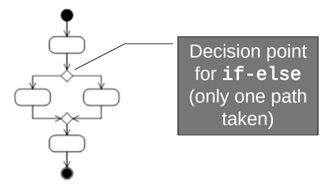
Chapter 12

Concurrency

Sequential Processing

Thread of Control: Sequence of program points reached as control passes through the program

Sequential: Has a single thread of control



Concurrent Systems

Concurrent:

More than one task can be underway at the same time

Parallel:

More than one task can be physically active at the same time

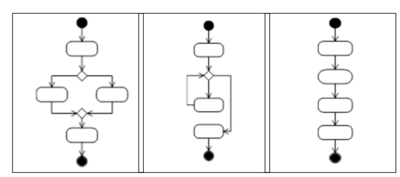
Distributed:

A parallel system with processors that are physically separate

Categories of Concurrency

Physical Concurrency (parallel)**:** Multiple independent processors (multiple threads of control)

Logical Concurrency: appearance of physical concurrency (time-slicing on one processor)



Levels of Parallelism

Instruction Level (ILP): Microprocessor architecture

Vector Parallelism: Perform repeated operations on every element of a large data set (single instruction multiple data - SIMD)

Thread-level Parallelism: Multicore processors/multiple processors (multiple instruction multiple data – MIMD)

Why Study Concurrency

1. *Capture logical structure of a problem*. Many real-world situations involve concurrency (operating systems, simulations, scientific visualization, AI, multimedia, …)
2. *Exploit extra processors.* Computers capable of physical concurrency are now common
3. *Cope with separate physical devices.* Embedded control systems, Internet applications, …

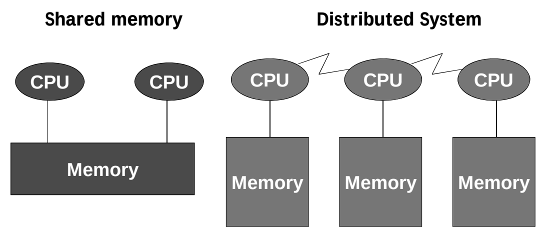
Multiple Cores/Processors

Computers capable of physical concurrency are now common

- Quad-core & Core-2 Quad (Intel)
- Xenon (3 core, Xbox 360)
- Cell (8 core, Sony PlayStation 3)
- Power7 (8 core, Watson has 650 of these)

Tianhe-1A (China)
14,336 Xeon X5670 processors (6 core) and
7,168 Nvidia Tesla M2050 GPUs + more

Models of Concurrency



Models of Concurrency

Important Issues

Synchronized access to shared memory

Message passing between processes that don't share memory

Race Conditions

Occurs when actions in two processes are not synchronized and program behavior depends on the order in which the actions happen

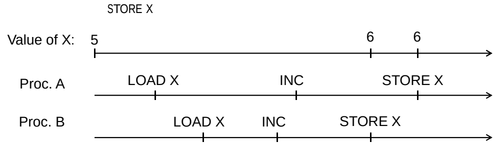
- Usually want to avoid this

Race Conditions

Suppose processors A and B share memory, and both try to increment variable X at the same time

- Each processor executes

```
LOAD X
INC
STORE X
```



Other possibilities?

Synchronization

Mechanism that controls the order in which processes/tasks execute

Can be used to eliminate race conditions

- In the example we need to synchronize the increment operations to enforce *mutually exclusive* access to X

Requires a mechanism for delaying task execution

- Task scheduling

Task communication is needed for synchronization

Kinds of Synchronization

Cooperation

- Task A waits for Task B to complete some activity before it continues
- The two tasks work on parts of the same problem

Competition

- Different tasks need exclusive use of the same resource