Introduction and Overview

ECE 3574 Applied Software Design

Dr. Roberto Palmieri
ECE Department
Virginia Tech

Textbooks

- Required:
  - An Introduction to Design Patterns in C++ with Qt, Second Edition, Alan Ezust and Paul Ezust, ISBN-10: 0-132-82645-3, ISBN-13: 978-0-132-82645-7, Prentice Hall, 2012
  - Book has a homepage at: https://www.ics.com/design-patterns#.U-4ouLxdVXO
    - Site may require free registration
    - Find links to HTML version of the book, lecture slides, source codes of examples in book, bug reporting system, etc.
  - VT library has the online version of the book at: http://proquest.safaribooksonline.com.ezproxy.lib.vt.edu/9780132851619

Textbooks (contd.)

- Recommended:
  - Operating System Concepts, Ninth Edition, Avi Silberschatz, Peter Baer Galvin, and Greg Gagne, ISBN 978-1-118-06333-0, John Wiley & Sons, Inc., 2012
    - http://www.os-book.com/
  - VT library has the online version of the book at (8th edition): http://proquest.safaribooksonline.com.ezproxy.lib.vt.edu/book/operating-systems-and-server-administration/9780470128725
  - Great OS book!

Outline

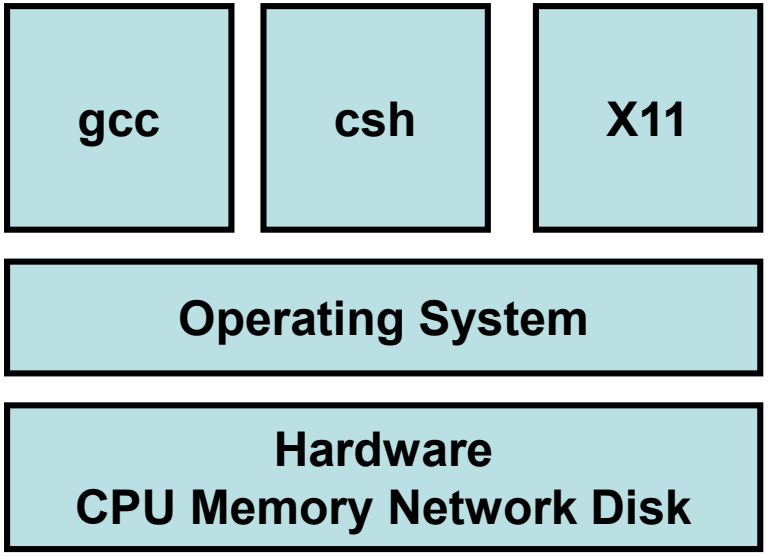
- Introduction to Operating System (NOT IN THE BOOK)
- Introduction to processes (Ch 3, Silberschatz)
- Inter-process communication: shared memory, message passing (Ch 3, Silberschatz)
- Introduction to threads (Ch 4, Silberschatz)

- Process synchronization
  - critical section problem (Ch 6, Silberschatz)
  - software solutions including Peterson's algorithm and generalizations (Ch 6, Silberschatz, Notes)
  - hardware solutions including TAS and CAS-based (Ch 6, Silberschatz, Notes)
  - semaphores (Ch 6, Silberschatz, Notes)
  - classical synchronization problems (Ch 6, Silberschatz, Notes)
  - condition synchronization (Ch 6, Silberschatz, Notes)

- Qt Concurrency (Ch 17, Ezust)

What is an Operating System

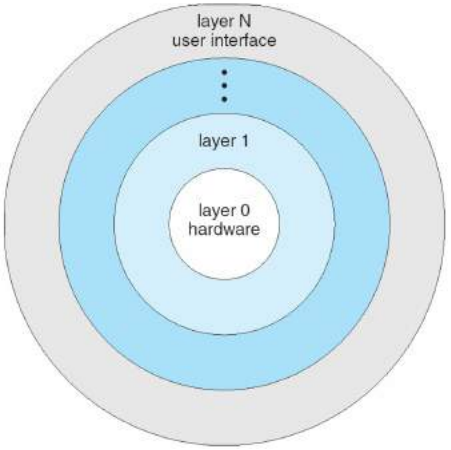
- Software layer that sits between applications and hardware
- Performs services:
  - Abstracts hardware
  - Provides protection
  - Manages resources
- Abstraction is fundamental!!!



OS vs Kernel

- Kernel:
  - the special piece of software that runs with special privileges and actually controls the machine.
- Operating System:
  - Includes also system programs, system libraries, servers, shells, GUI etc.
    - Example: is the Terminal really needed to run your operating system?

What does ”Privileged Mode” mean?

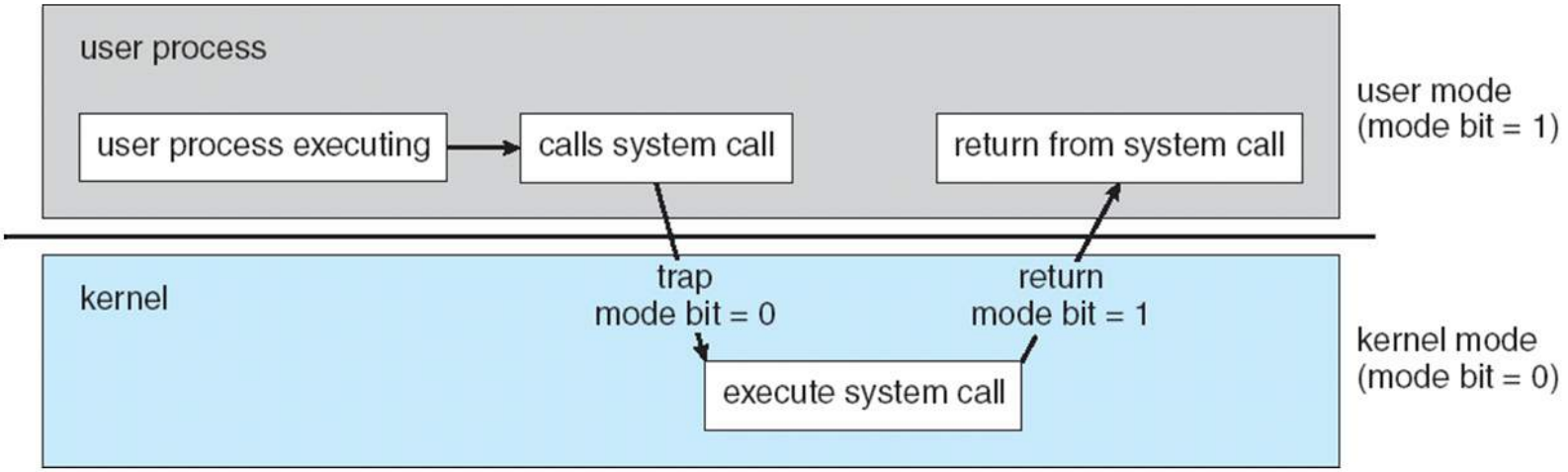


- Two fundamental modes:
  - kernel or privileged mode
  - user or non-privileged mode
- Protection operations can only be performed in kernel mode.
  - Example: HLT is an assembly language instruction that halts the central processing unit (CPU). Pretty much your CPU is gone!

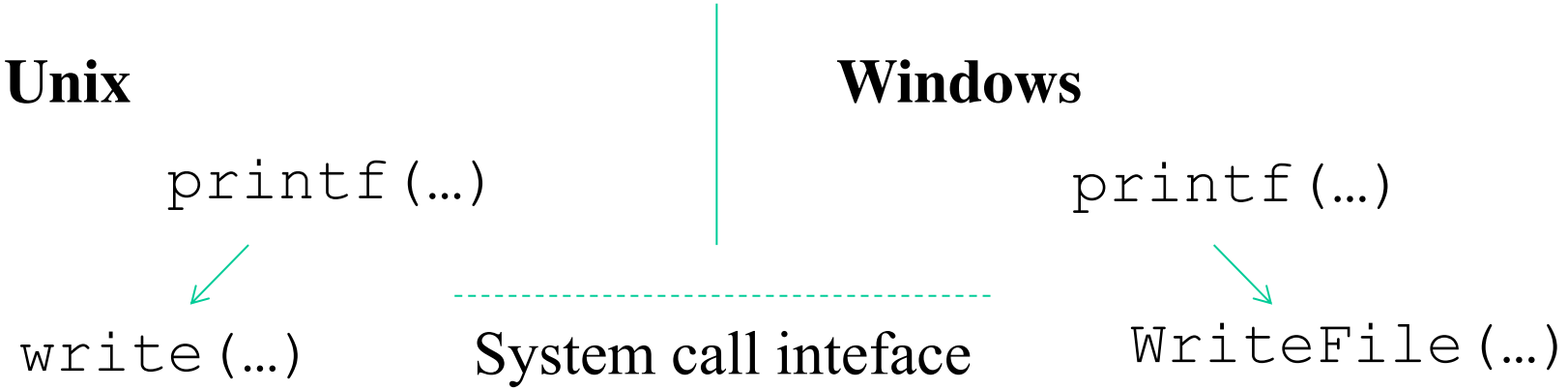
```
int main() {
        ...
        asm(“hlt“)
        ...
}
```

System Calls

- System Calls are functions that represent the gateway to access the operating system's services
- Most high-level instructions produce <u>many</u>



System call is machine dependent



How can the OS manage multiple programs running in parallel?

- Interactive Time-Sharing:
  - Ability of running multiple programs in parallel on a single processor unit
  - Each program receives a slice (*quantum*) of time to execute. To support that:
    - Preemption
    - Scheduler
    - Technique to save and restore execution contexts

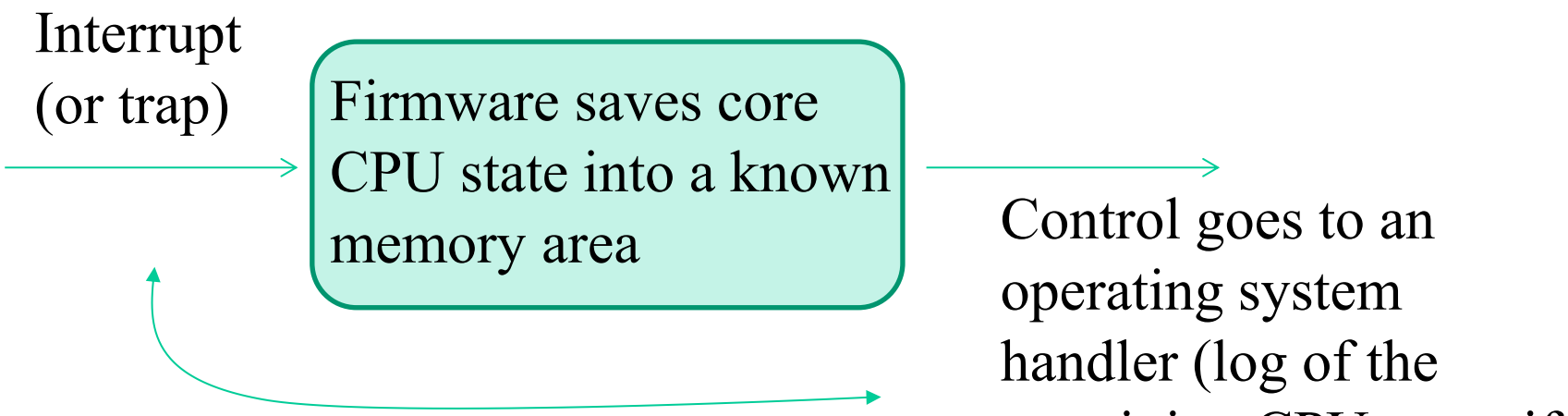
Preemption

- Preemption is a mechanism to reclaim the execution of a program while it is executing
- Preemption can happen when (examples):
  - An interrupt is received, so that the OS can handle it
  - The time slice assigned to a program to execute expires
  - …

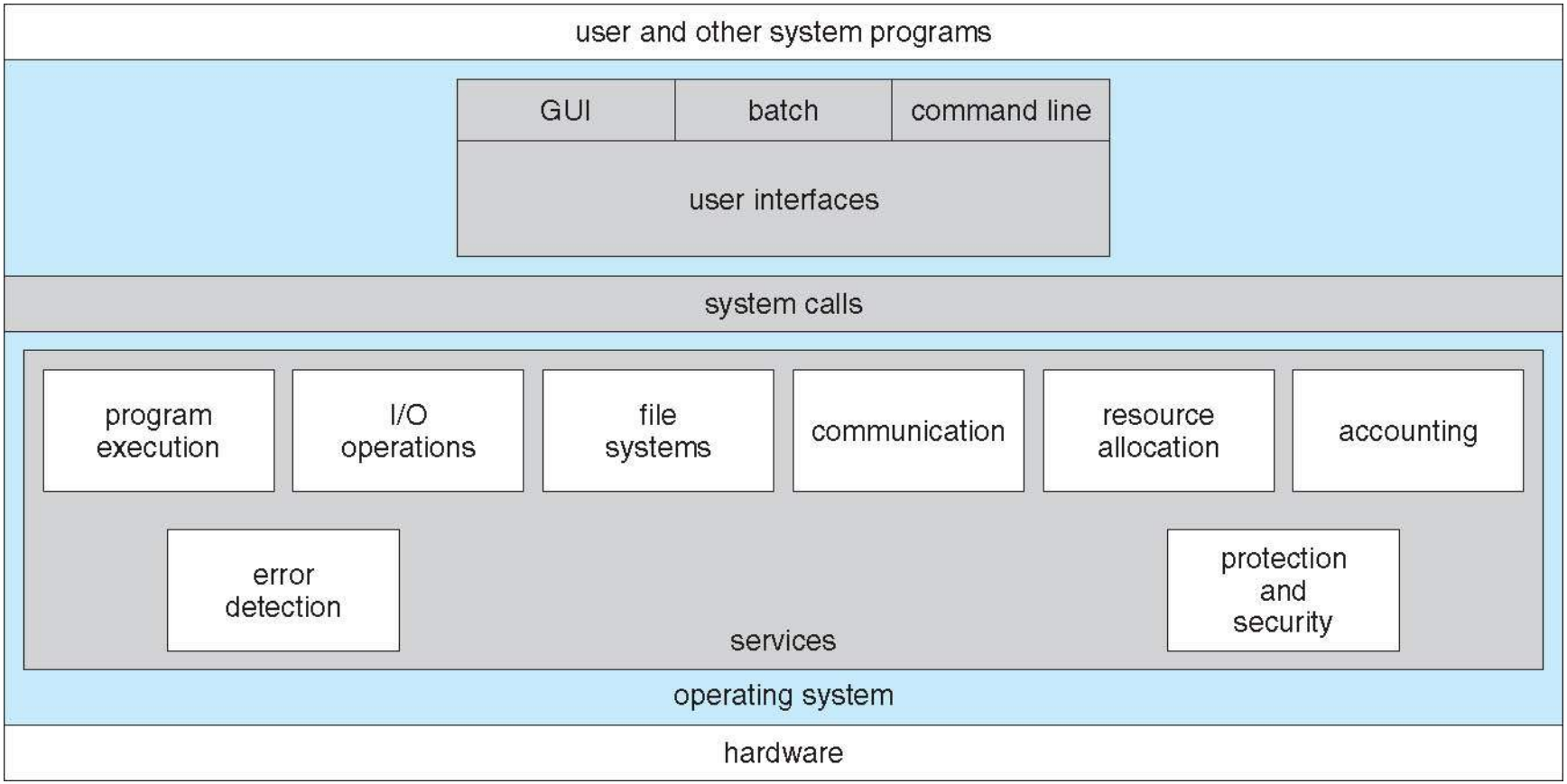
Interrupt Driven

- Modern operating systems are interrupt driven
  - Interrupt is a software or hardware notification that some event happened. Examples:
    - Mouse click
    - Time slice is over
    - Reset button pressed
  - A routine is activated once an interrupt is received, the driver

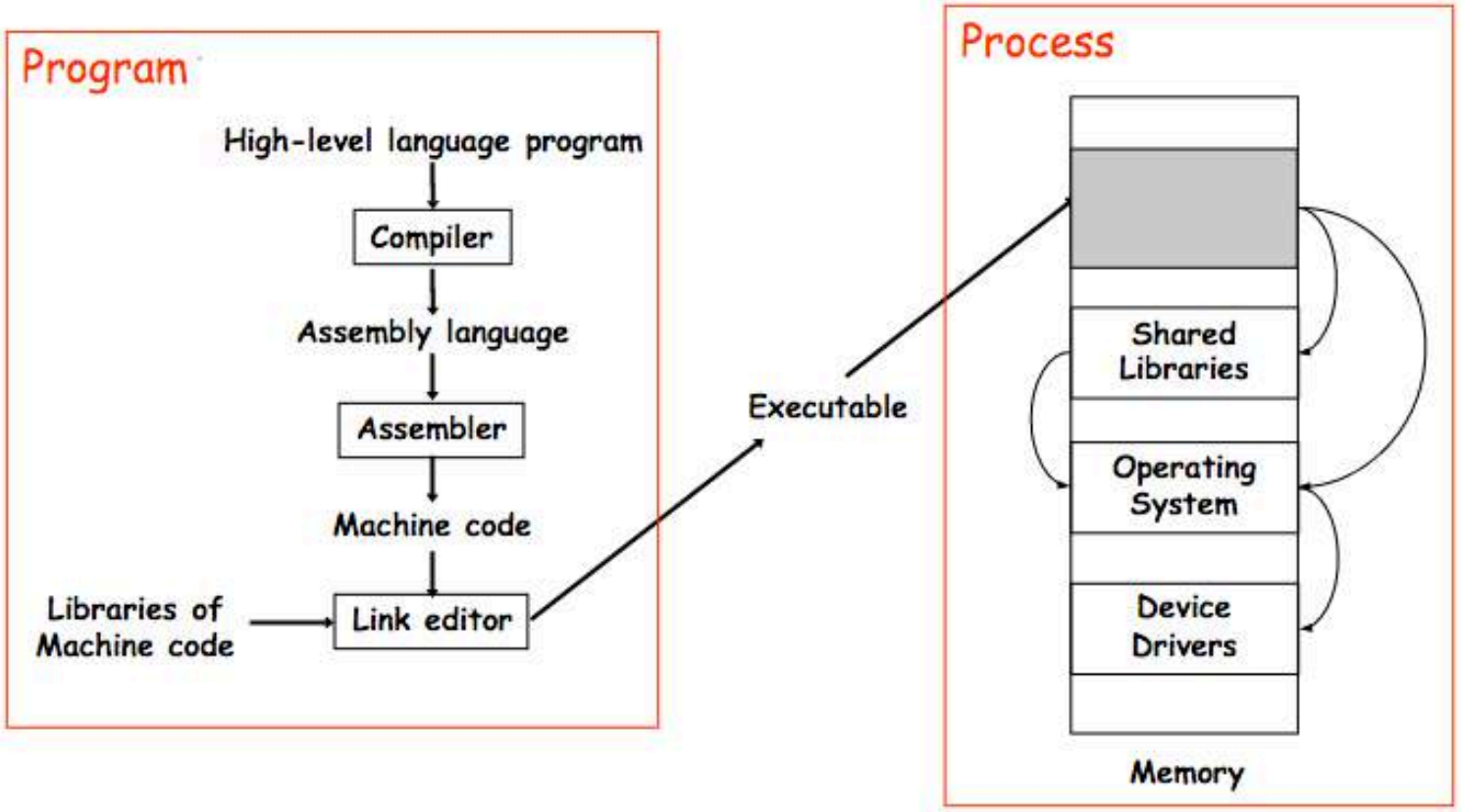
Interrupt driven example



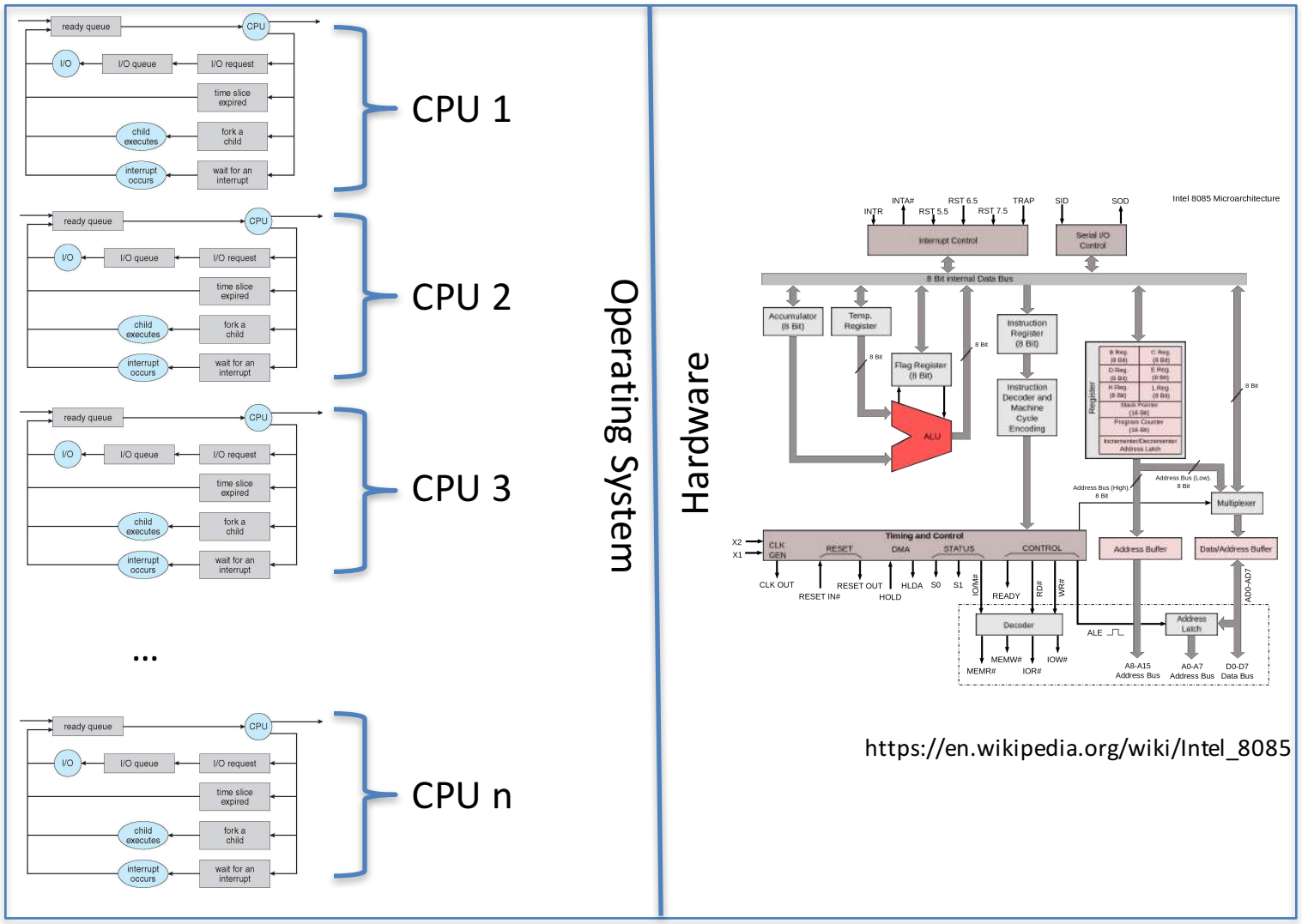
General Architecture OS



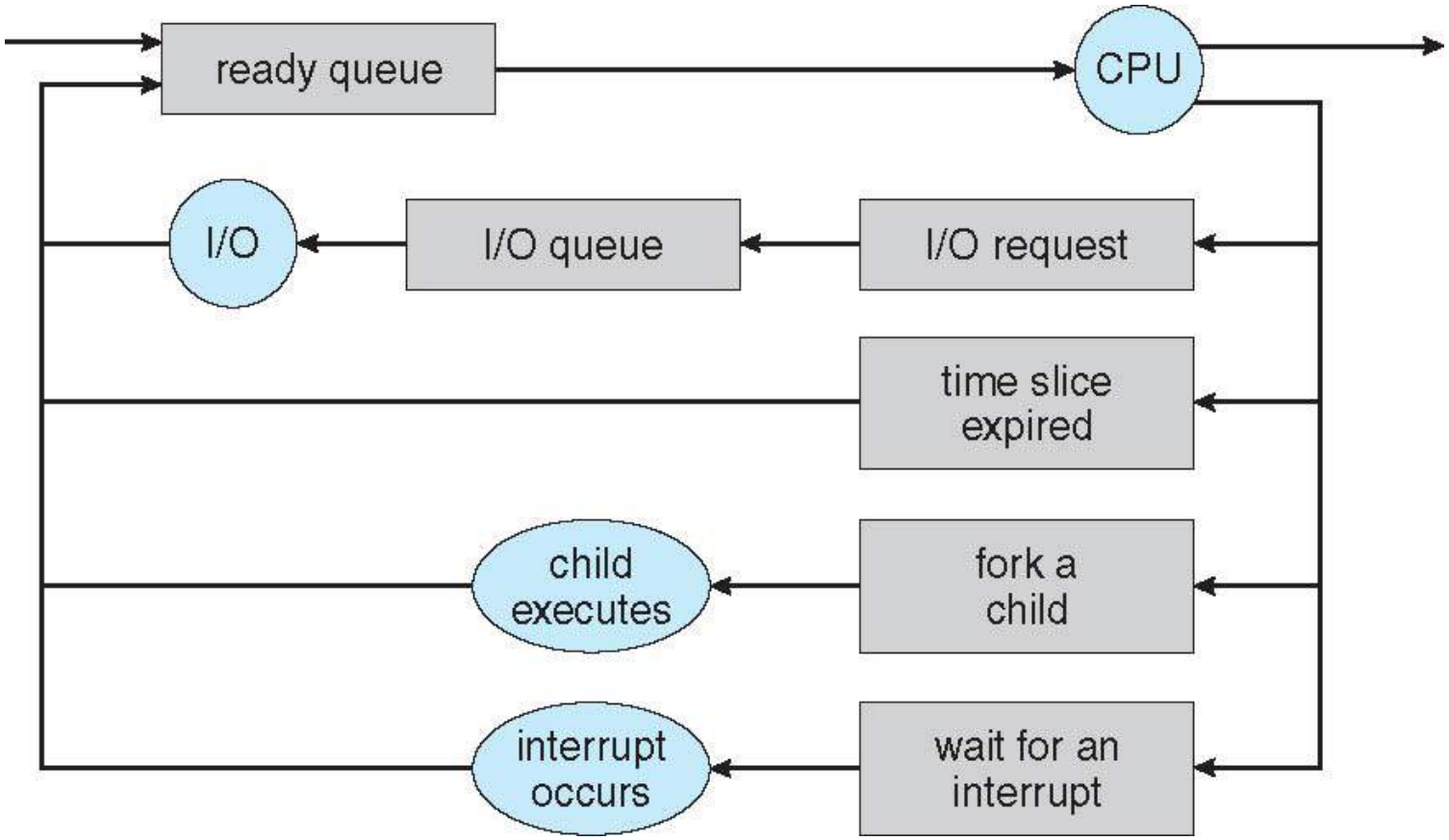
How does a program execute in the OS?



Software and Hardware interaction



Zoom In


ready queue
CPU
I/O
I/O queue
I/O request
time slice expired
child executes
fork a child
interrupt occurs
wait for an interrupt

How the program executes…so far?

- You assumed so far that your code executes sequentially, from top to bottom.

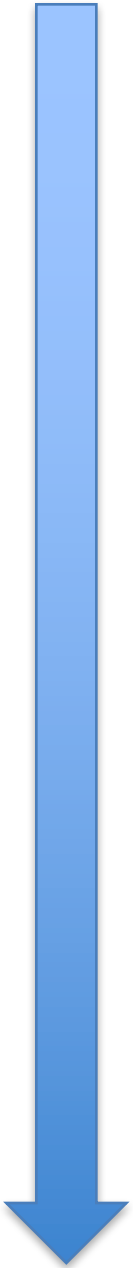
```
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    QFile textFile("playlist1.tsv");
    QFile binaryFile("playlist1.bin");
    QTextStream textStream;
    QDataStream dataStream;

    if (textFile.open(QIODevice::ReadOnly)) {
        textStream.setDevice(&textFile);
        qtCout << "open text file for mdv" << endl;
    }
    else
        return EXIT_FAILURE;
    if (binaryFile.open(QIODevice::WriteOnly))  {
        dataStream.setDevice(&binaryFile);
        qtCout << "open binary file for mdv" << endl;
    }
    else
        return EXIT_FAILURE;

        ……..……
```

1st

2nd



What about they execute in parallel?

```
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    QFile textFile("playlist1.tsv");
    QFile binaryFile("playlist1.bin");
    QTextStream textStream;
    QDataStream dataStream;

    if (textFile.open(QIODevice::ReadOnly)) {
        textStream.setDevice(&textFile);
        qtCout << "open text file for mdv" << endl;
    }
    else
        return EXIT_FAILURE;
    if (binaryFile.open(QIODevice::WriteOnly)) {
        dataStream.setDevice(&binaryFile);
        qtCout << "open binary file for mdv" << endl;
    }
    else
        return EXIT_FAILURE;

        ..............
```

```
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    QFile textFile("playlist1.tsv");
    QFile binaryFile("playlist1.bin");
    QTextStream textStream;
    QDataStream dataStream;

    if (textFile.open(QIODevice::ReadOnly)) {
        textStream.setDevice(&textFile);
        qtCout << "open text file for mdv" << endl;
    }
    else
        return EXIT_FAILURE;
    if (binaryFile.open(QIODevice::WriteOnly)) {
        dataStream.setDevice(&binaryFile);
        qtCout << "open binary file for mdv" << endl;
    }
    else
        return EXIT_FAILURE;

        ..............
```

What about they execute in parallel?

```
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    QFile textFile("playlist1.tsv");
    QFile binaryFile("playlist1.bin");
    QTextStream textStream;
    QDataStream dataStream;

    if (textFile.open(QIODevice::ReadOnly)) {
        textStream.setDevice(&textFile);
        qtCout << "open text file for mdv" << endl;
    }
    else
        return EXIT_FAILURE;
    if (binaryFile.open(QIODevice::WriteOnly)) {
        dataStream.setDevice(&binaryFile);
        qtCout << "open binary file for mdv" << endl;
    }
    else
        return EXIT_FAILURE;

        ..............
```

```
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    QFile textFile("playlist1.tsv");
    QFile binaryFile("playlist1.bin");
    QTextStream textStream;
    QDataStream dataStream;

    if (textFile.open(QIODevice::ReadOnly)) {
        textStream.setDevice(&textFile);
        qtCout << "open text file for mdv" << endl;
    }
    else
        return EXIT_FAILURE;
    if (binaryFile.open(QIODevice::WriteOnly)) {
        dataStream.setDevice(&binaryFile);
        qtCout << "open binary file for mdv" << endl;
    }

                FAILURE;

            ..............
```

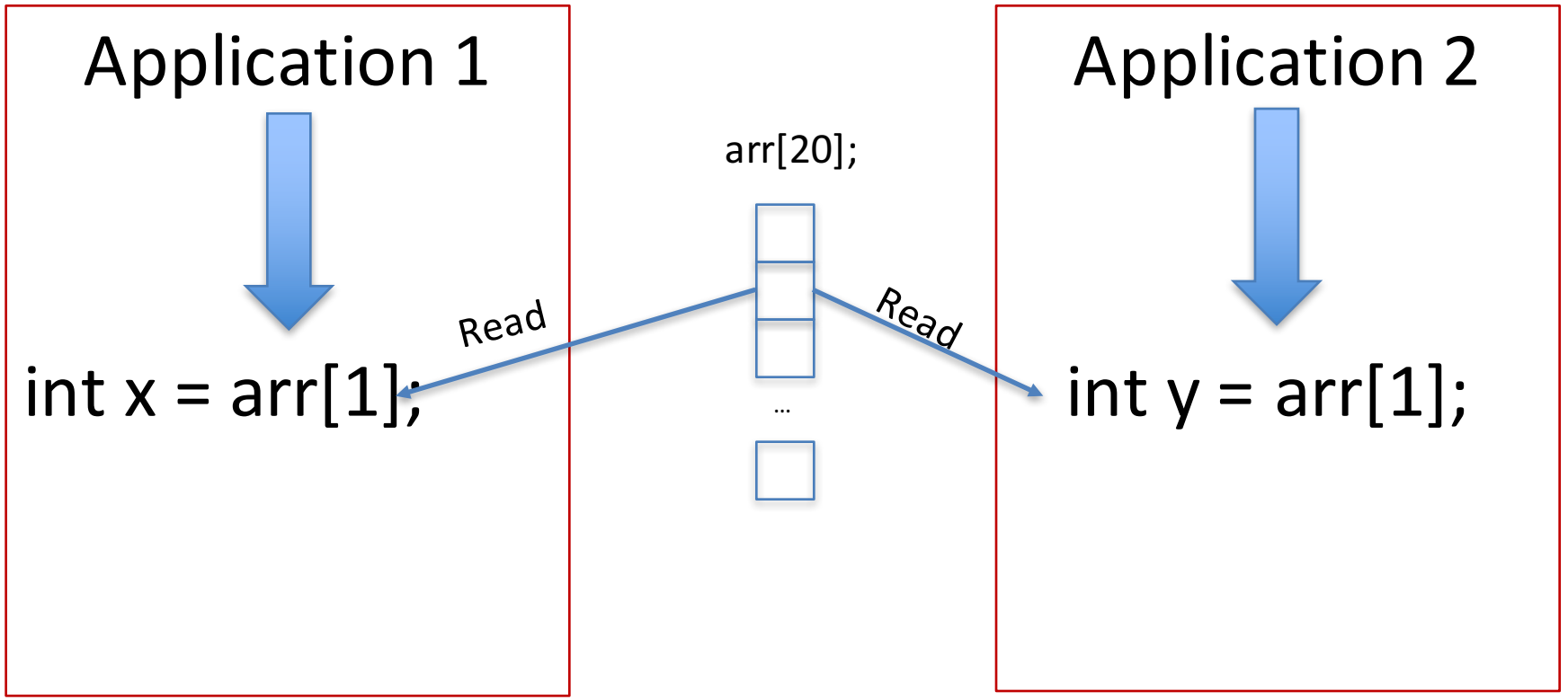
CONFLICT!!

Why do we need parallelism?

- To improve performance (how fast the application can go)
- Ideal example:
  - Application execution time is 10s when executes on one CPU (Sequentially)
  - If it executes in parallel:
    - 5s on two CPUs
    - 2.5s on four CPUs
    - 1.25s on 8 CPUs
    - ….

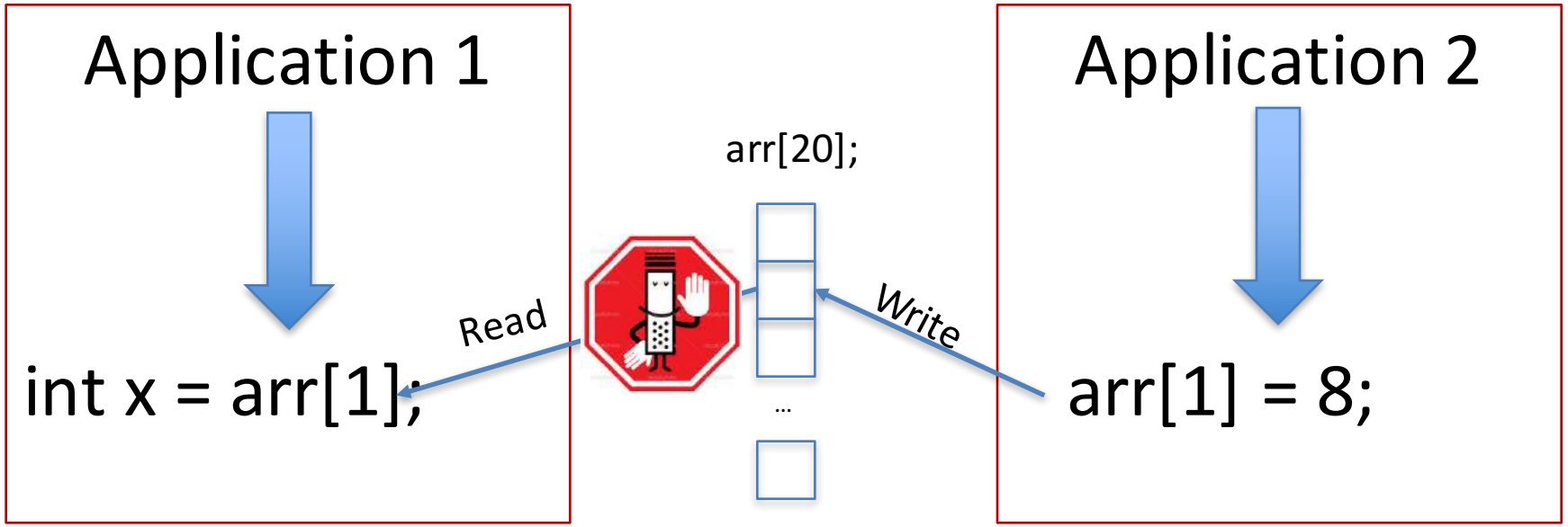
With or Without Contention?

Shared array: *int arr[20];*



With or Without Contention?

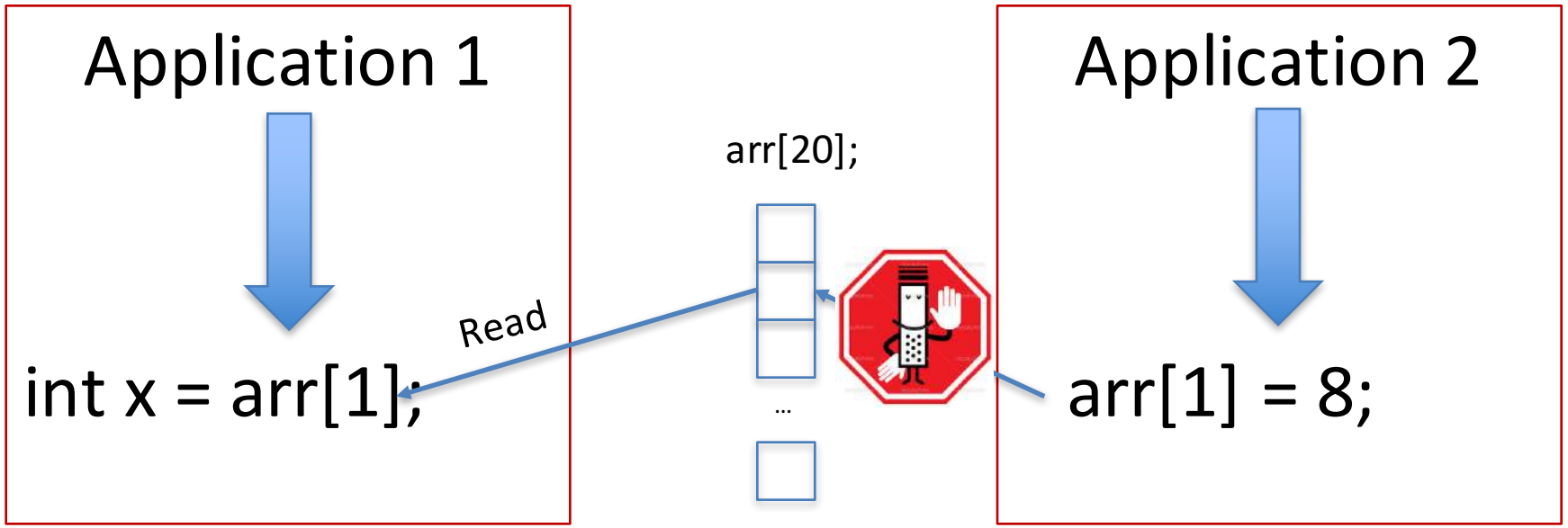
Shared array *int arr[20];*



- Either the Read or Write operation has to stop in order to preserve data coherency
  - Application execution time reduces due to the STOP

With or Without Contention?

Shared array *int arr[20];*



- Either the Read or Write operation has to stop in order to preserve data coherency
  - Application execution time reduces due to the STOP

Performance improvement?

- Example with conflicts:
  - Application execution time is 10s when executes on one CPU (Sequentially)
  - If it executes in parallel:
    - 7s on two CPUs
    - 5.5s on four CPUs
    - 4s on 8 CPUs
    - ….

Parallelism Vs Concurrency

- Informal agreement on the definitions
- Parallel code:
  - executes in parallel and independently and provides performance similar to the ideal
- Concurrent code:
  - Executes in parallel but accessing common shared memory locations, thus performance cannot be ideal

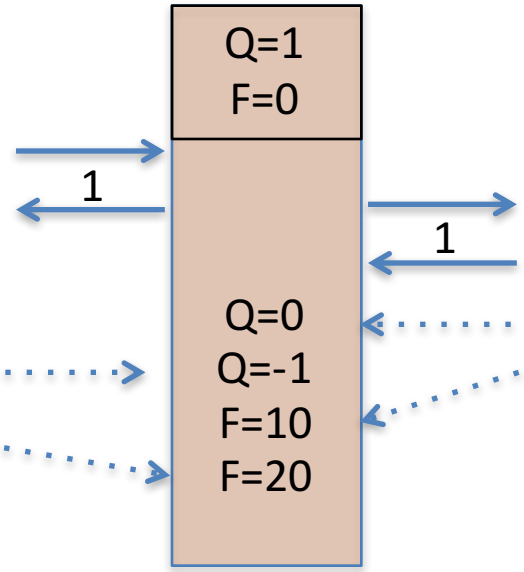
Manage concurrency: real example

BOB

Quantity (Q)=1
Store Fund (F)=0

ALICE

Buy a book {
1) How many available copies of the book are there?
2) Quantity = Quantity - 1
3) Fund = Fund + 10$
}



Buy a book {
1) How many available copies of the book are there?
2) Quantity = Quantity – 1
3) Fund = Fund + 10$
}

Manage concurrency: real example

BOB

Quantity (Q)=1
Store Fund (F)=0

ALICE

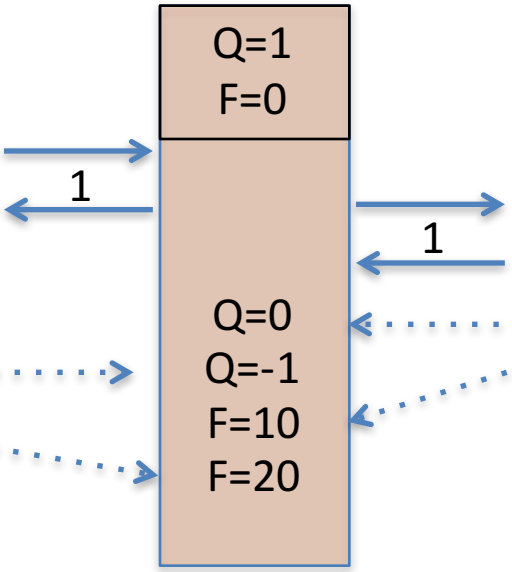
Buy a book {

1) How many available copies of the book are there?

2) Quantity = Quantity - 1

3) Fund = Fund + 10$

}



Buy a b

1) How

the

2) Qua

3) Fun

}

s of

Manage concurrency: real example

BOB

ALICE

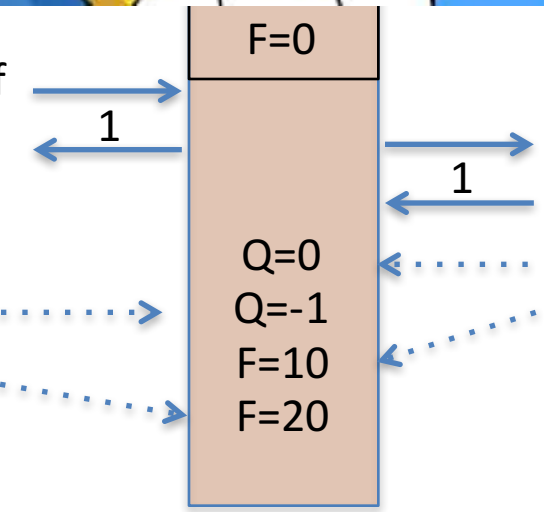
Buy a book {

1) How many available copies of the book are there?

2) Quantity = Quantity - 1

3) Fund = Fund + 10$

}



Buy a b

1) How

the

2) Qua

3) Fun

}

s of

Manage concurrency: real example

BOB

ALICE

Buy a book {

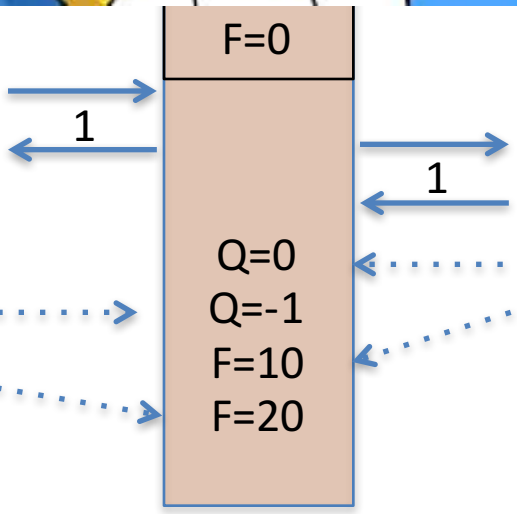
1

2

3

}

ies of



Buy a b

1) How

the

2) Qua

3) Fun

}

s of

Manage concurrency: real example

BOB

Flight = 1 seat available
Hotel = 100 rooms available
Colosseum = 100 tickets

ALICE

Buy a vacation to visit Colosseum

1) Is there any free seat on the flight to Rome (Italy)? Yes!
2) Is there any available room? Yes!
3) Is there any available ticket for visiting the Colosseum? Yes!
4) Reserve Colosseum. Done!
5) Reserve room. Done!
6) Reserve flight. KO!

Buy a vacation to visit Rome

1) Is there any free seat on the flight to Rome (Italy)? Yes!
2) Reserve flight. Done!

Manage concurrency: real example

BOB

Flight = 1 seat available
Hotel = 100 rooms available
Colosseum = 100 tickets

ALICE

Buy a vacation to visit Colosseum

1) Is there any free seat on the flight to Rome (Italy)? Yes!
2) Is there any available room? Yes!
3) Is there any available ticket for visiting the Colosseum? Yes!
4) Reserve Colosseum. Done!
5) Reserve room. Done!
6) Reserve flight. KO!

Buy a va
1) Is th ne
fligh s!
2) Reserve flight. Done!

Manage concurrency: real example

BOB

Flight = 1 seat available
Hotel = 100 rooms available
Colosseum = 100 tickets

ALICE

Buy a vacation to visit Colosseum

1) the
s!
2) m? Yes!
3) et for
Yes!
4) e!
5) Reserve room. Done!
6) Reserve flight. KO!

Buy a va
1) Is th ne
fligh s!
2) Reserve flight. Done!

Race condition

- A race condition is an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly.

Lessons

- Parallelism does not entail concurrency
- Manage concurrency is important.
  - If ignored, application's behavior is unpredictable, which means NOT correct
  - If done superficially, it will be easy but performance will be bad!!
  - If done well, it will be painful but performance are great!!