Principles of Operating Systems

Abhishek Dubey

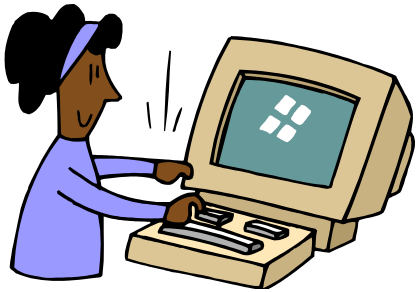
Slides Based On Power points and book material from William Stallings

Fall 2015

Basic Elements of a Computer

Processor

I/O Modules



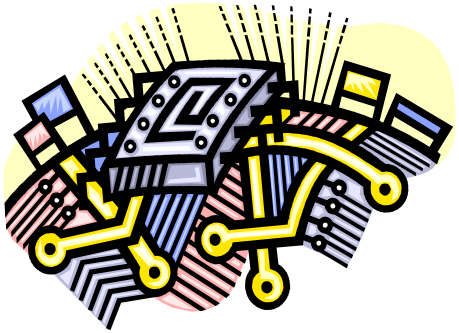
Main Memory

System Bus

Processor

Controls the operation of the computer

Performs the data processing functions



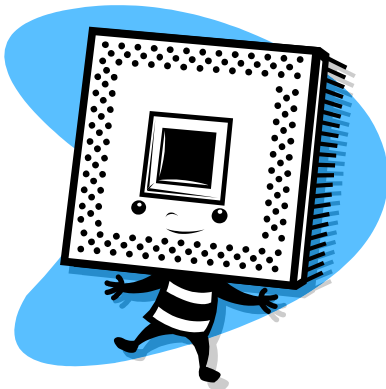
Referred to as the *Central Processing Unit* (CPU)

Main Memory

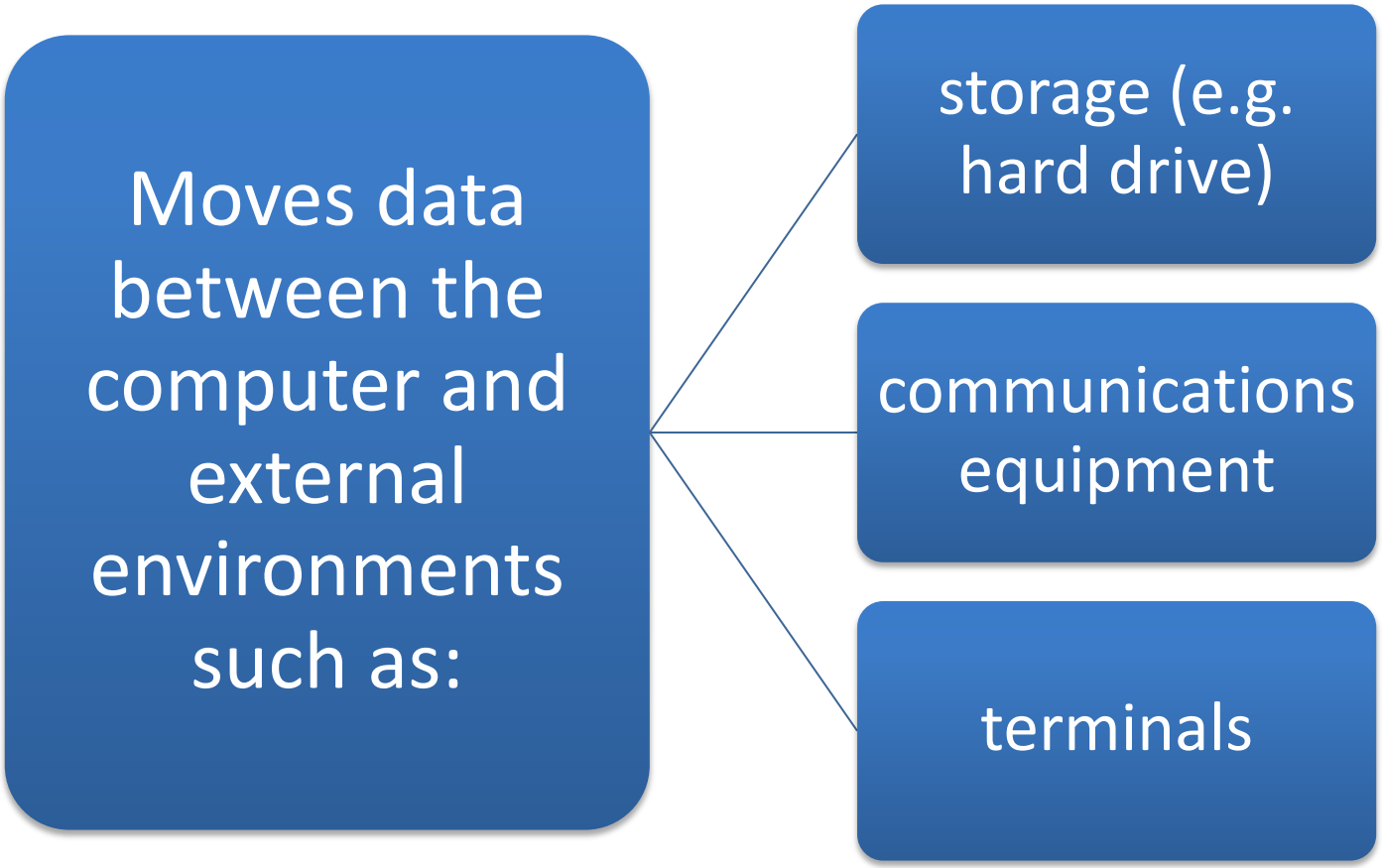
- Volatile

–Contents of the memory is lost when the computer is shut down

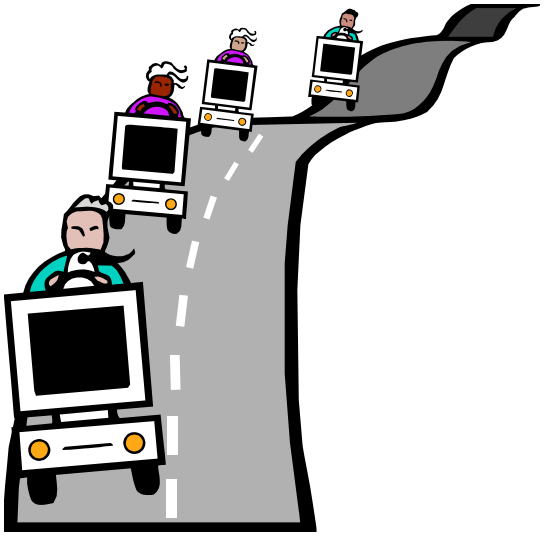
- Referred to as real memory or primary memory



I/O Modules

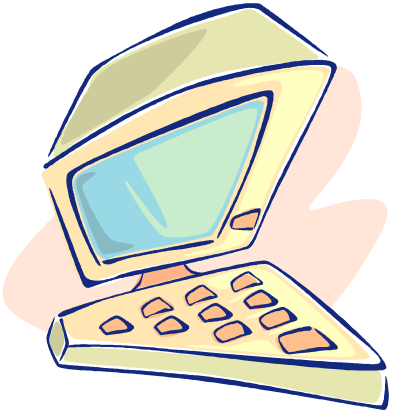

Moves data between the computer and external environments such as:
storage (e.g. hard drive)
communications equipment
terminals

System Bus



– Provides for communication among processors, main memory, and I/O modules

A program consists of a set of instructions stored in memory



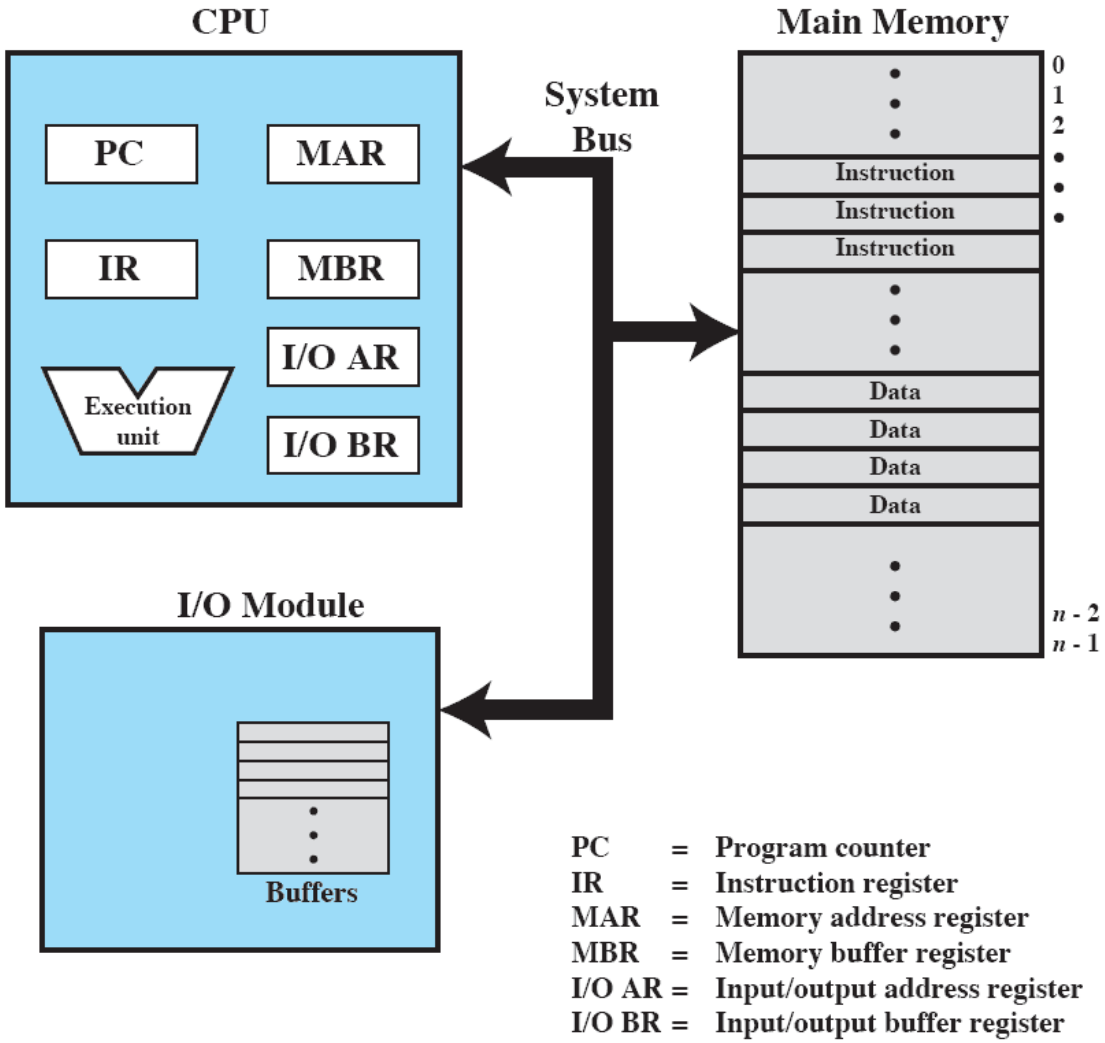


Figure 1.1 Computer Components: Top-Level View

Basic Instruction Cycle

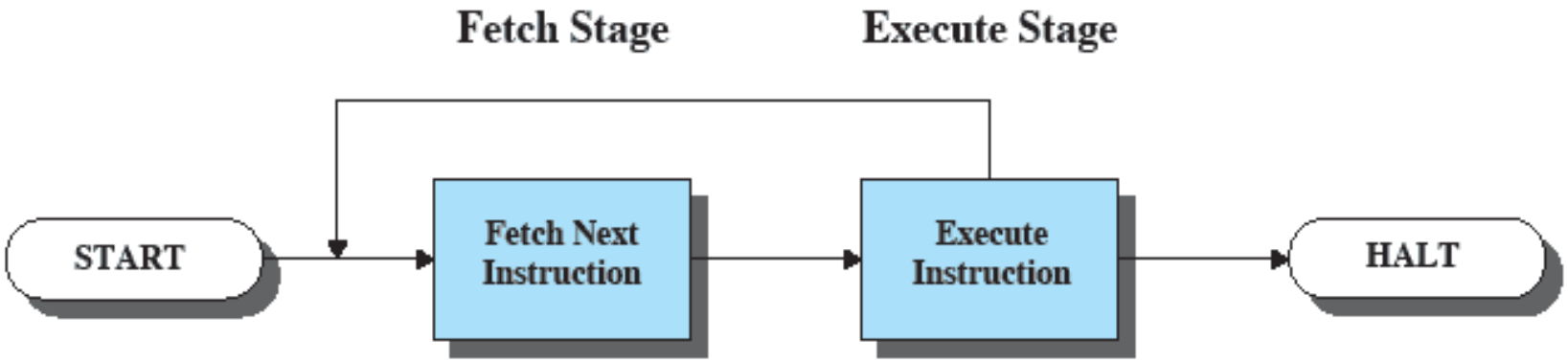
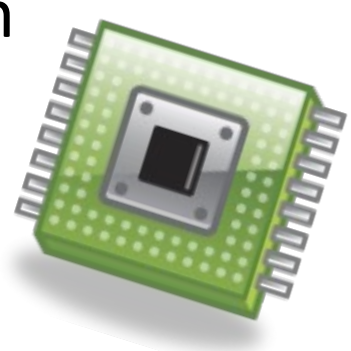


Figure 1.2 Basic Instruction Cycle

Instruction Fetch and Execute

- The processor fetches the instruction from memory
- Program counter (PC) holds address of the instruction to be fetched next
  - PC is incremented after each fetch



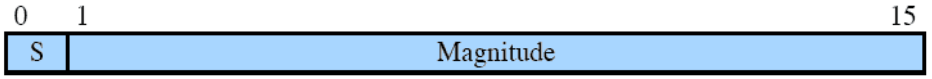
Instruction Register (IR)

Fetched instruction is loaded into Instruction Register (IR)

- Processor interprets the instruction and performs required action:
  - Processor-memory
  - Processor-I/O
  - Data processing
  - Control

Characteristics of a Hypothetical Machine

(a) Instruction format



(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

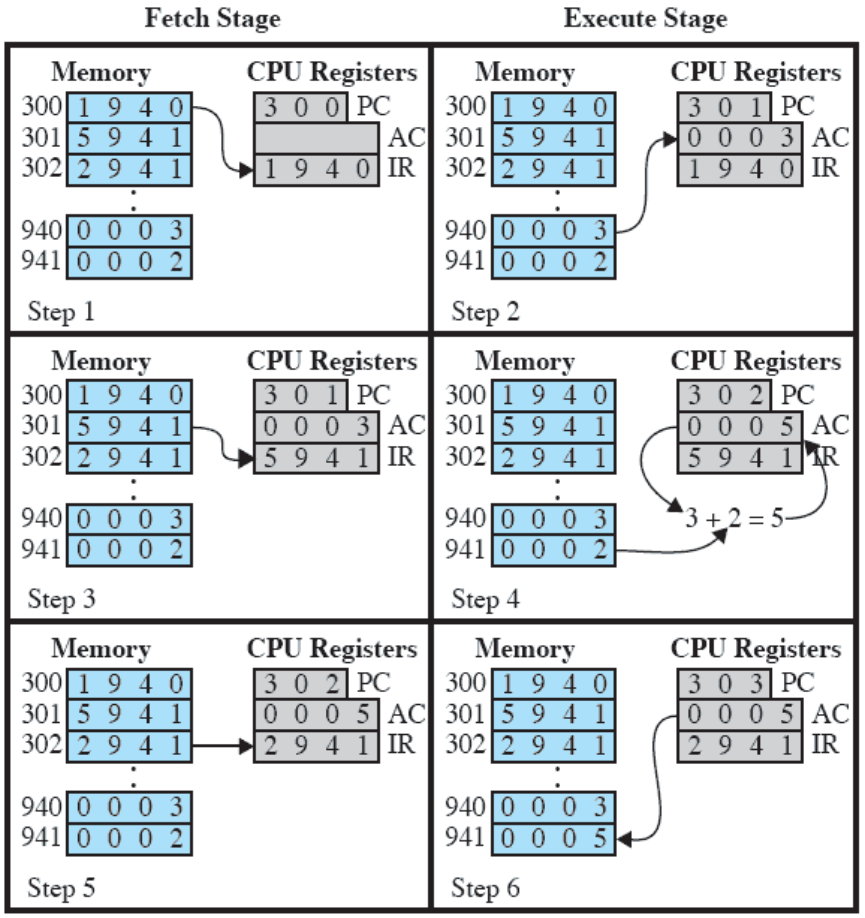


Figure 1.4 Example of Program Execution (contents of memory and registers in hexadecimal)

Interrupts

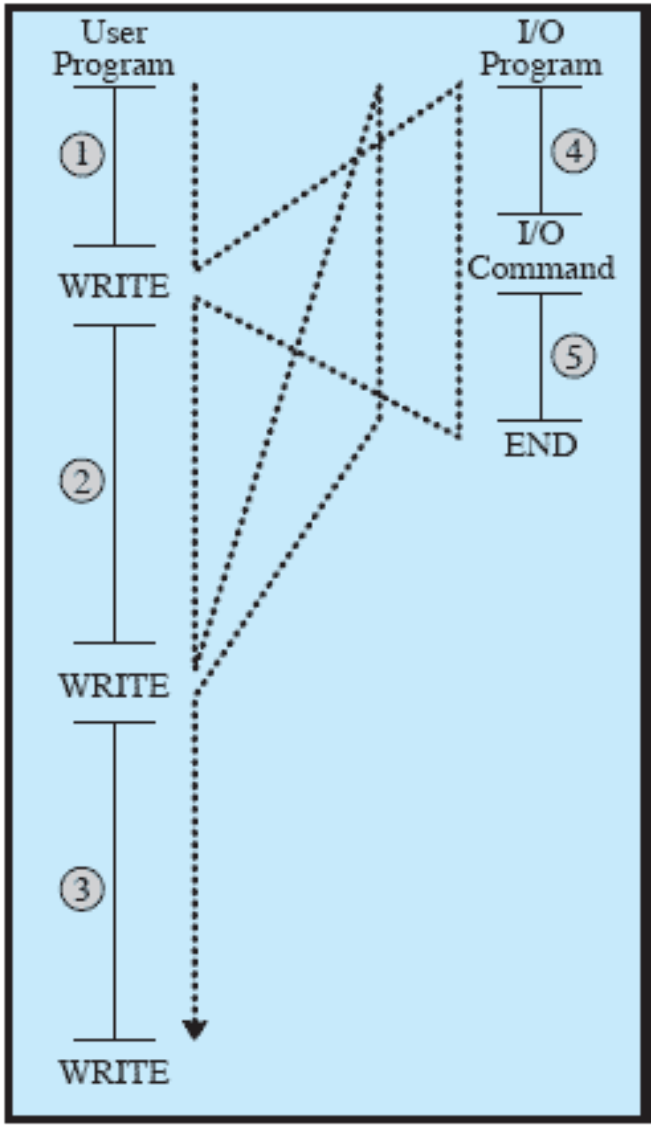
- Interrupt the normal sequencing of the processor
- Provided to improve processor utilization
  - most I/O devices are slower than the processor
  - processor must pause to wait for device
  - wasteful use of the processor

Common Classes of Interrupts

Table 1.1 Classes of Interrupts

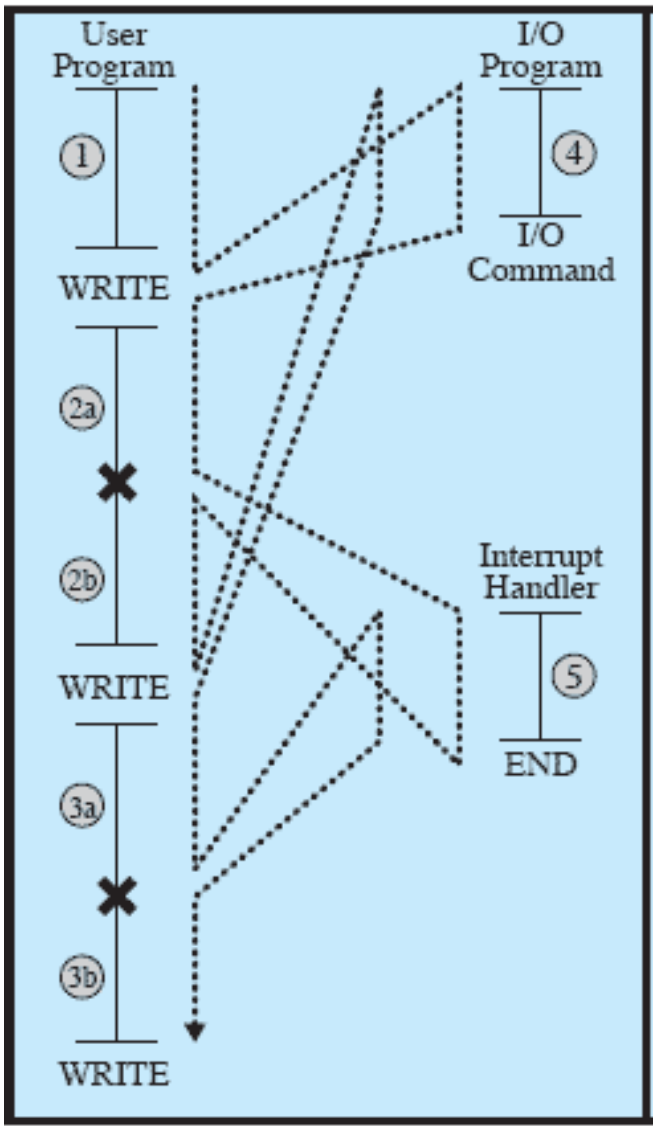
Program	Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.
Timer	Generated by a timer within the processor. This allows 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.

Flow of Control Without Interrupts



(a) No interrupts

Interrupts: Short I/O Wait



(b) Interrupts; short I/O wait

Transfer of Control via Interrupts

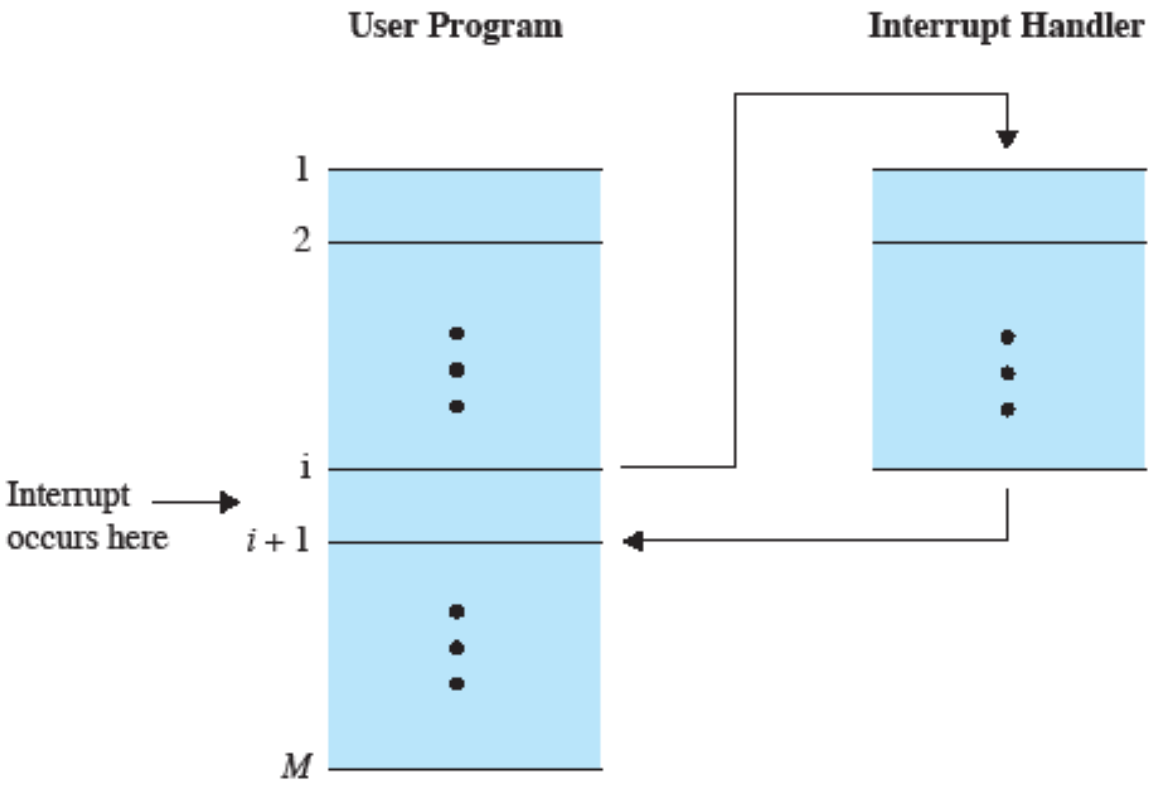


Figure 1.6 Transfer of Control via Interrupts

Instruction Cycle With Interrupts

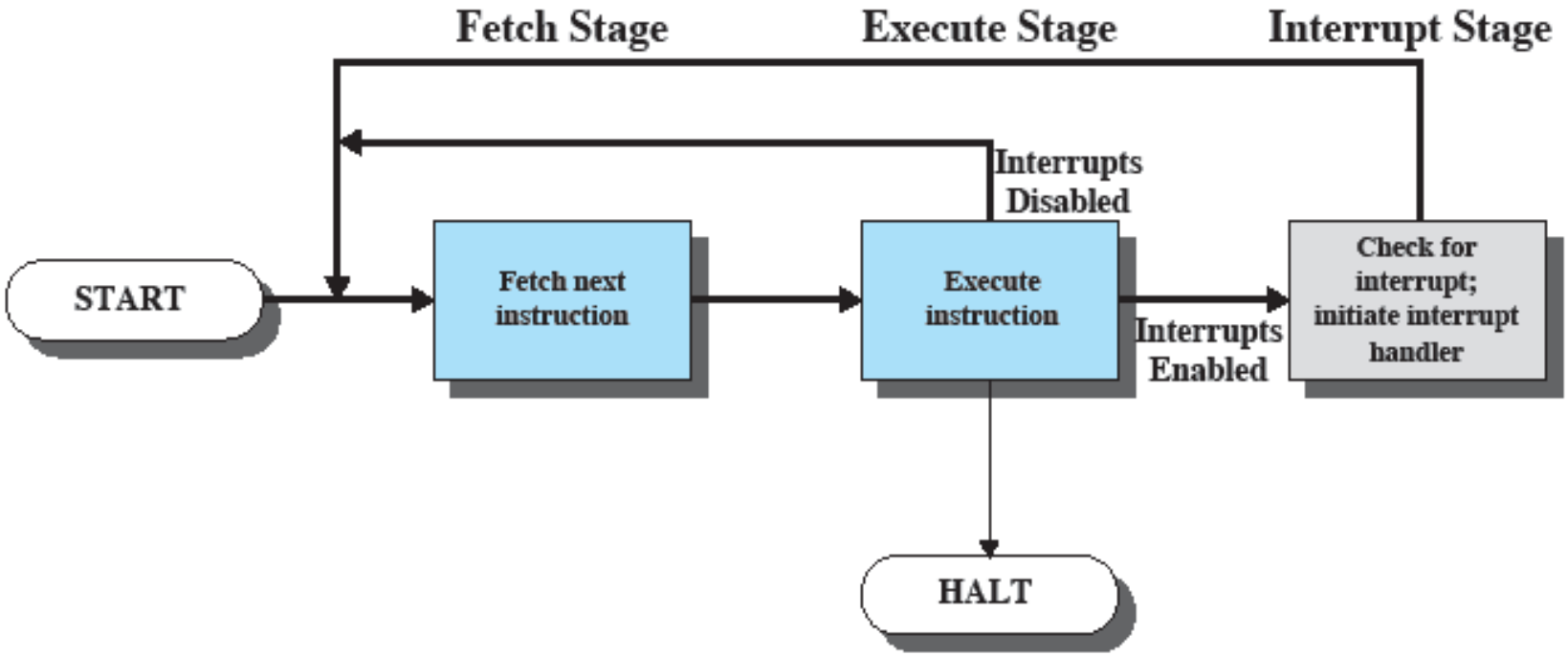
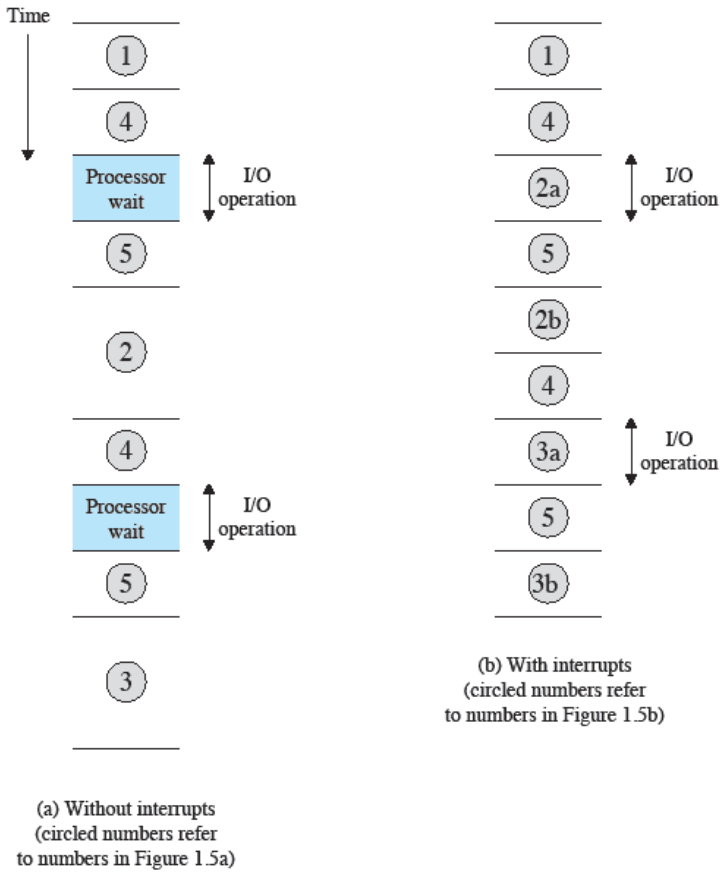


Figure 1.7 Instruction Cycle with Interrupts

Program Timing: Short I/O Wait

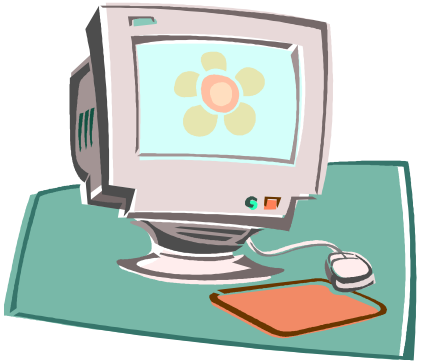


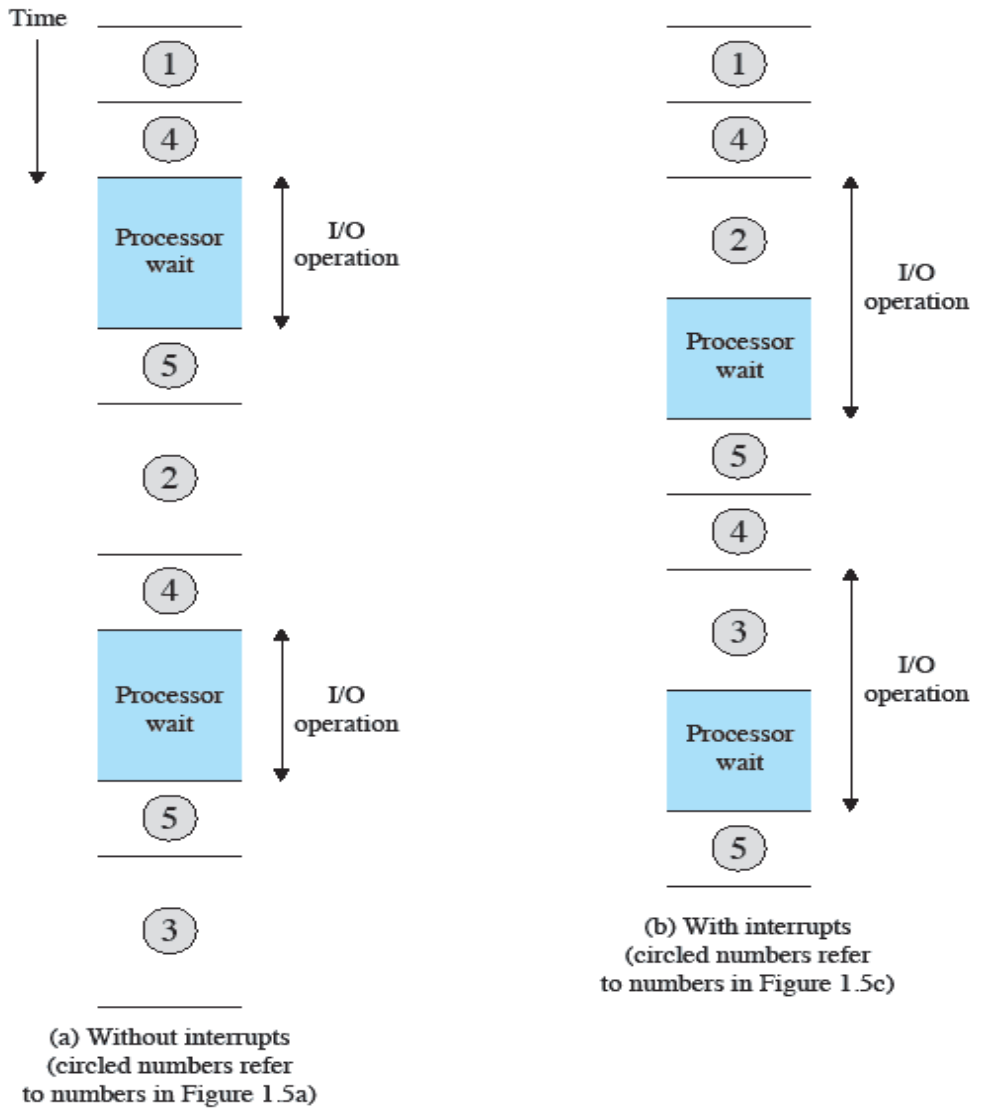
(a) Without interrupts (circled numbers refer to numbers in Figure 1.5a)

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

Figure 1.8 Program Timing: Short I/O Wait

Program Timing: Long I/O wait



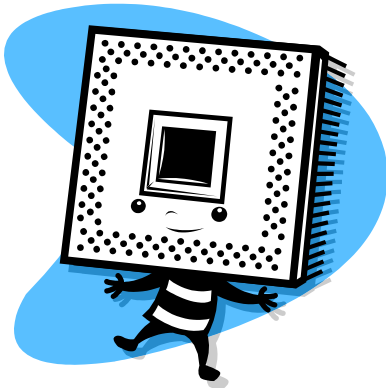


(a) Without interrupts (circled numbers refer to numbers in Figure 1.5a)

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

Figure 1.9 Program Timing: Long I/O Wait

Simple Interrupt Processing



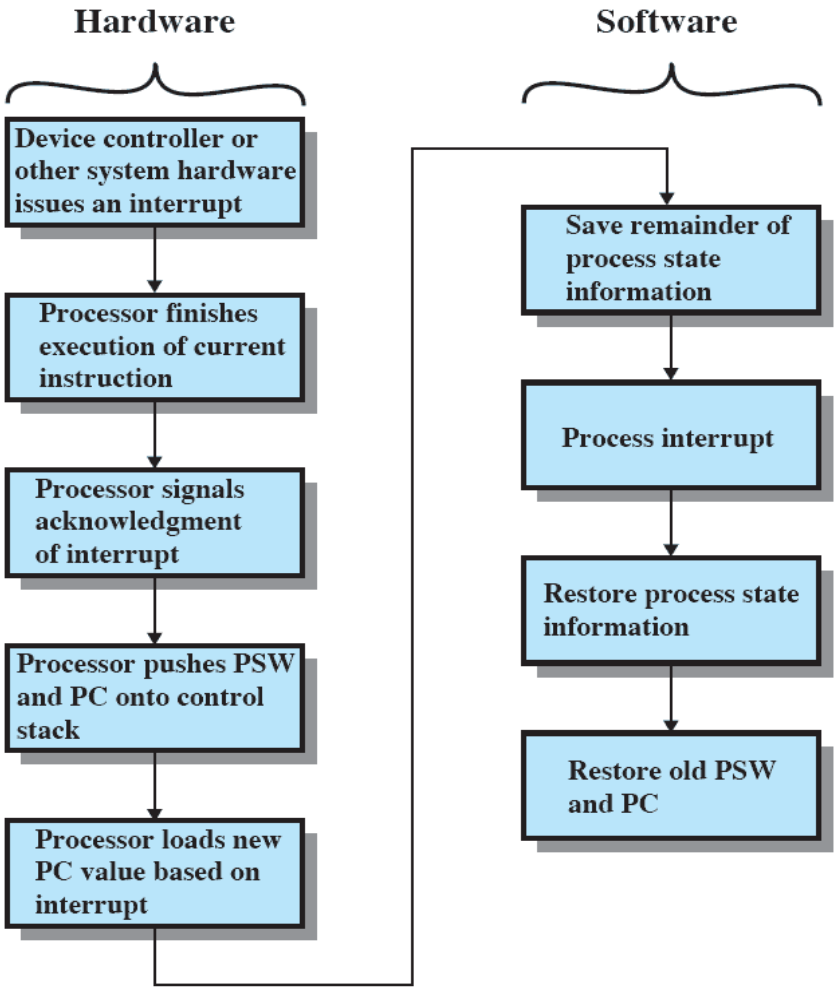
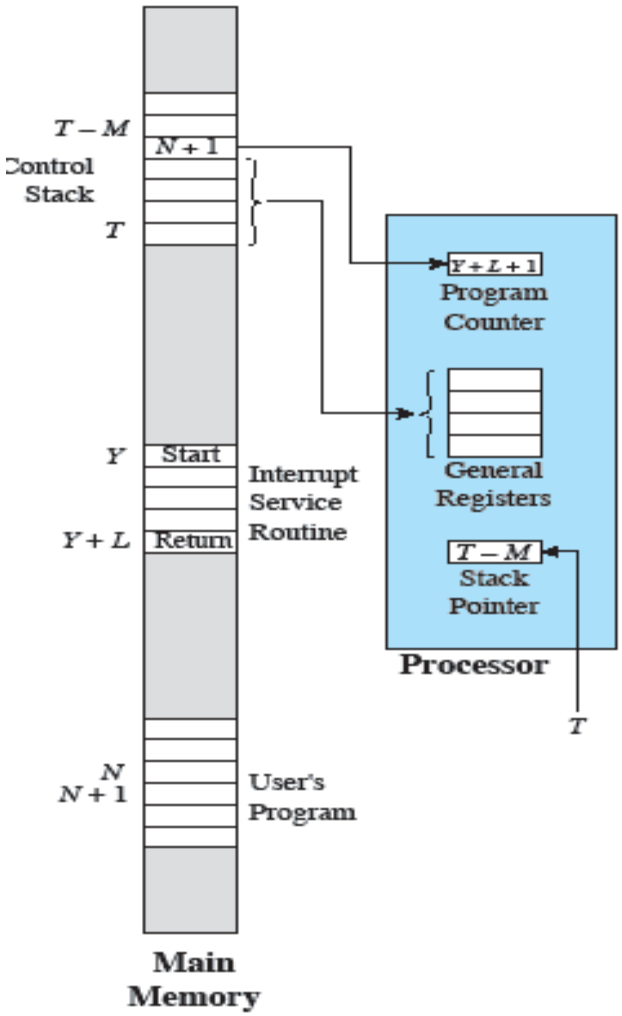


Figure 1.10 Simple Interrupt Processing



(b) Return from interrupt

Changes for an Interrupt

Multiple Interrupts

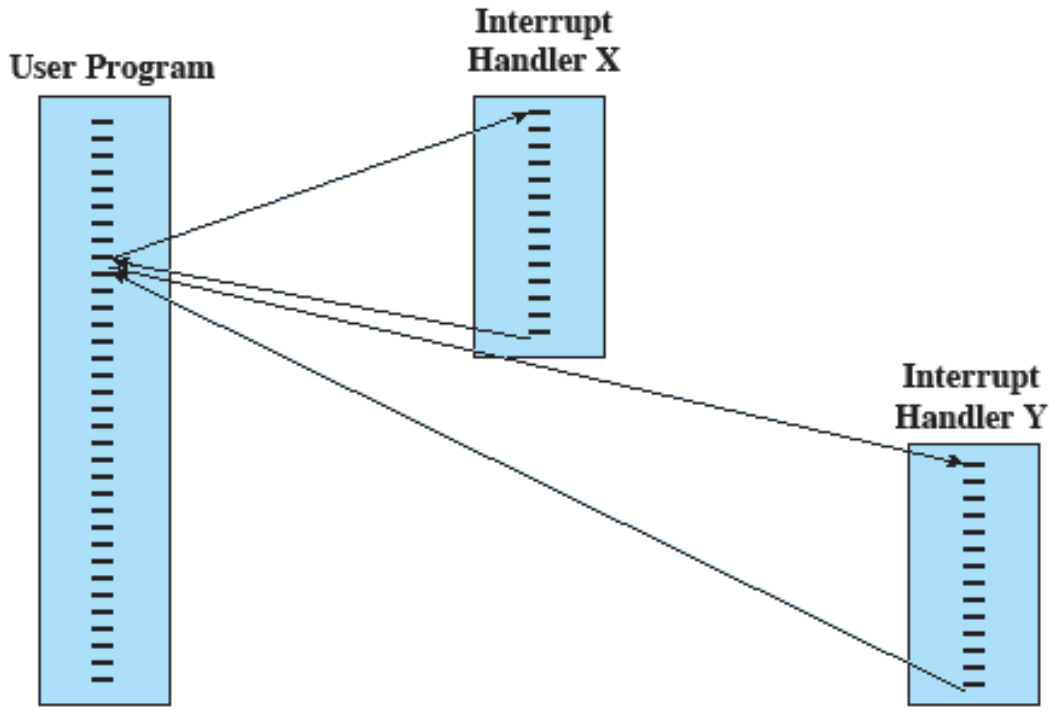
An interrupt occurs while another interrupt is being processed

- e.g. receiving data from a communications line and printing results at the same time

Two approaches:

- disable interrupts while an interrupt is being processed
- use a priority scheme

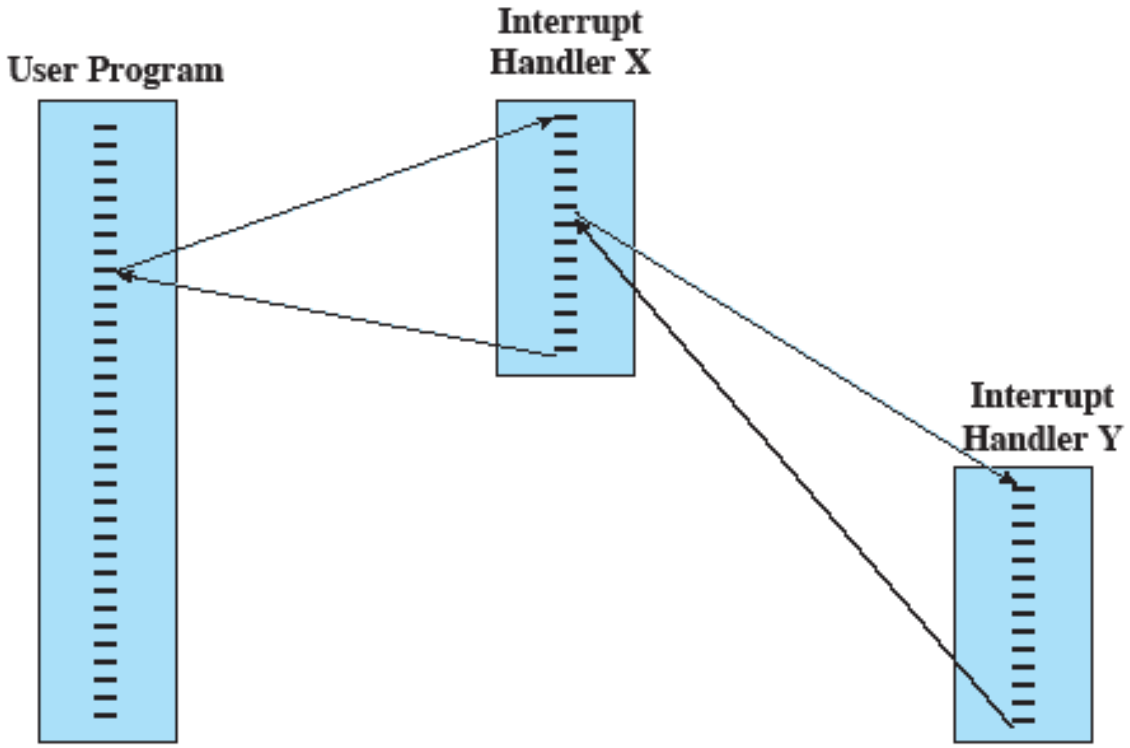
Transfer of Control With Multiple Interrupts:



(a) Sequential interrupt processing

Sequential

Transfer of Control With Multiple Interrupts:

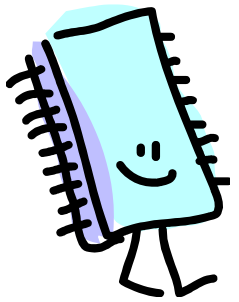


(b) Nested interrupt processing

Nested

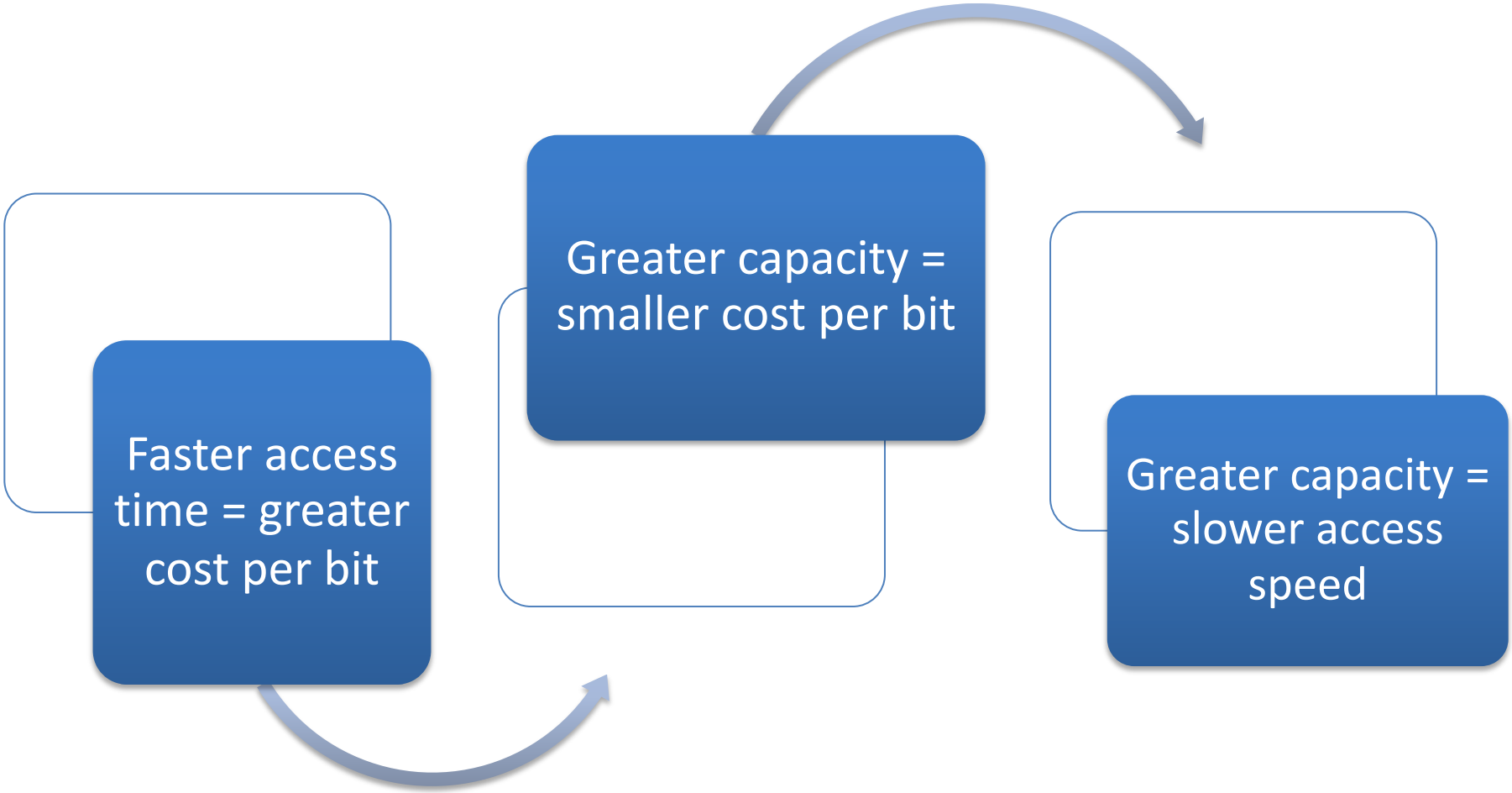
Memory Hierarchy

- Major constraints in memory
  - amount
  - speed
  - expense



- Memory must be able to keep up with the processor
- Cost of memory must be reasonable in relationship to the other components

Memory Relationships



The Memory Hierarchy

- Going down the hierarchy:
  - decreasing cost per bit
  - increasing capacity
  - increasing access time
  - decreasing frequency of access to the memory by the processor

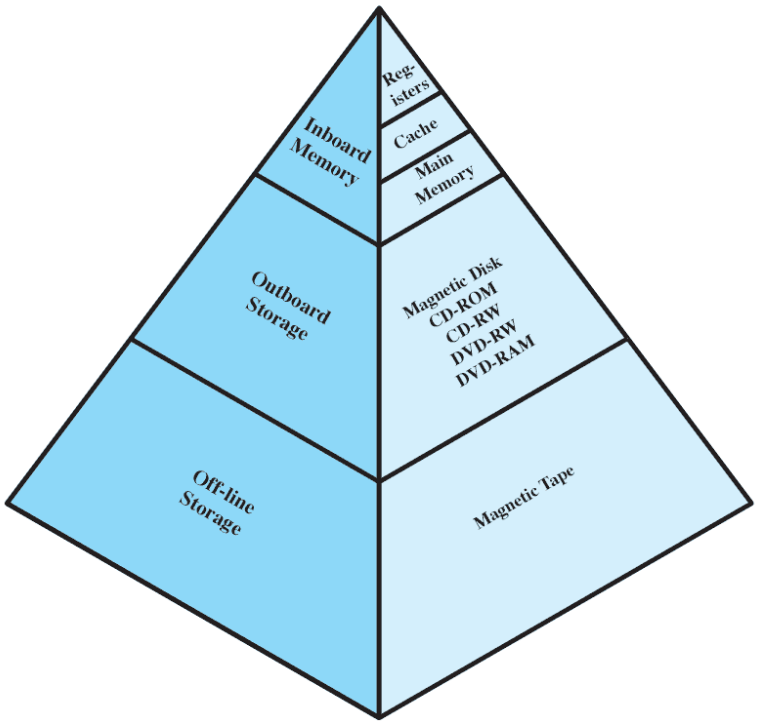
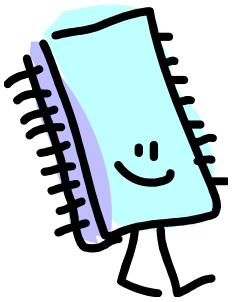


Figure 1.14 The Memory Hierarchy

Performance of a Simple Two-Level Memory



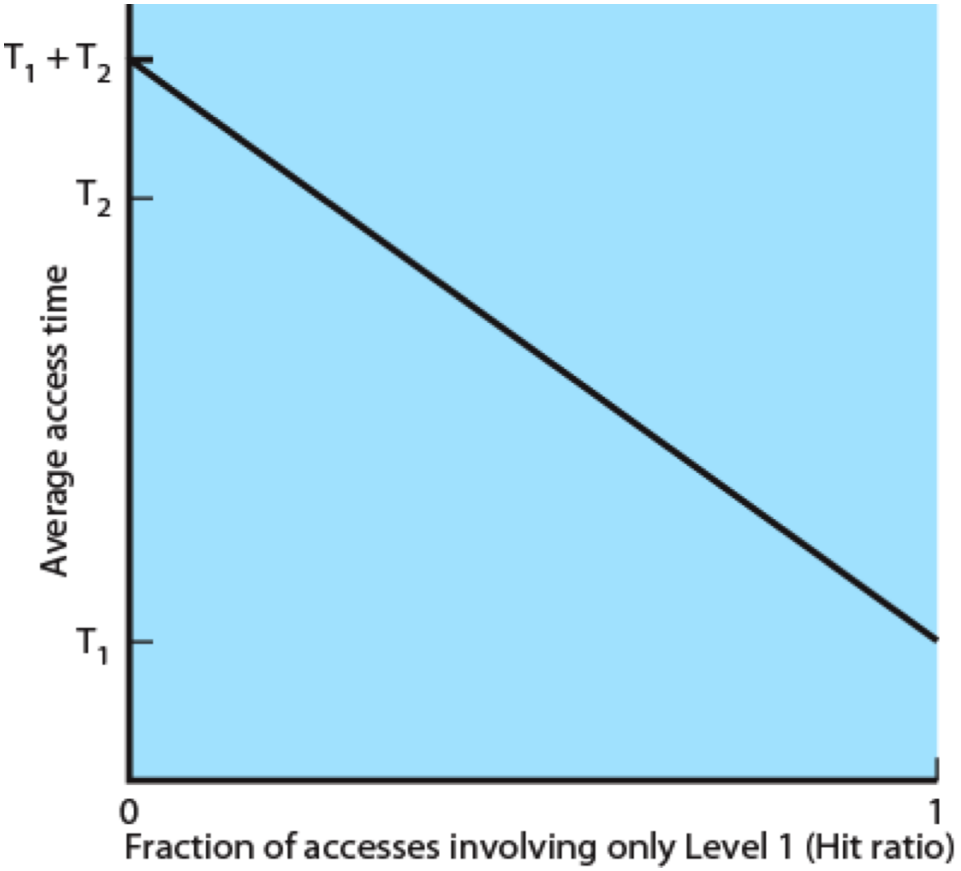
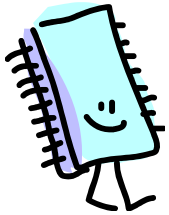


Figure 1.15 Performance of a Simple Two-Level Memory

Principle of Locality

- Memory references by the processor tend to cluster
- Data is organized so that the percentage of accesses to each successively lower level is substantially less than that of the level above
- Can be applied across more than two levels of memory

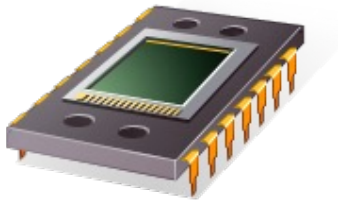
Secondary Memory



Also referred to as auxiliary memory

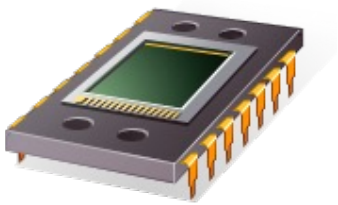
- External
- Nonvolatile
- Used to store program and data files

Cache Memory



- Invisible to the OS
- Interacts with other memory management hardware
- Processor must access memory at least once per instruction cycle
- Processor execution is limited by memory cycle time
- Exploit the principle of locality with a small, fast memory

Cache Principles

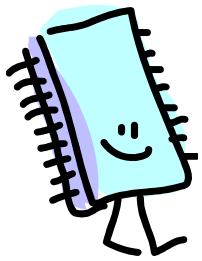


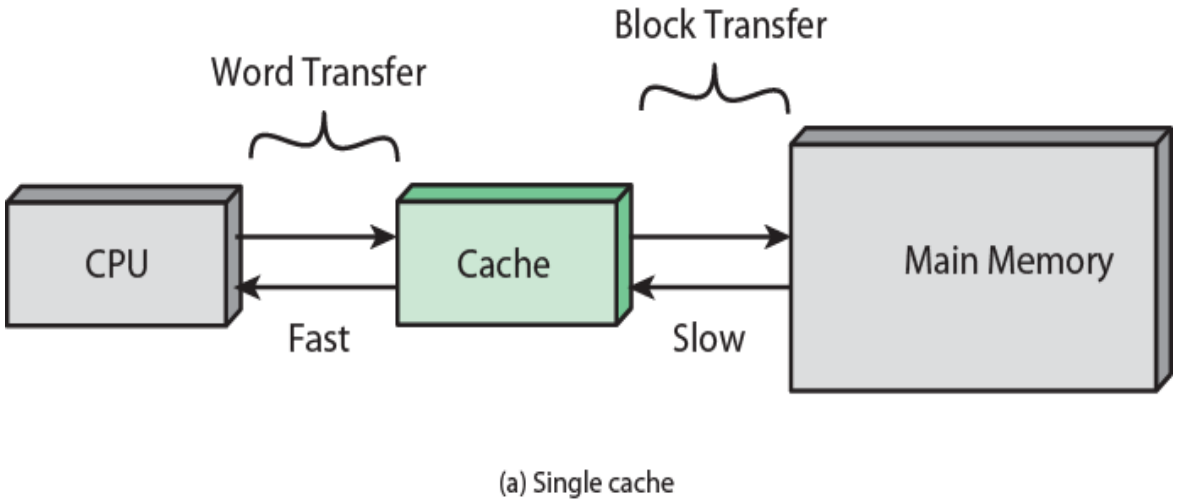
- Contains a copy of a portion of main memory
- Processor first checks cache

– If not found, a block of memory is read into cache

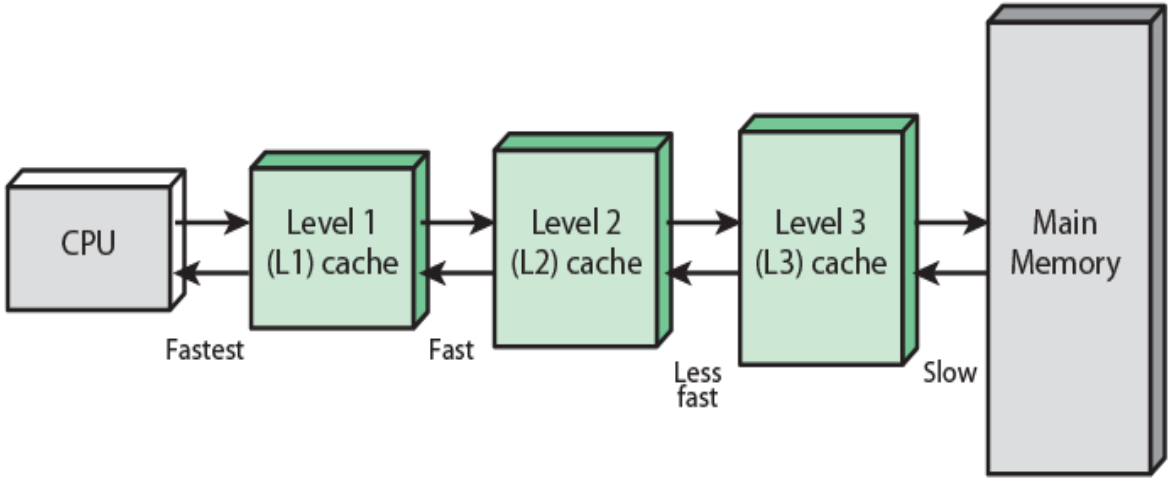
- Because of locality of reference, it is likely that many of the future memory references will be to other bytes in the block

Cache and Main Memory





(a) Single cache



(b) Three-level cache organization

Cache/Main-Memory Structure

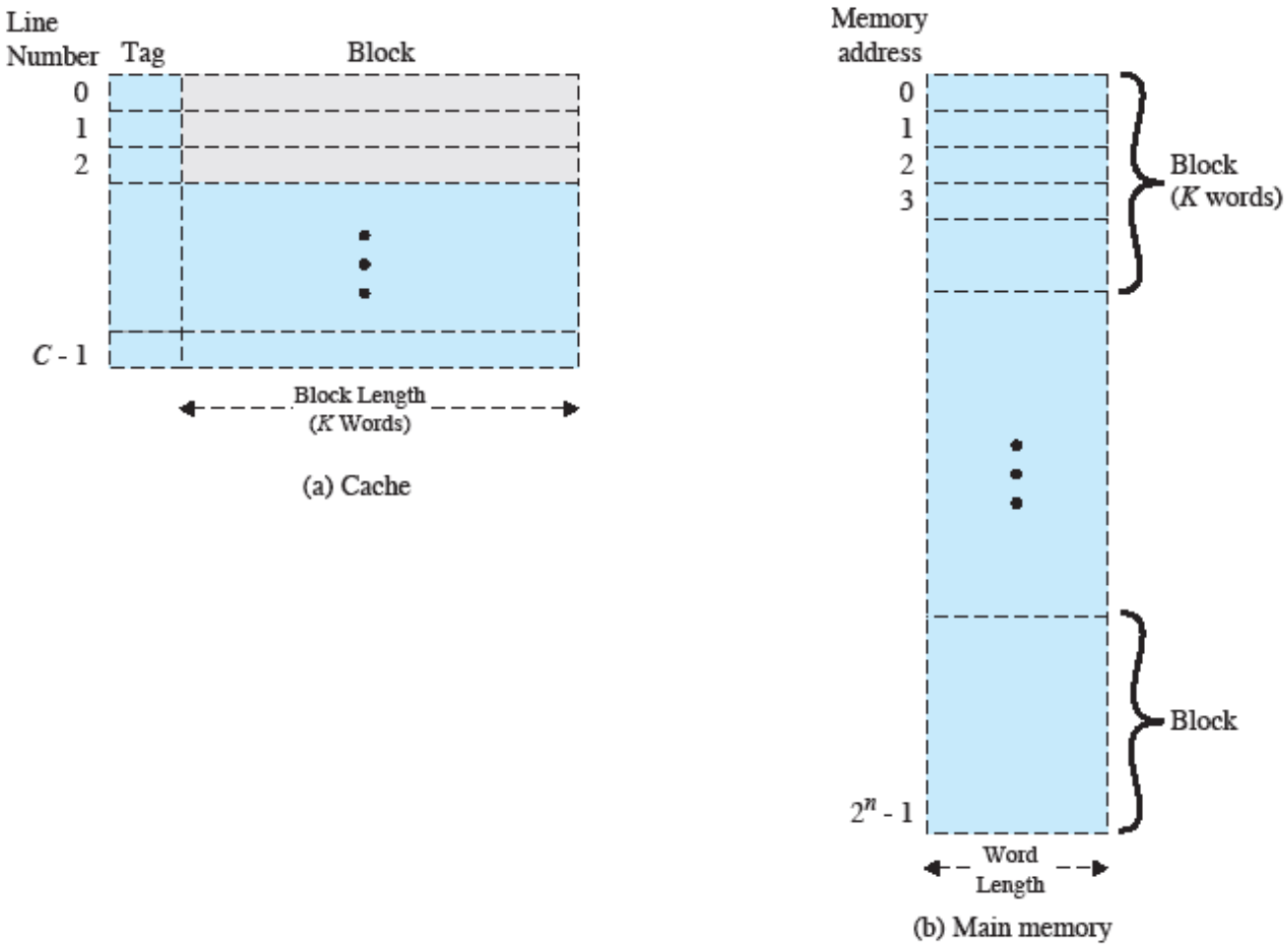
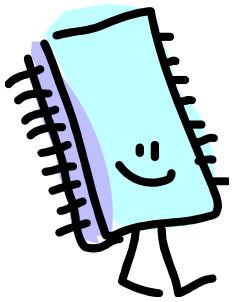


Figure 1.17 Cache/Main-Memory Structure

Cache Read Operation



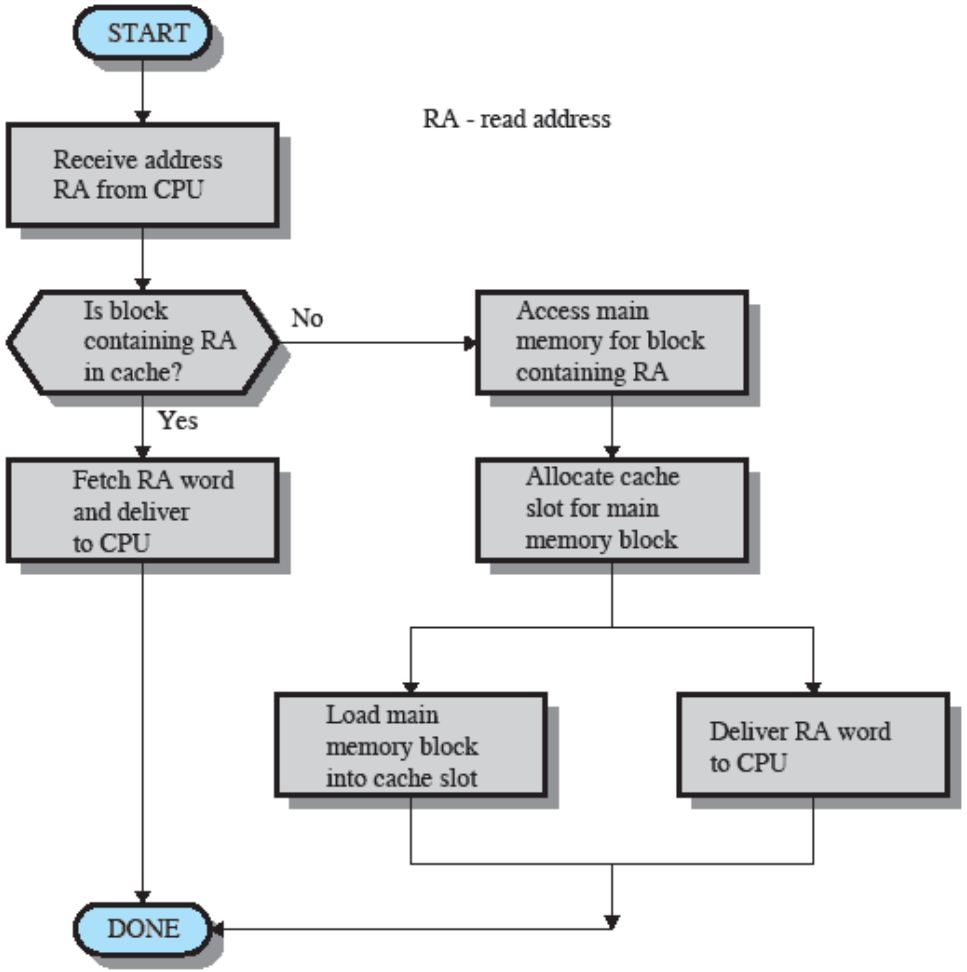
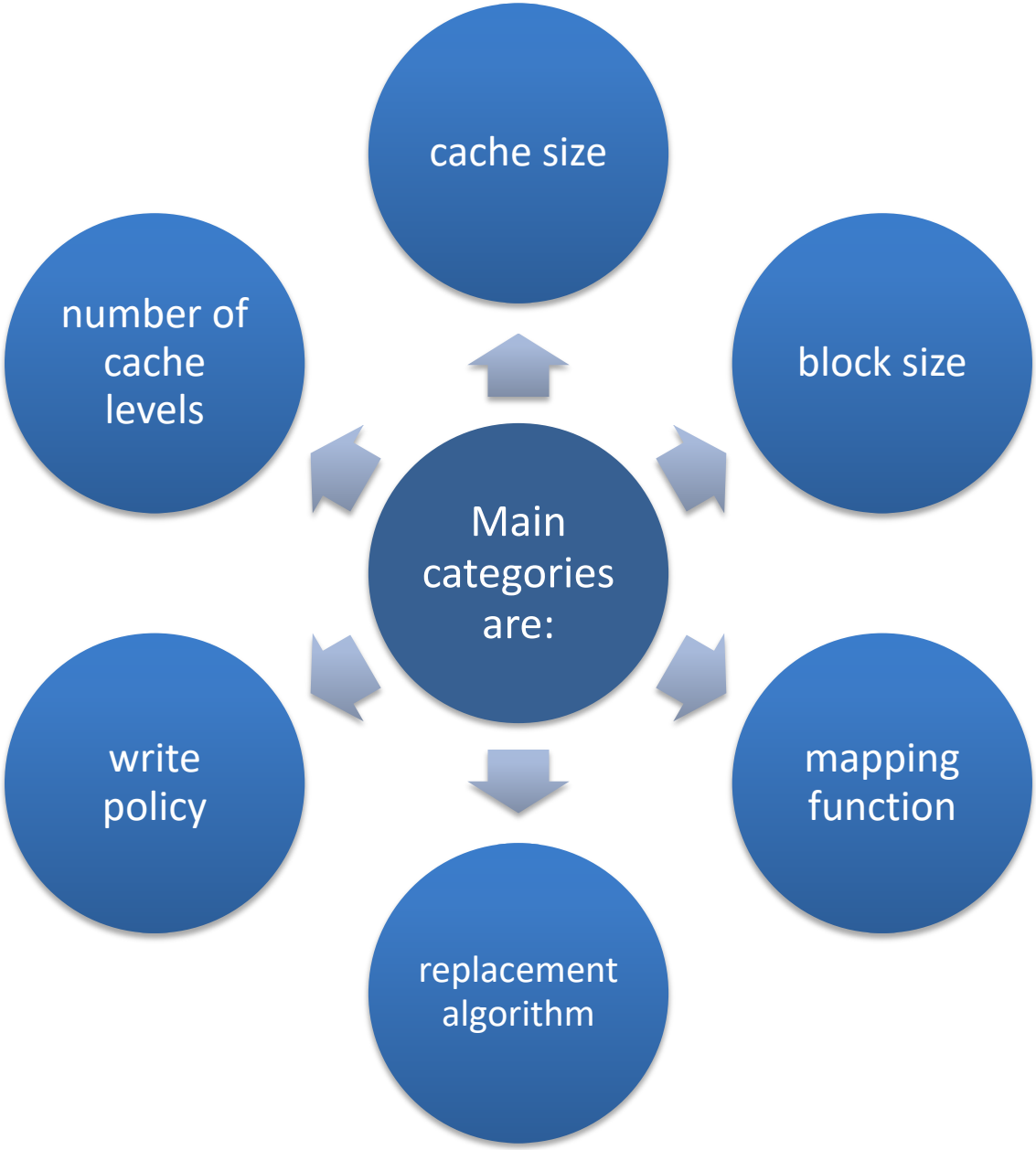


Figure 1.18 Cache Read Operation

CACHE DESIGN



Cache and Block Size

Cache Size

Small caches have significant impact on performance

Block Size

The unit of data exchanged between cache and main memory

Mapping Function

* Determines which cache location the block will occupy

Two constraints affect design:

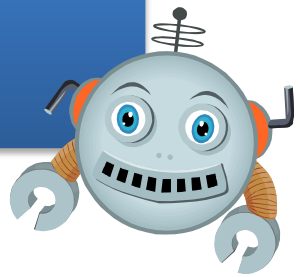
- When one block is read in, another may have to be replaced
- The more flexible the mapping function, the more complex is the circuitry required to search the cache

Replacement Algorithm

- Least Recently Used (LRU) Algorithm
  - effective strategy is to replace a block that has been in the cache the longest with no references to it
  - hardware mechanisms are needed to identify the least recently used block
    - chooses which block to replace when a new block is to be loaded into the cache

Write Policy

Dictates when the memory write operation takes place



- can occur every time the block is updated
- can occur when the block is replaced
  - minimizes write operations
  - leaves main memory in an obsolete state

I/O Techniques

* When the processor encounters an instruction relating to I/O, it executes that instruction by issuing a command to the appropriate I/O module

Three techniques are possible for I/O operations:

- Programmed I/O
- Interrupt-Driven I/O
- Direct Memory Access (DMA)

Programmed I/O

- The I/O module performs the requested action then sets the appropriate bits in the I/O status register
- The processor periodically checks the status of the I/O module until it determines the instruction is complete
- With programmed I/O the performance level of the entire system is severely degraded

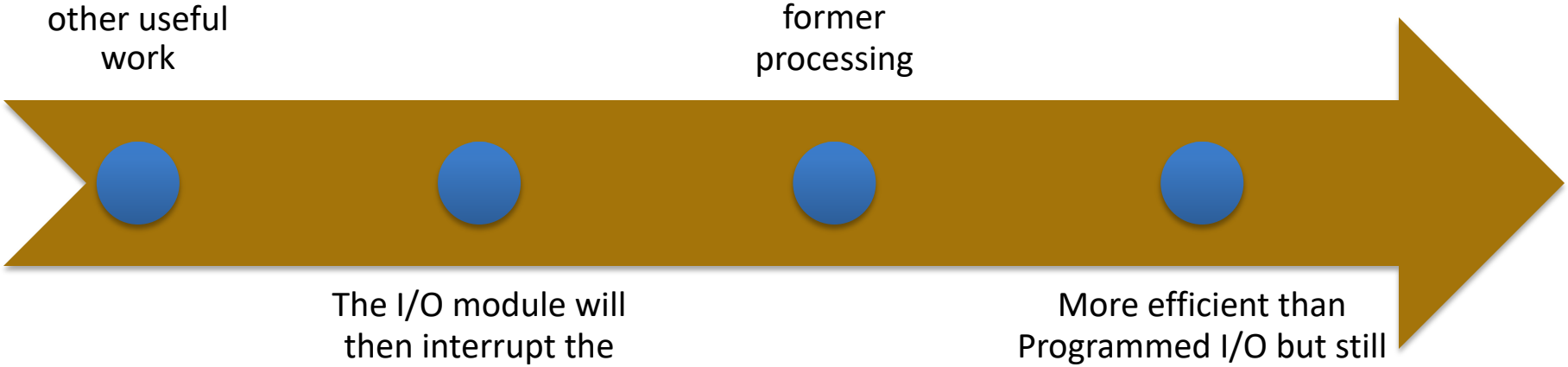
Interrupt-Driven I/O

Processor issues an I/O command to a module and then goes on to do some other useful work

The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor

The processor executes the data transfer and then resumes its former processing

More efficient than Programmed I/O but still requires active intervention of the processor to transfer data between memory and an I/O module



Interrupt-Driven I/O Drawbacks

- Transfer rate is limited by the speed with which the processor can test and service a device
- The processor is tied up in managing an I/O transfer
    - a number of instructions must be executed for each I/O transfer

Direct Memory Access (DMA)

* Performed by a separate module on the system bus or incorporated into an I/O module

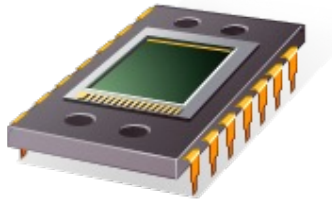
When the processor wishes to read or write data it issues a command to the DMA module containing:

- whether a read or write is requested
- the address of the I/O device involved
- the starting location in memory to read/write
- the number of words to be read/written

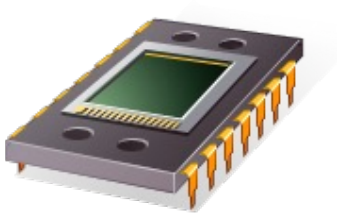
Direct Memory Access

- Transfers the entire block of data directly to and from memory without going through the processor
  - processor is involved only at the beginning and end of the transfer
  - processor executes more slowly during a transfer when processor access to the bus is required
- More efficient than interrupt-driven or programmed I/O

Symmetric Multiprocessors (SMP)



- A stand-alone computer system with the following characteristics:
  - two or more similar processors of comparable capability
  - processors share the same main memory and are interconnected by a bus or other internal connection scheme
  - processors share access to I/O devices
  - all processors can perform the same functions
  - the system is controlled by an integrated operating system that provides interaction between processors and their programs at the job, task, file, and data element levels



Multicore Computer

- Also known as a chip multiprocessor
- Combines two or more processors (cores) on a single piece of silicon (die)
  - each core consists of all of the components of an independent processor
- In addition, multicore chips also include L2 cache and in some cases L3 cache

Summary

- Basic Elements
  - processor, main memory, I/O modules, system bus
  - GPUs, SIMD, DSPs, SoC
  - Instruction execution
    - » processor-memory, processor-I/O, data processing, control
  - Interrupt/Interrupt Processing
  - Memory Hierarchy
  - Cache/cache principles and designs
  - Multiprocessor/multicore