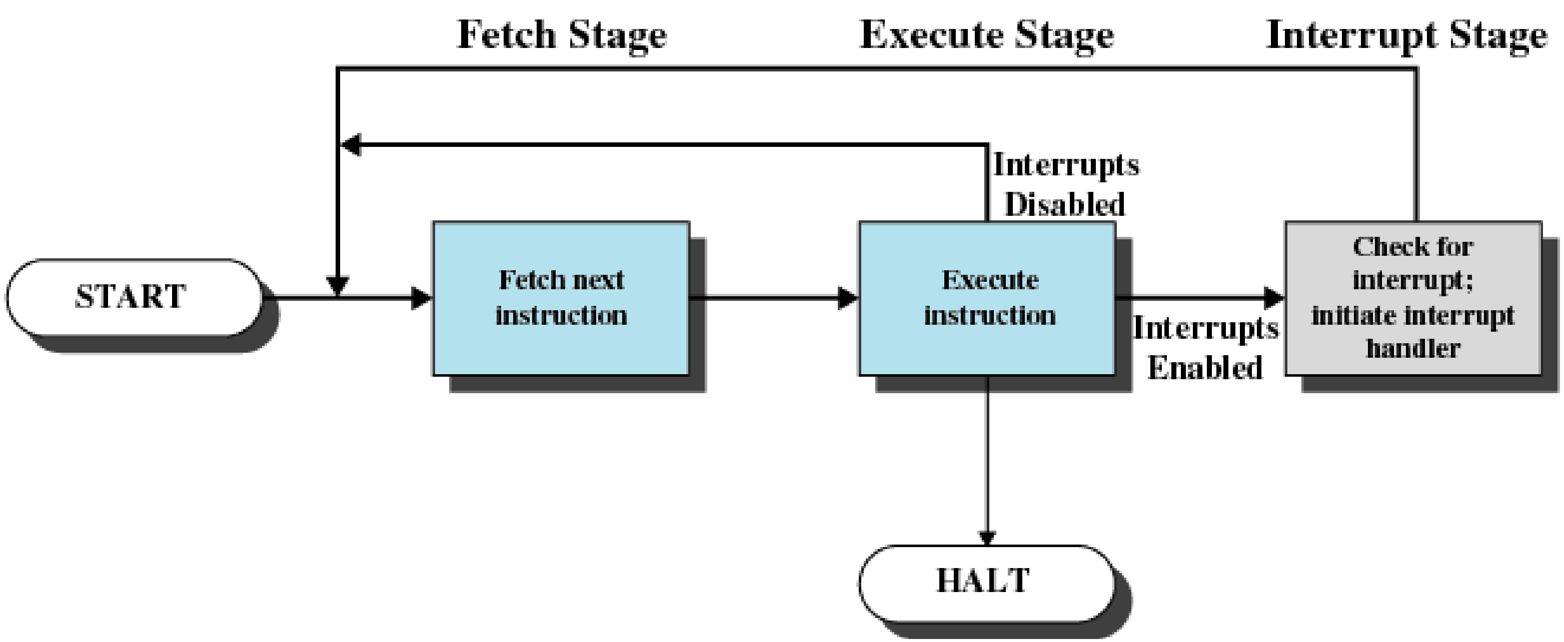


Figure 1.7 Instruction Cycle with Interrupts

Changes in Memory and Registers for an Interrupt

addresses grow

M=size of registers

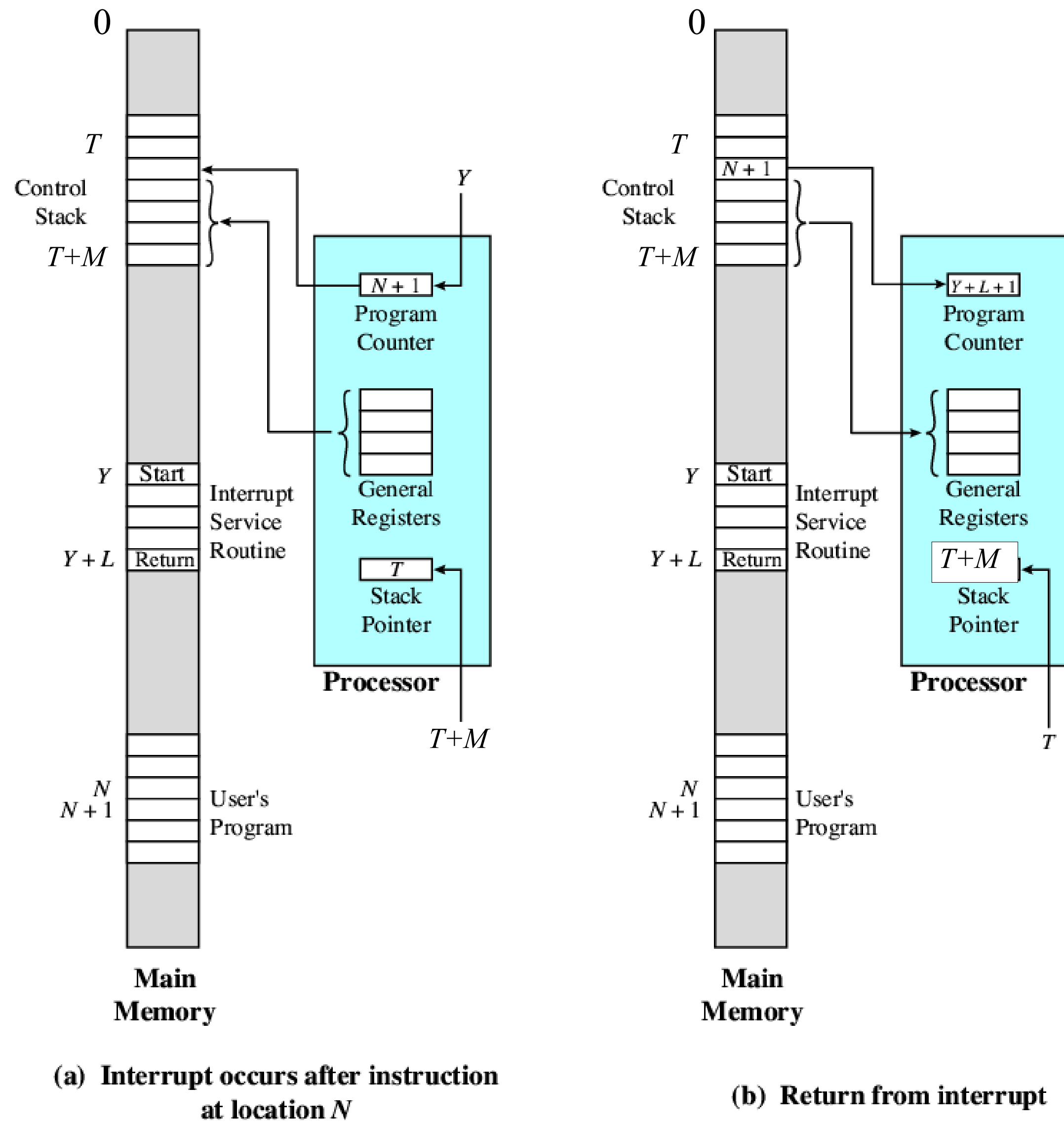


Figure 1.11 Changes in Memory and Registers for an Interrupt

Simple Interrupt Processing

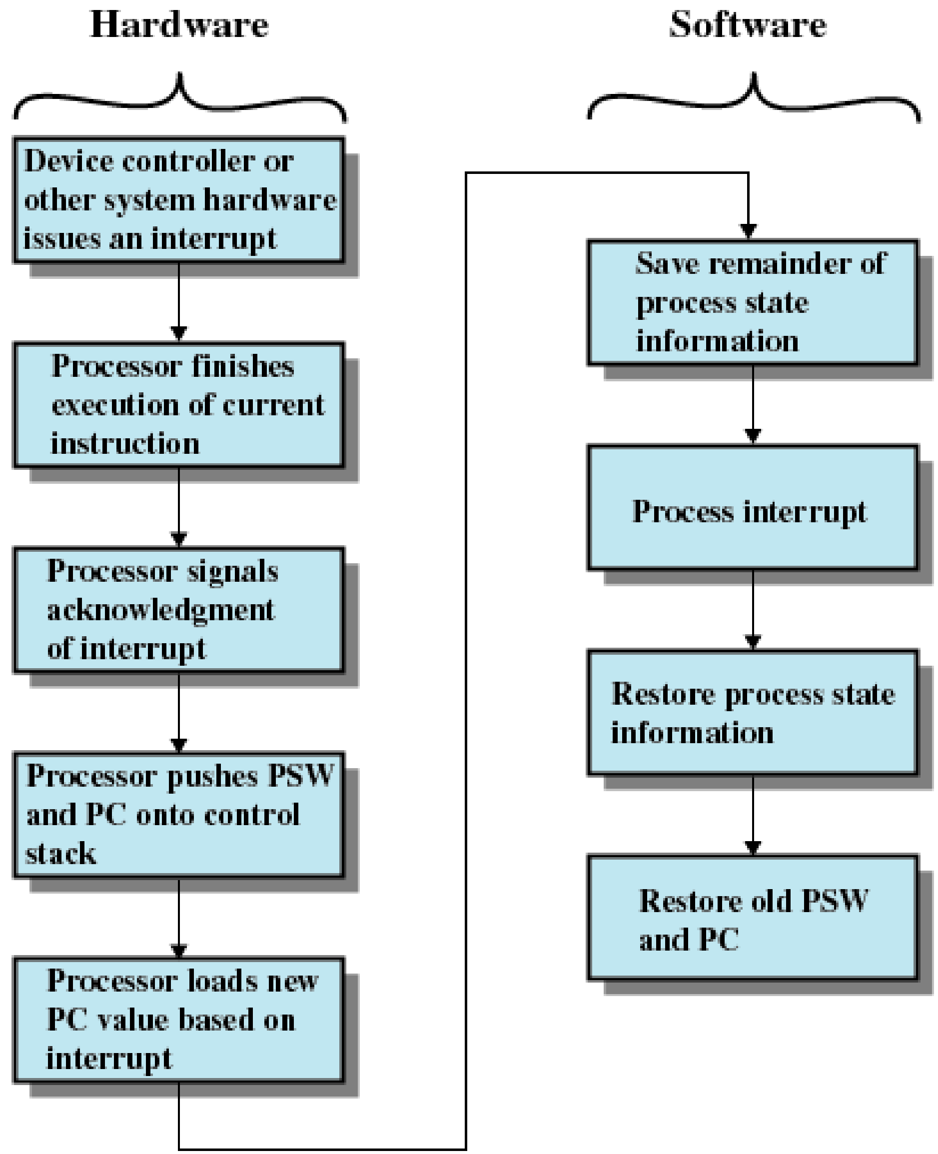
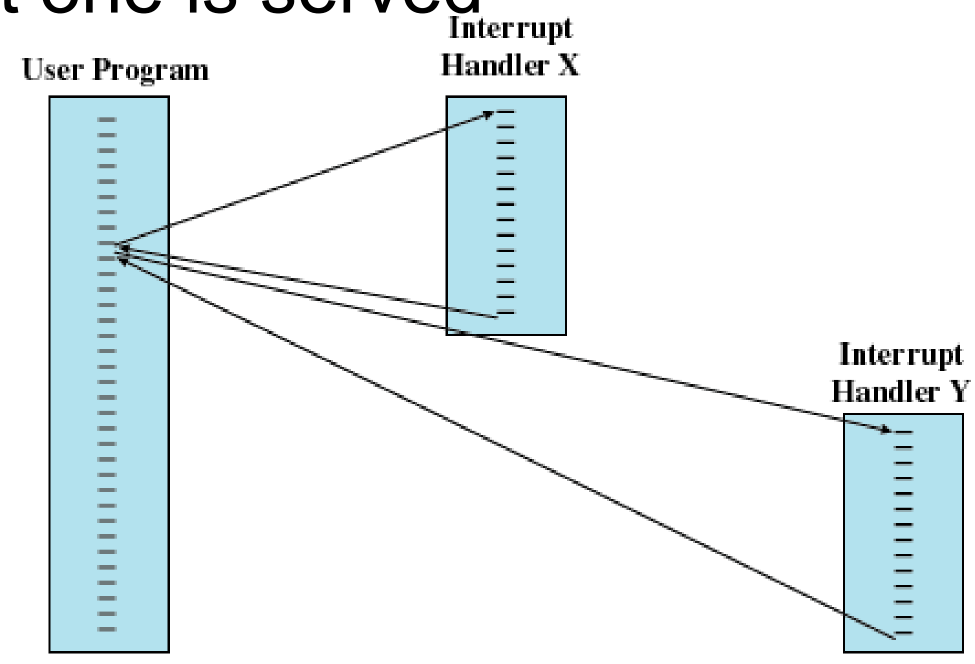


Figure 1.10 Simple Interrupt Processing

Multiple Interrupts

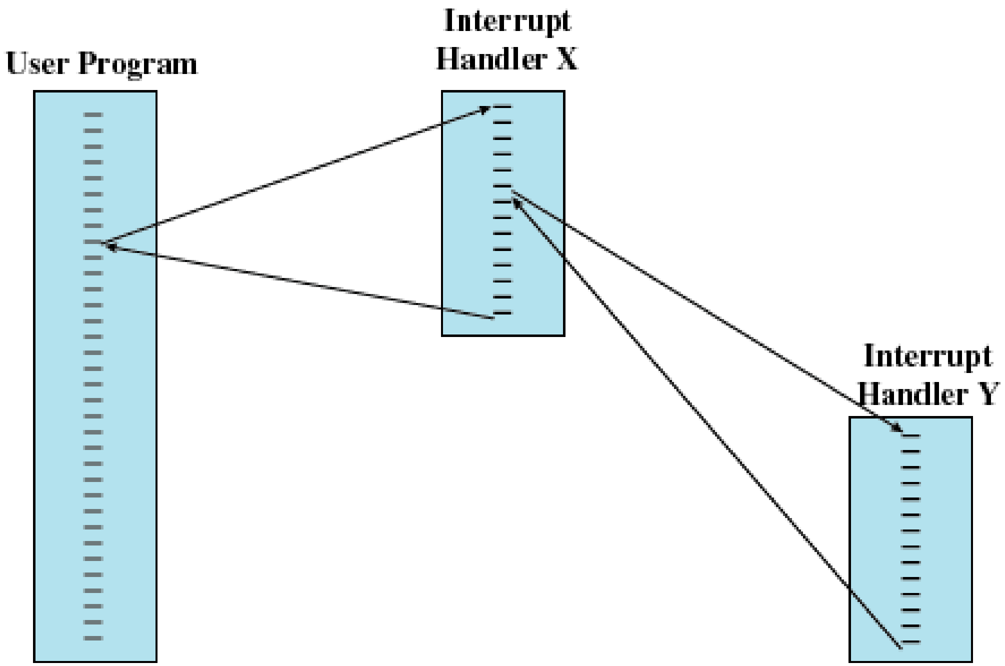
- what if an interrupt is generated during the execution of the interrupt handler?
- one solution: disable interrupts during interrupt handler execution
  - the pending interrupts will be executed after the current one is served



(a) Sequential interrupt processing

nested interrupt processing

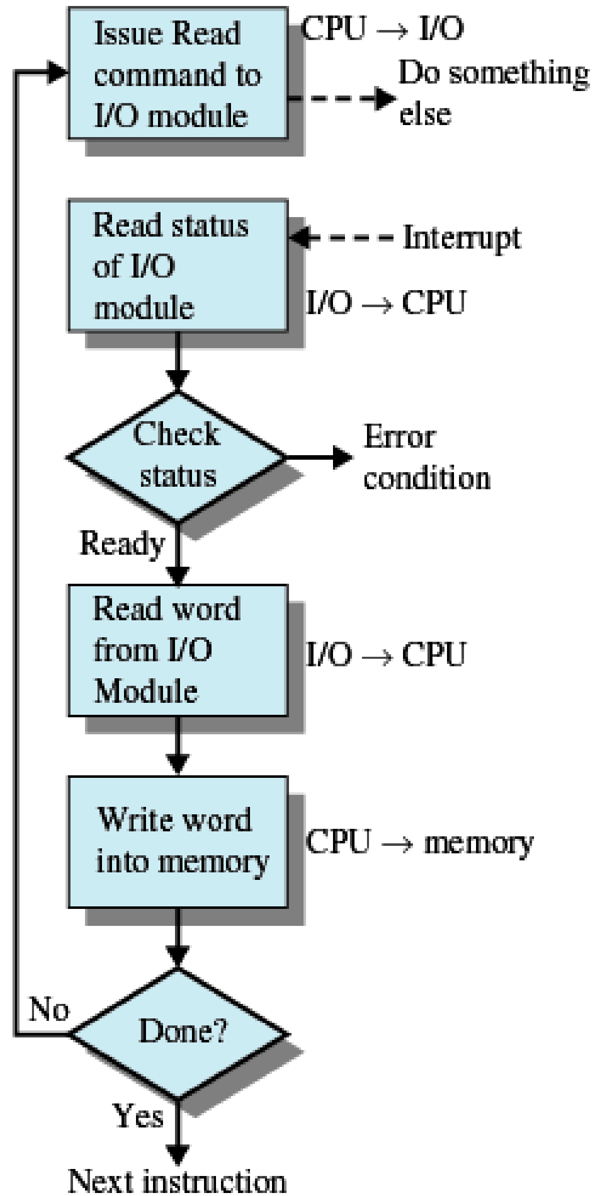
- define priorities for interrupts
- a high priority interrupt can interrupt an interrupt handler that is serving an interrupt at a lower priority



(b) Nested interrupt processing

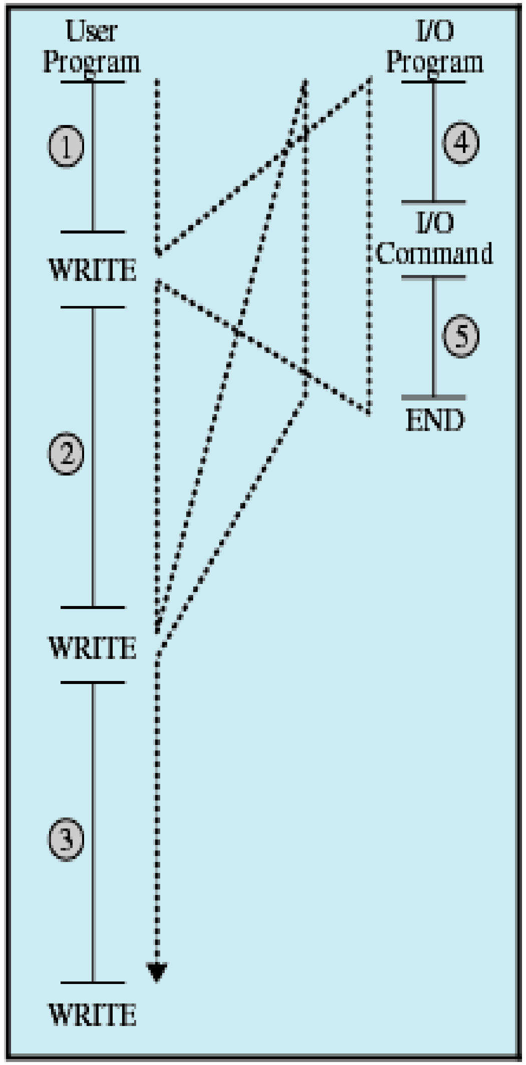
Interrupt-Driven I/O

- Processor is interrupted when I/O module ready to exchange data
- No needless waiting
- Processor saves context of program executing and begins executing interrupt-handler

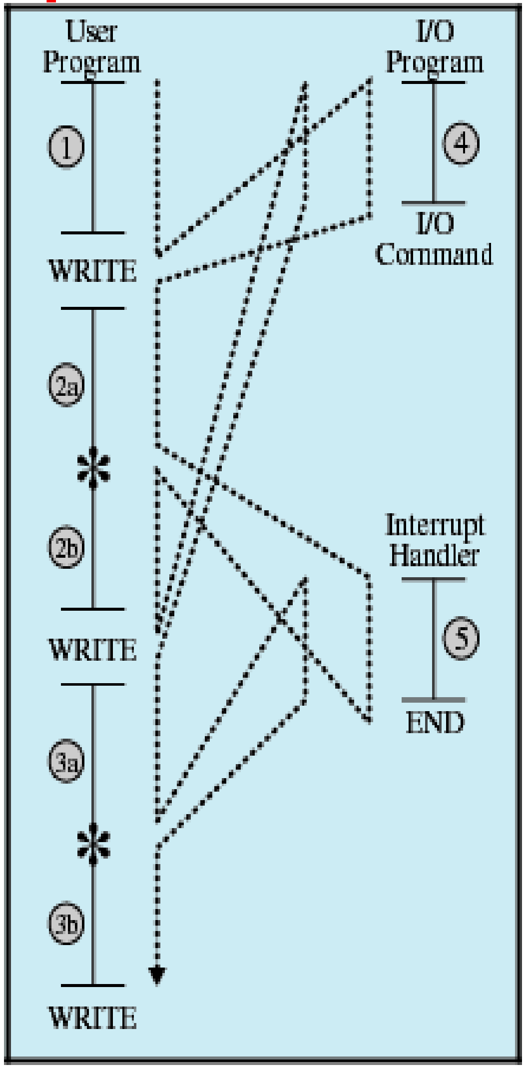


(b) Interrupt-driven I/O

I/O With and Without Interrupts

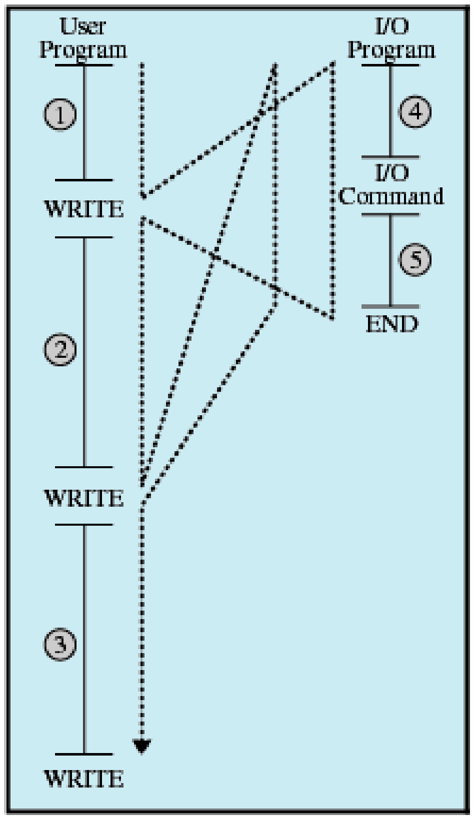


(a) No interrupts

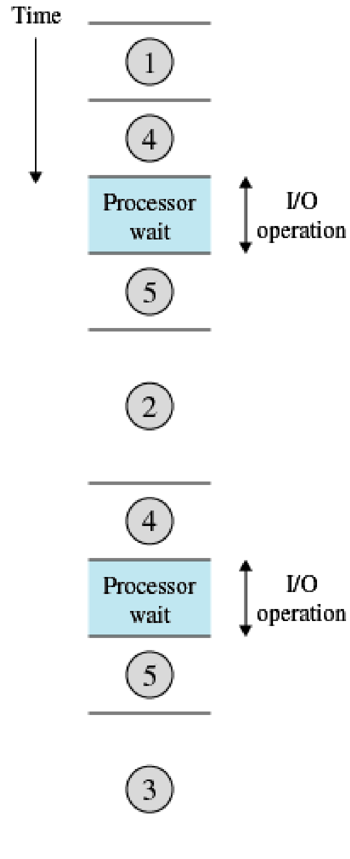


(b) Interrupts; short I/O wait

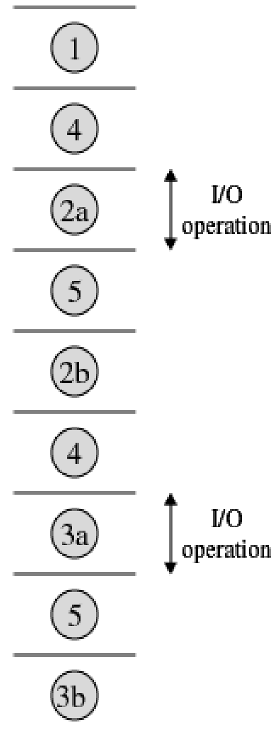
Timing Diagram Based on Short I/O Wait



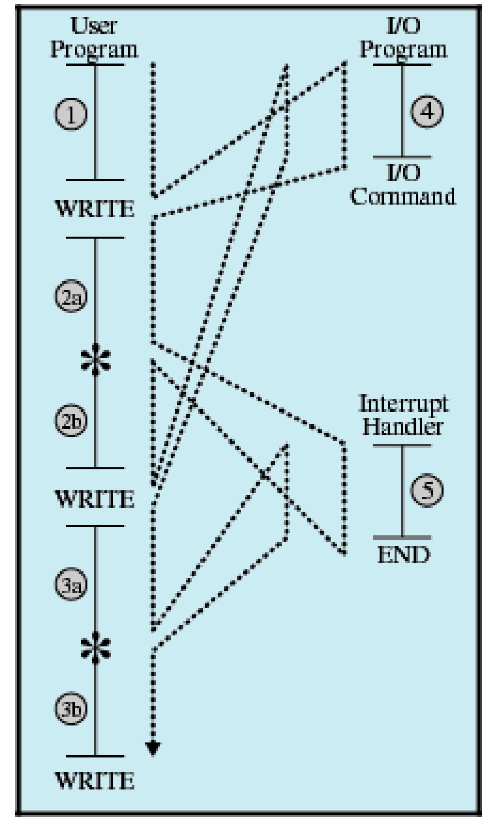
(a) No interrupts



(a) Without interrupts (circled numbers refer to numbers in Figure 1.5a)

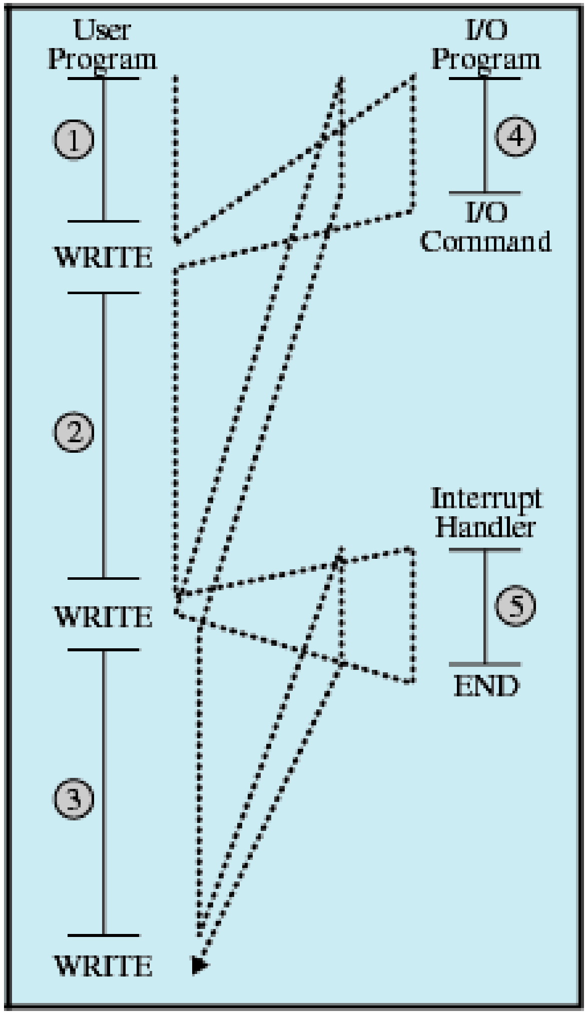


(b) With interrupts (circled numbers refer to numbers in Figure 1.5b)



(b) Interrupts; short I/O wait

long I/O wait and Interrupts

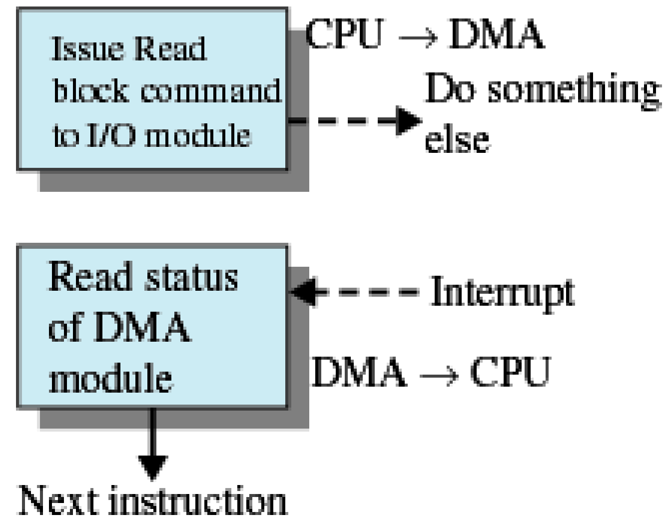


(c) Interrupts; long I/O wait

- the most efficient and general approach
- it needs I/O queues (buffers)
- however, an interrupt for each transferred word si very inefficient
  - cpu would be very busy in serving interrupts for doing I/O

Direct Memory Access

- I/O to/from memory is performed by a special purpose chip (DMA controller)
- Moderated CPU slowdown
  - setup time
  - shared bus
- An interrupt is sent when the transfer is complete
- Processor continues with other work



(c) Direct memory access

Memory Hierarchy

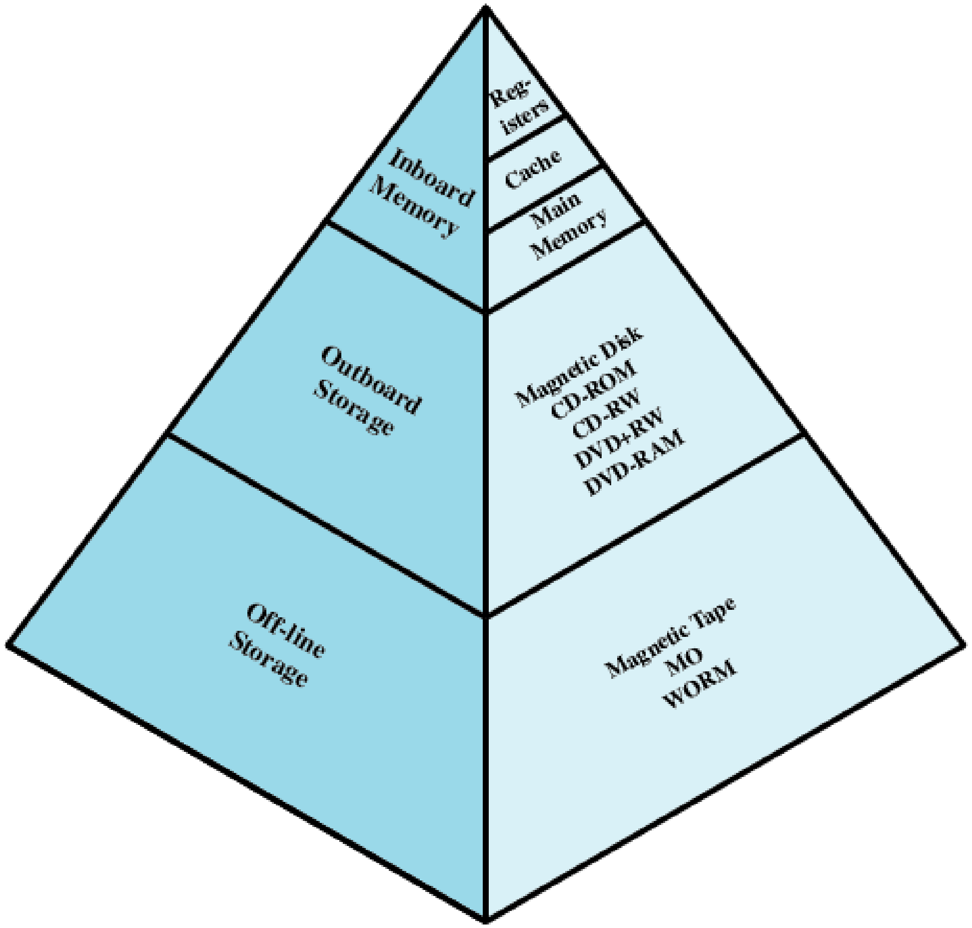


Figure 1.14 The Memory Hierarchy

- Faster access time, greater cost per bit
- Greater capacity
  - smaller cost per bit
  - slower access speed
- Based on Locality
  - temporal
  - spatial

Disk Cache

- A portion of main memory used as a buffer to temporarily to hold data for the disk
- Disk writes are clustered
- Some data written out may be referenced again. The data are retrieved rapidly from the software cache instead of slowly from disk