Overview of a System

Systems Programming

(CST-210)

Dr. Arka P. Mazumdar

Outline

- **Revisit C Compilation**
- Tour of a Computer System
- Running a C program
- Cache Memory
- Storage Hierarchy
- Operating System Concepts

Revisit C Compilation

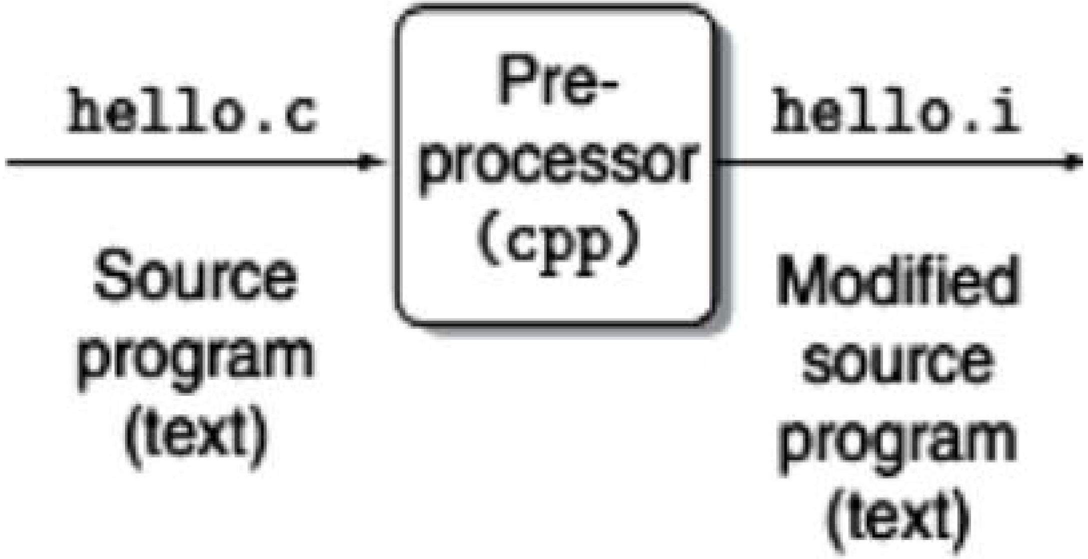
- A very simple C program:

```
#include<stdio.h>
void main() {
     printf(“Hello World \n”);
}
```

- We stored the program in ***hello.c***

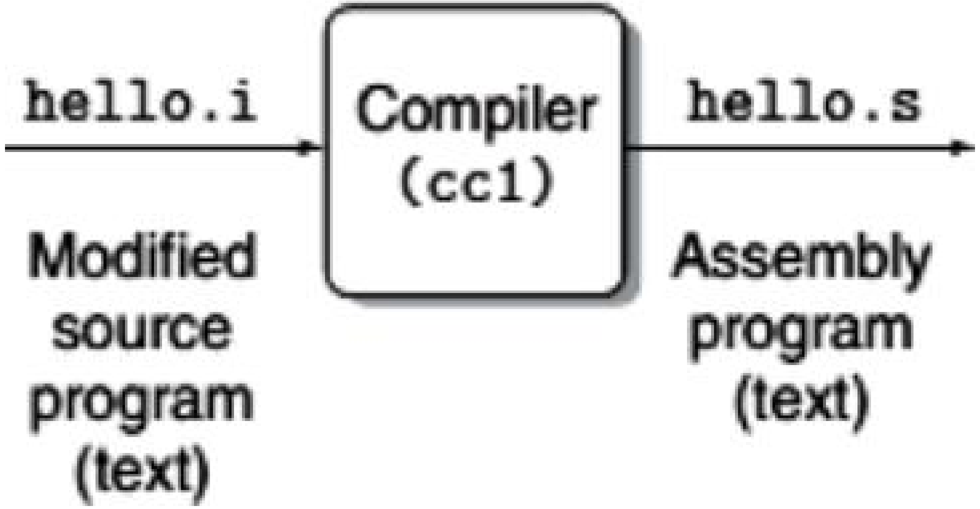
Revisit C Compilation (contd.)

- STEP 1:



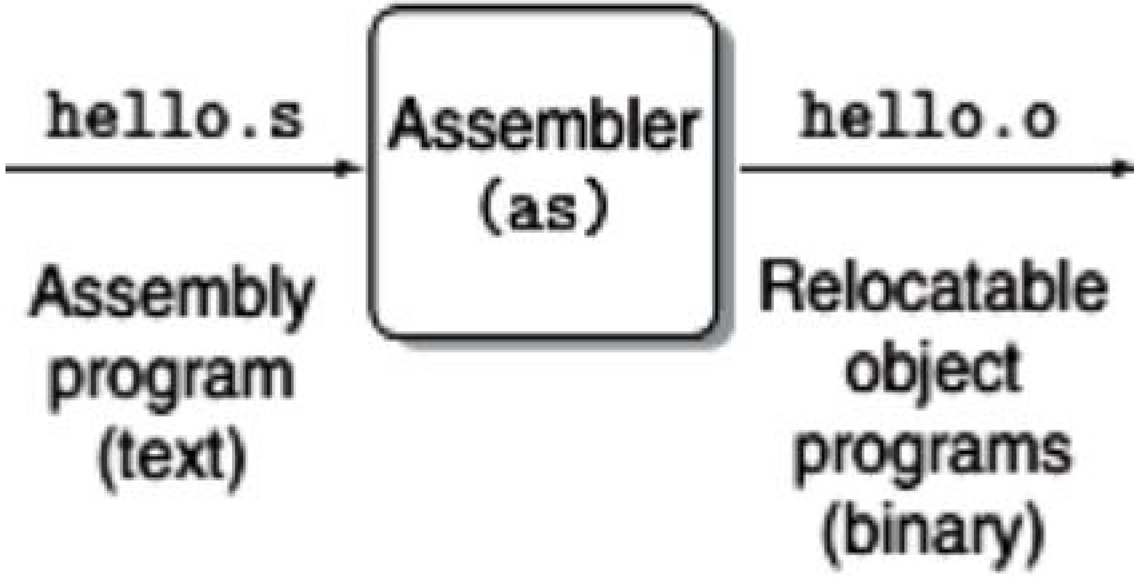
Revisit C Compilation (contd.)

- STEP 2:



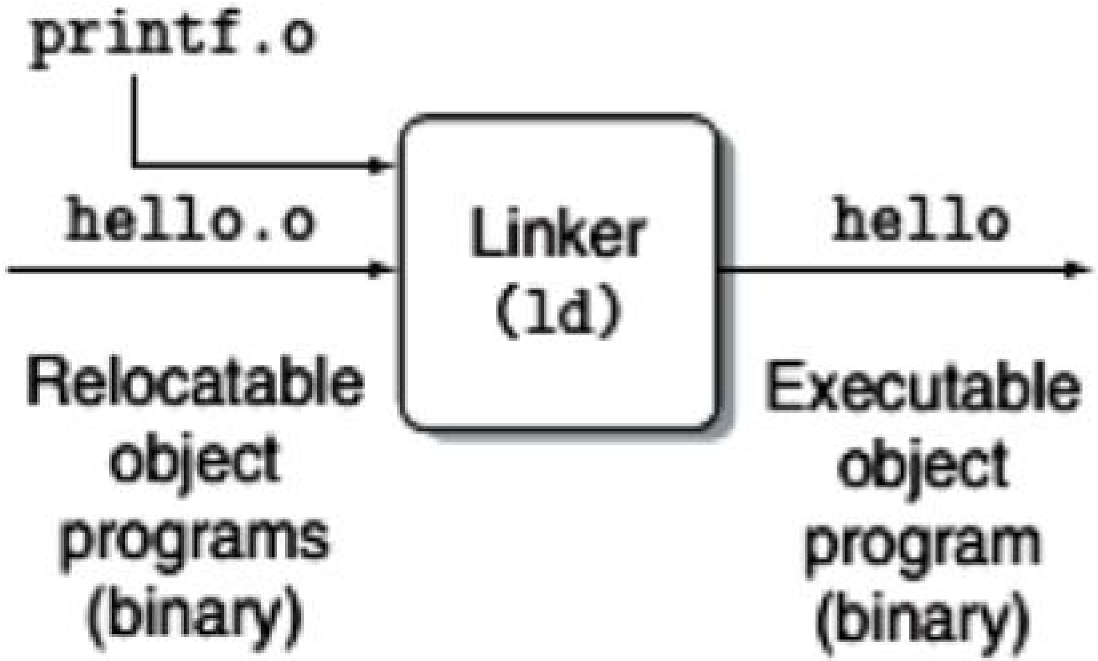
Revisit C Compilation (contd.)

- STEP 3

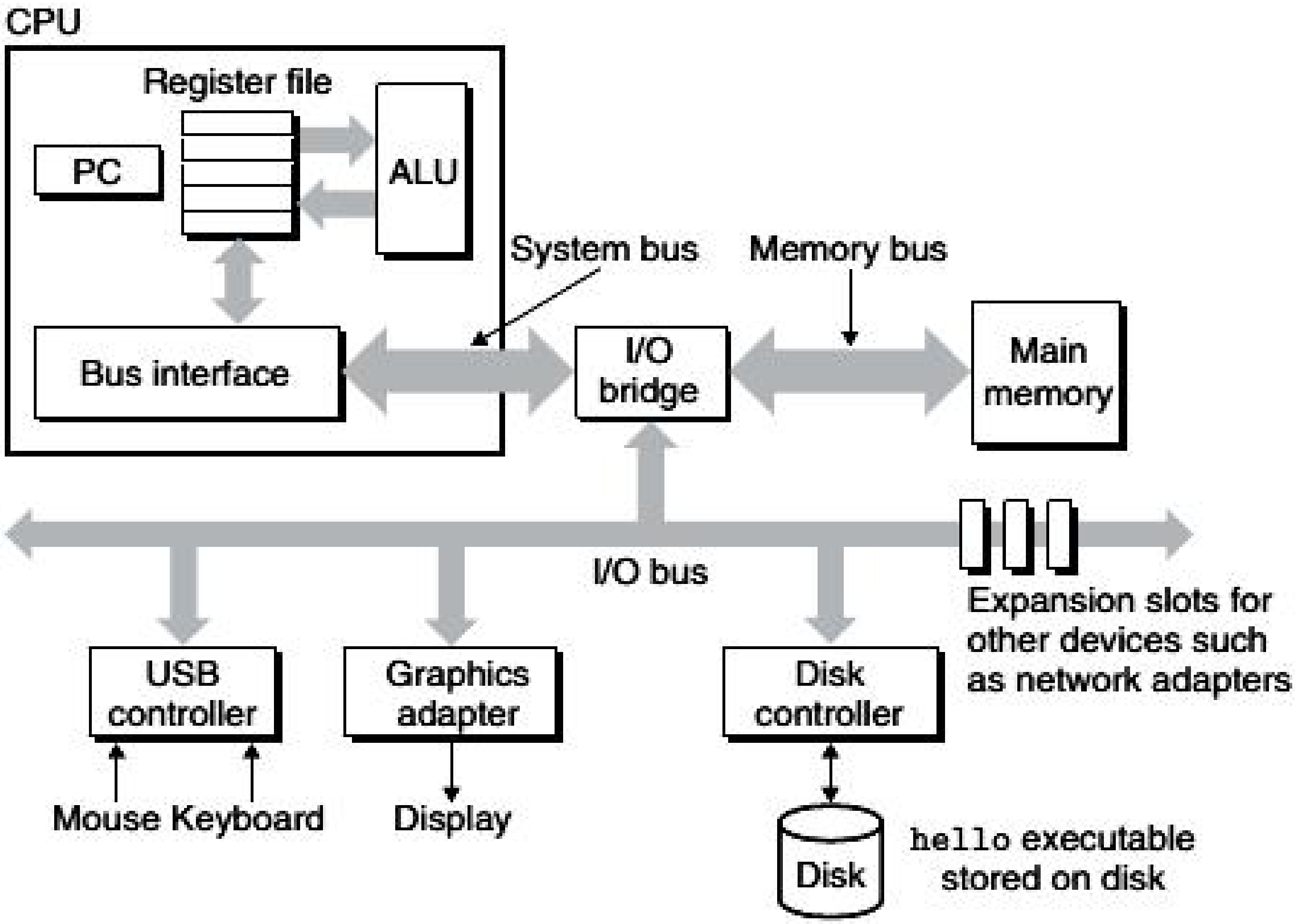


Revisit C Compilation (contd.)

- STEP 4:



Tour of a Computer System



Running a C program

- Compile:

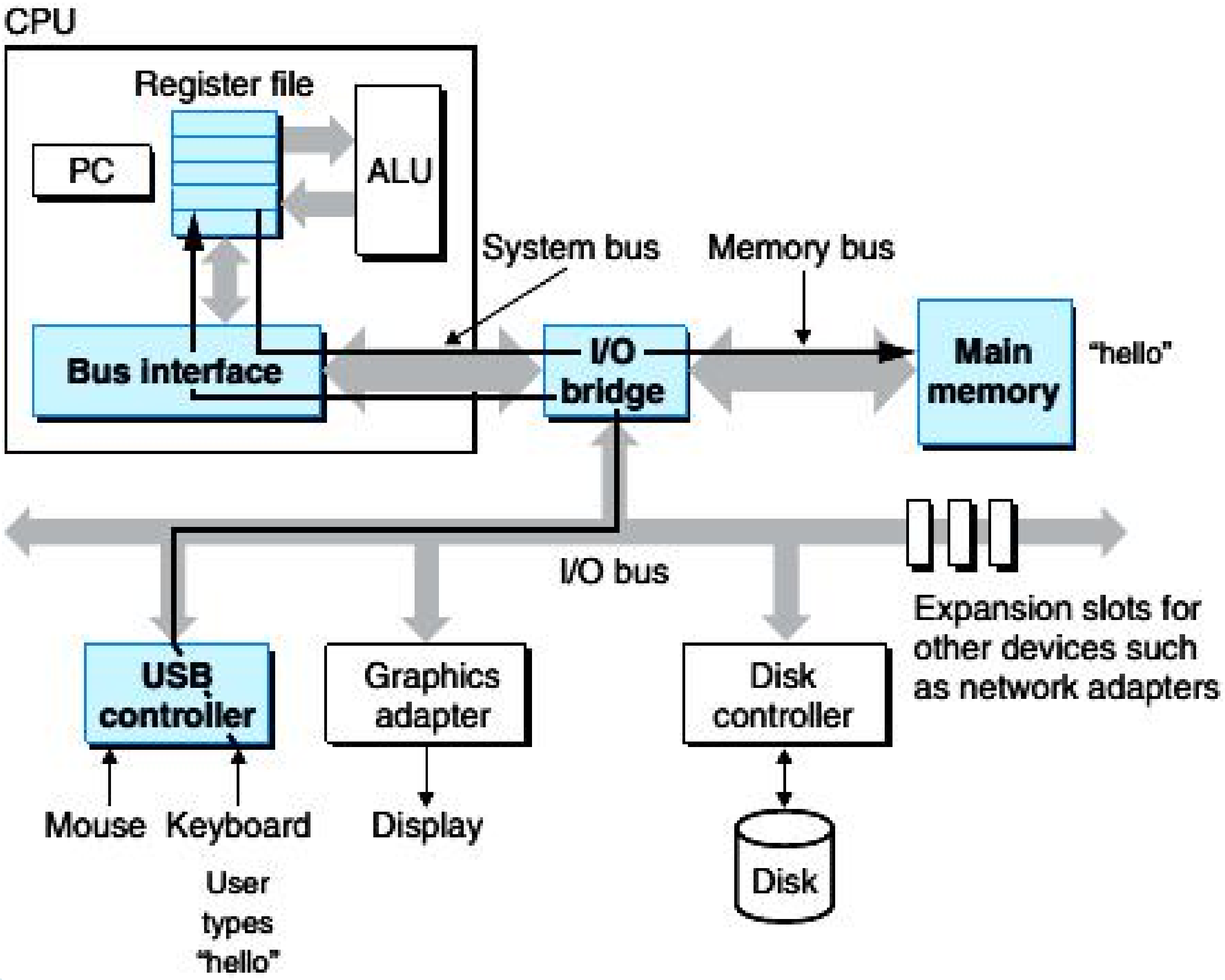
```
$ gcc –o hello hello.c
```

- Run

```
$ ./hello
Hello World
$ _
```

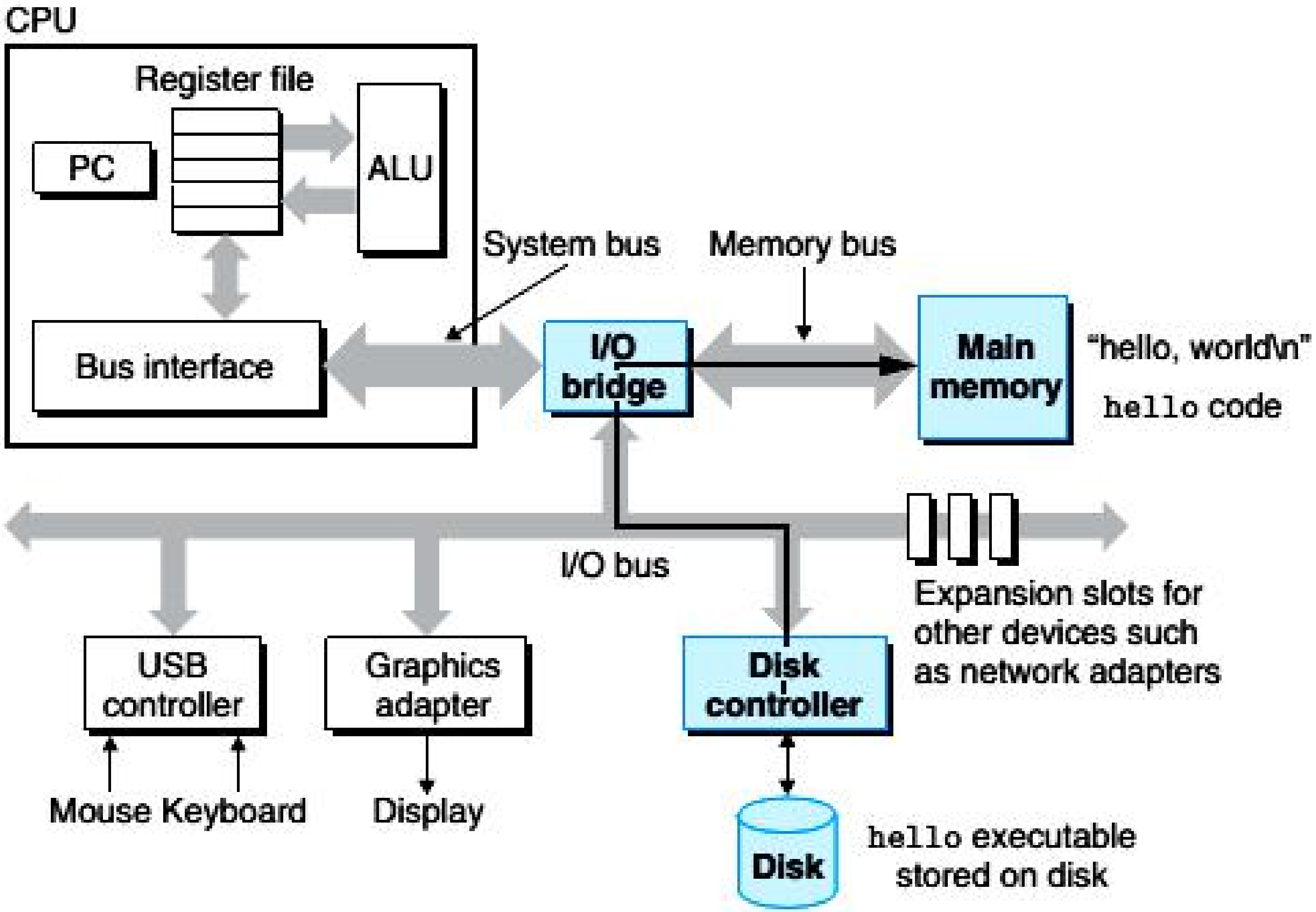
Running a C program (contd.)

- Reading ./**hello**



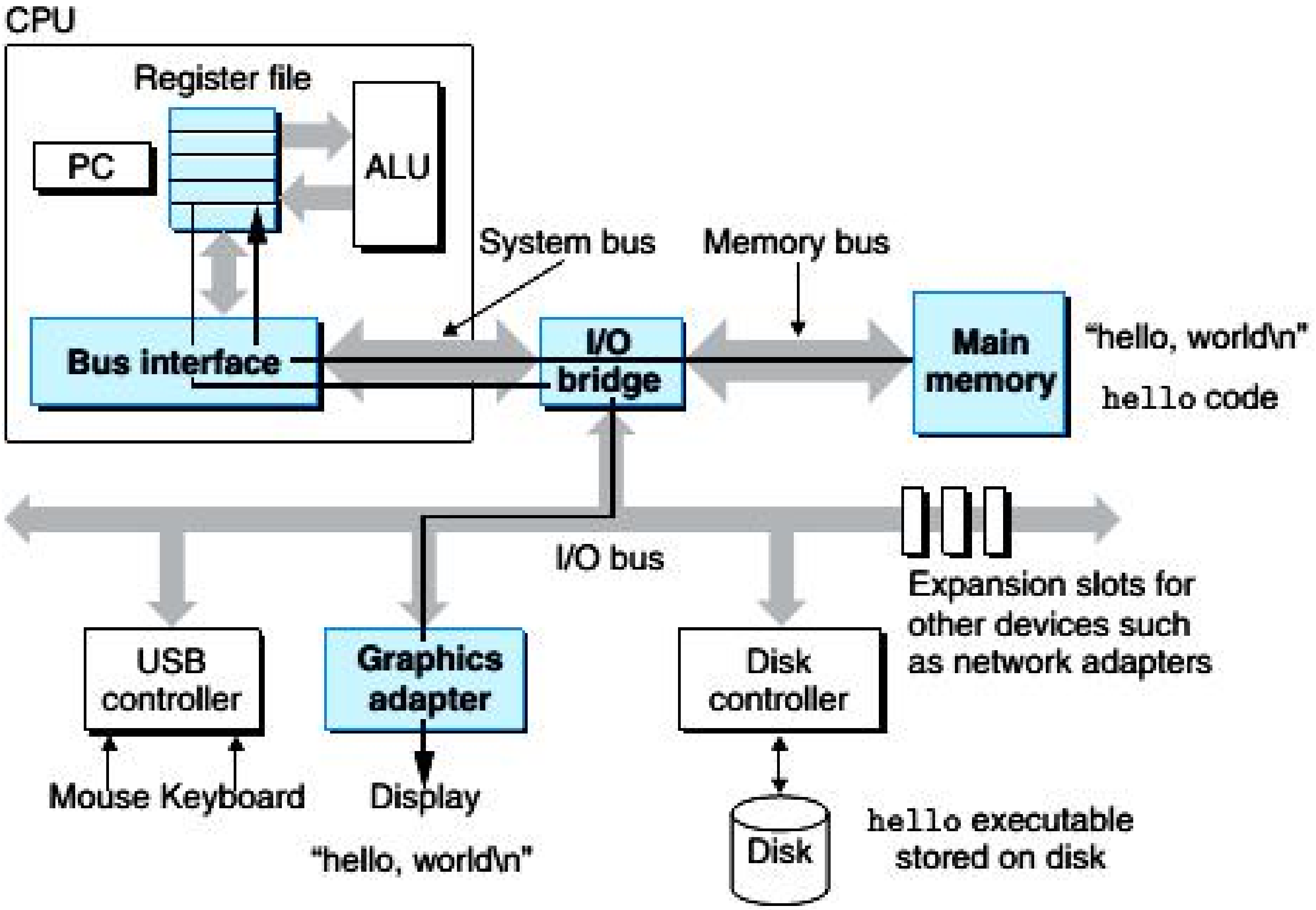
Running a C program (contd.)

- Loading the executable

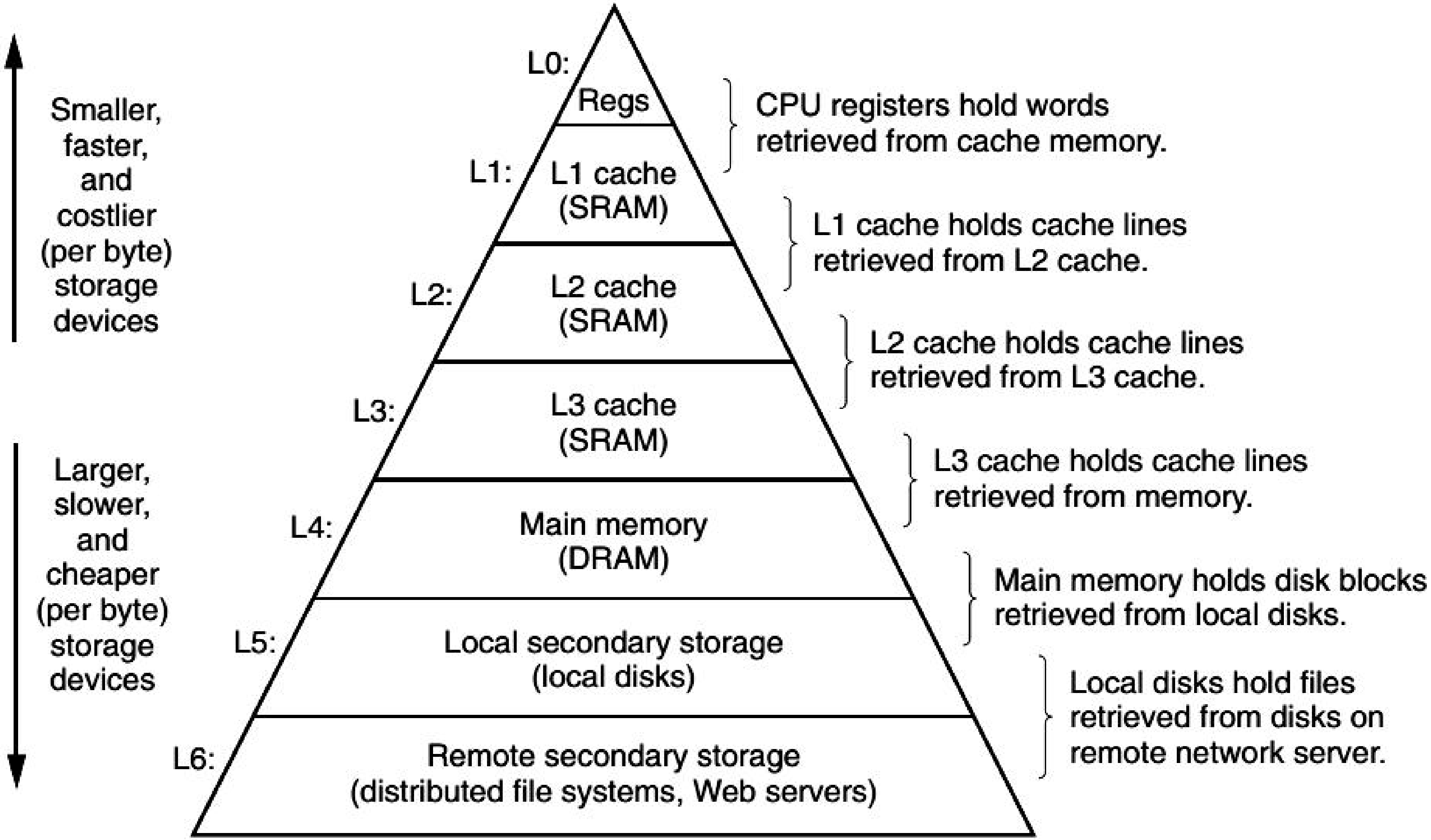


Running a C program (contd.)

- Writing output “String”

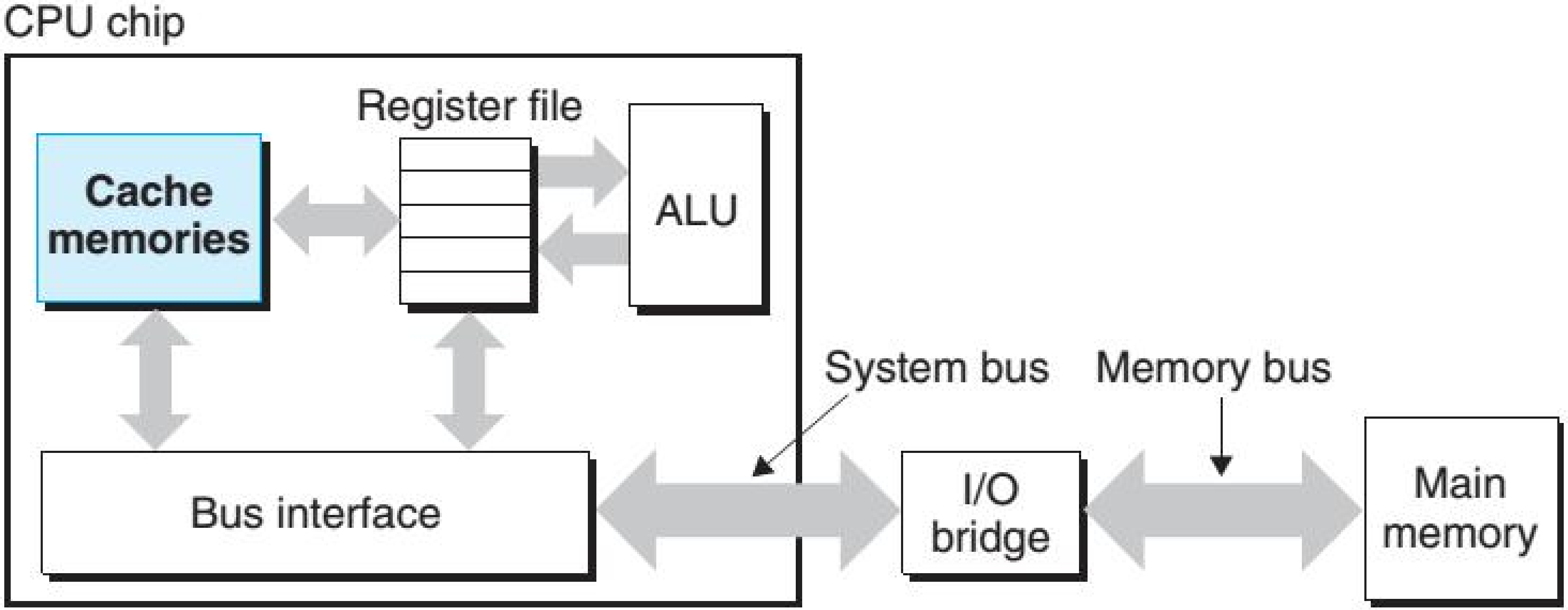


Storage Hierarchy



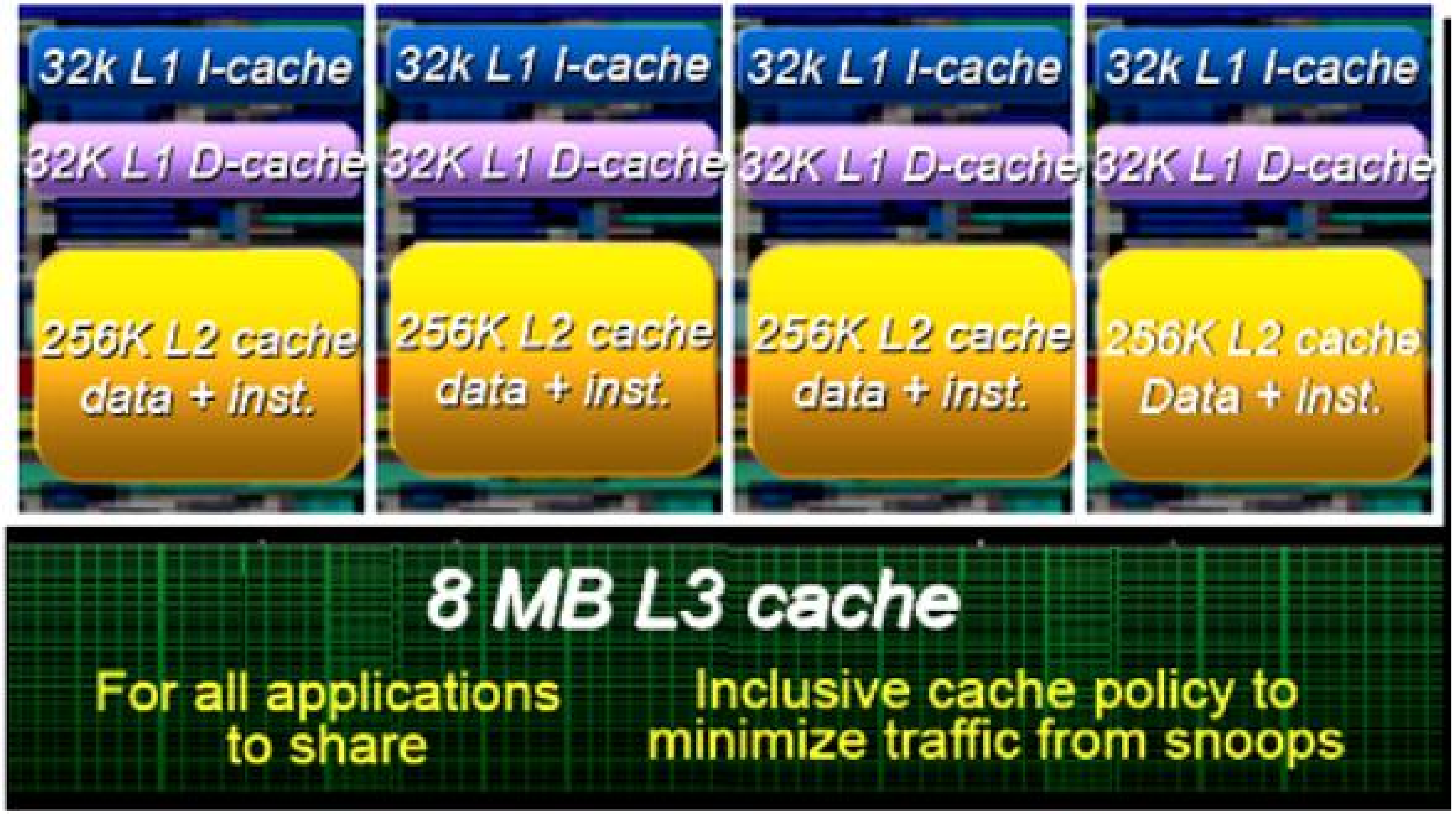
Smaller, faster, and costlier (per byte) storage devices
Larger, slower, and cheaper (per byte) storage devices
L0:
Regs
L1:
L1 cache (SRAM)
L2:
L2 cache (SRAM)
L3:
L3 cache (SRAM)
L4:
Main memory (DRAM)
L5:
Local secondary storage (local disks)
L6:
Remote secondary storage (distributed file systems, Web servers)
CPU registers hold words retrieved from cache memory.
L1 cache holds cache lines retrieved from L2 cache.
L2 cache holds cache lines retrieved from L3 cache.
L3 cache holds cache lines retrieved from memory.
Main memory holds disk blocks retrieved from local disks.
Local disks hold files retrieved from disks on remote network server.

Cache Memory



Cache Memory (contd.)

- Intel Core i7


32k L1 I-cache
32K L1 D-cache
256K L2 cache data + inst.
32k L1 I-cache
32K L1 D-cache
256K L2 cache data + inst.
32k L1 I-cache
32K L1 D-cache
256K L2 cache data + inst.
32k L1 I-cache
32K L1 D-cache
256K L2 cache Data + inst.
8 MB L3 cache
For all applications to share
Inclusive cache policy to minimize traffic from snoops

Cache Memory (contd.)

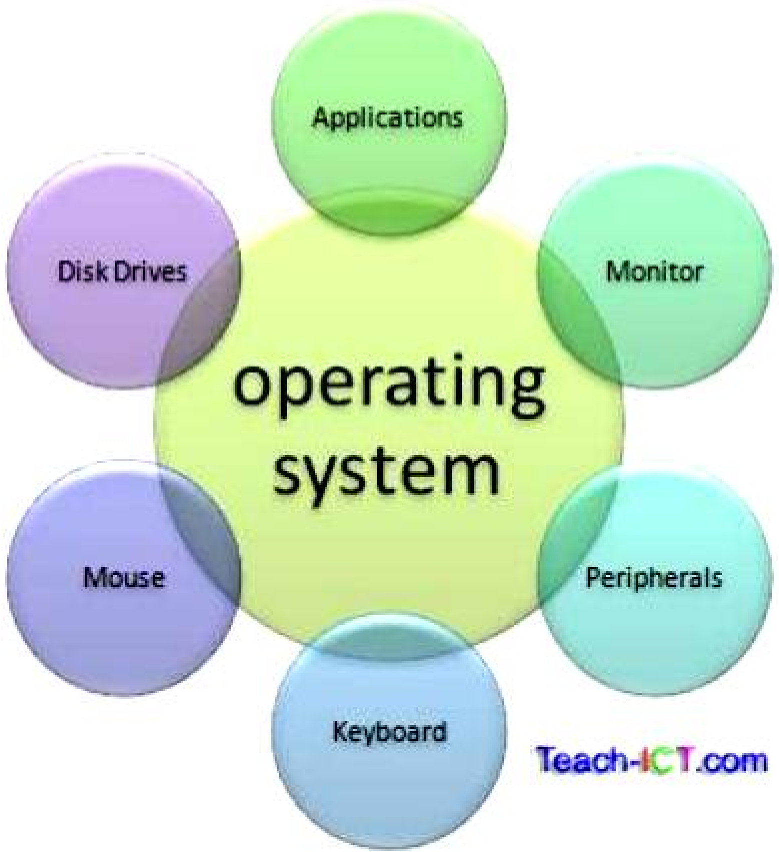
- Cache: L1
  - As fast as the Registers
- Cache: L2
  - 5-10 times faster than main memory
- Cache: L3
  - About 2-times faster

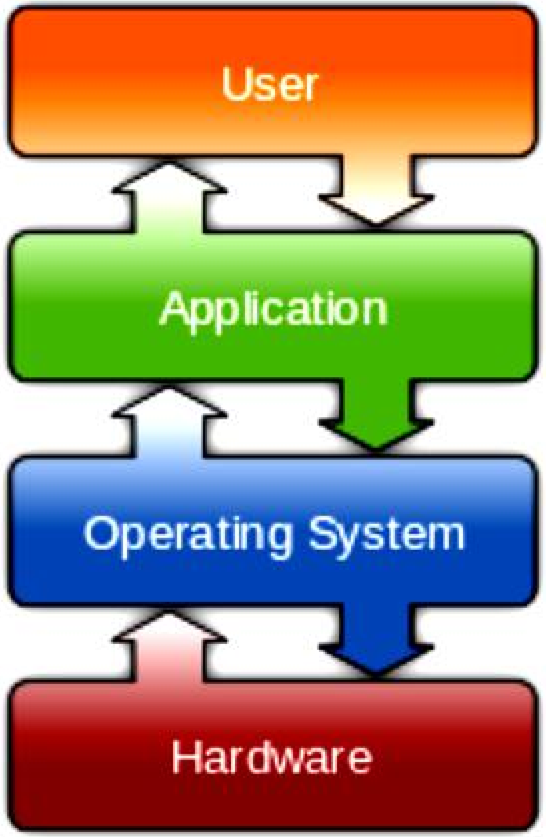
- All types are implemented using SRAM

Operating System Concepts

- A software layer that abstracts away the messy details of hardware into a useful, portable, powerful interface
  - Modules:
    - File-system, virtual memory management, network stack, protection system, scheduler
    - Each of these “subsystems” is a major system of its own!
- Design and implementation has many engineering tradeoffs
  - e.g., speed vs. portability, maintainability, simplicity etc.

Operating System Concepts





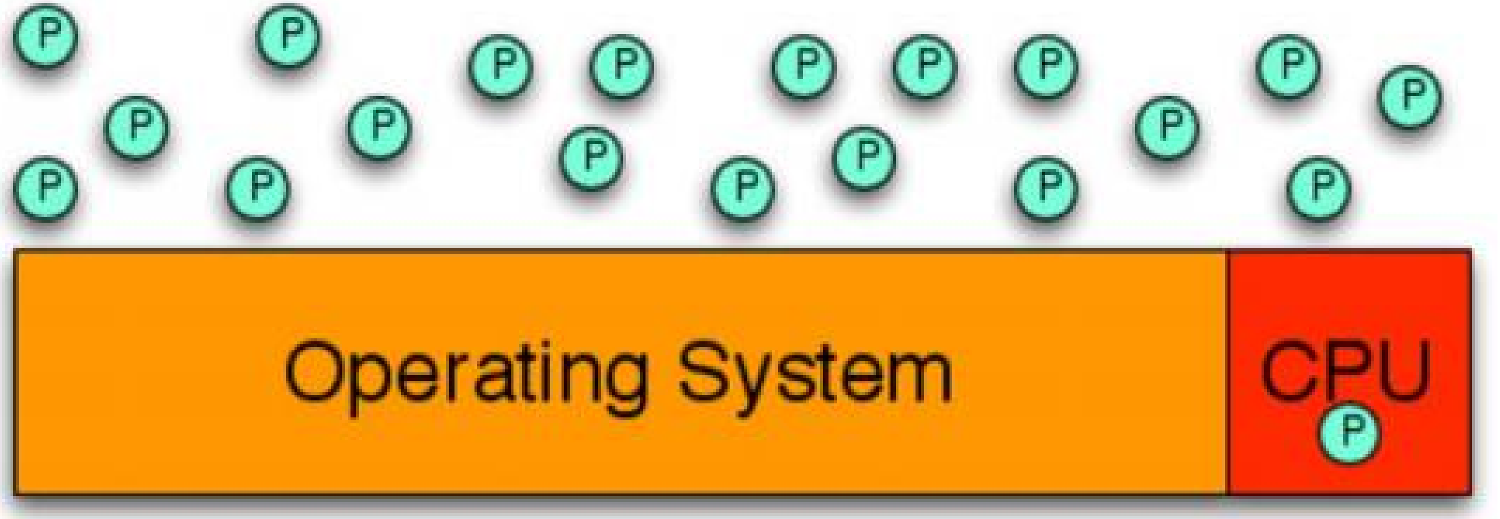
Operating System Concepts (contd.)

- Single-Tasking
- Multi-Tasking
- Multi-User / Time-Shared
- Real-Time
- Distributed
- Embedded

Operating System Concepts (contd.)

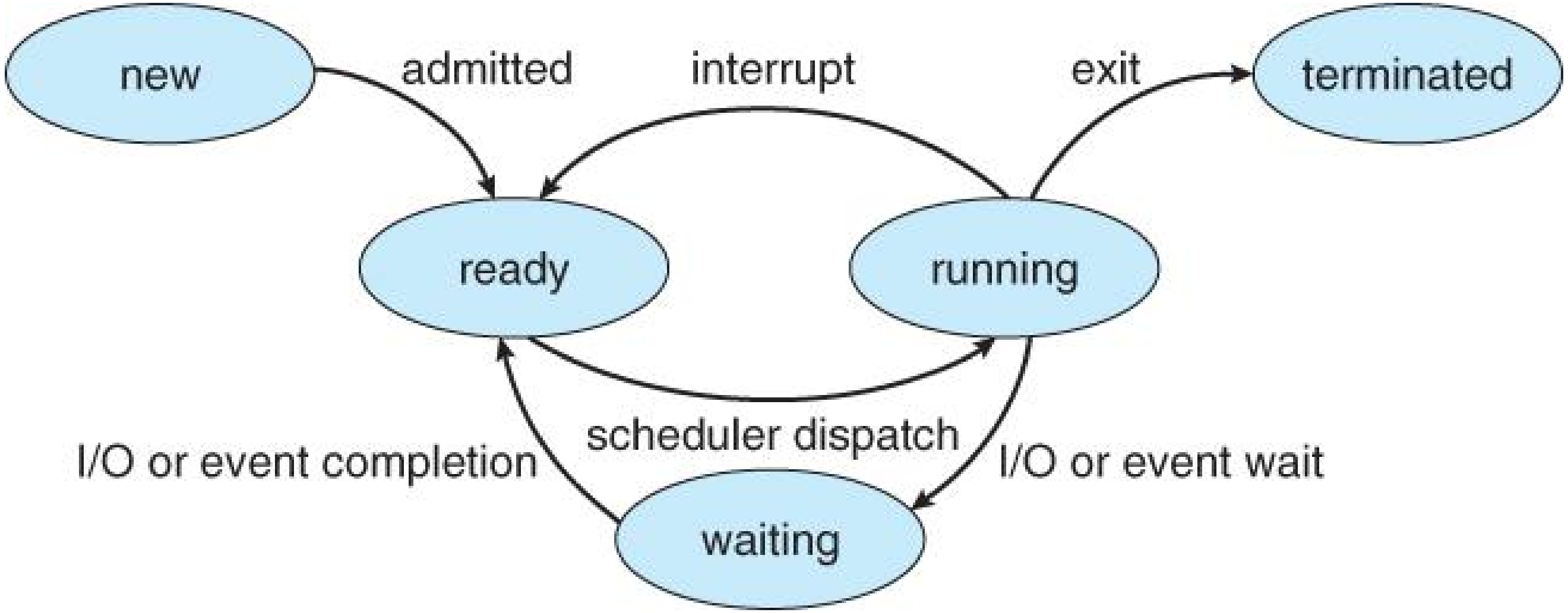
- Process: *Program in Execution*
- Processes are independent programs running concurrently within the operating system

- to see what processes are running on a UNIX system, use the **ps** command



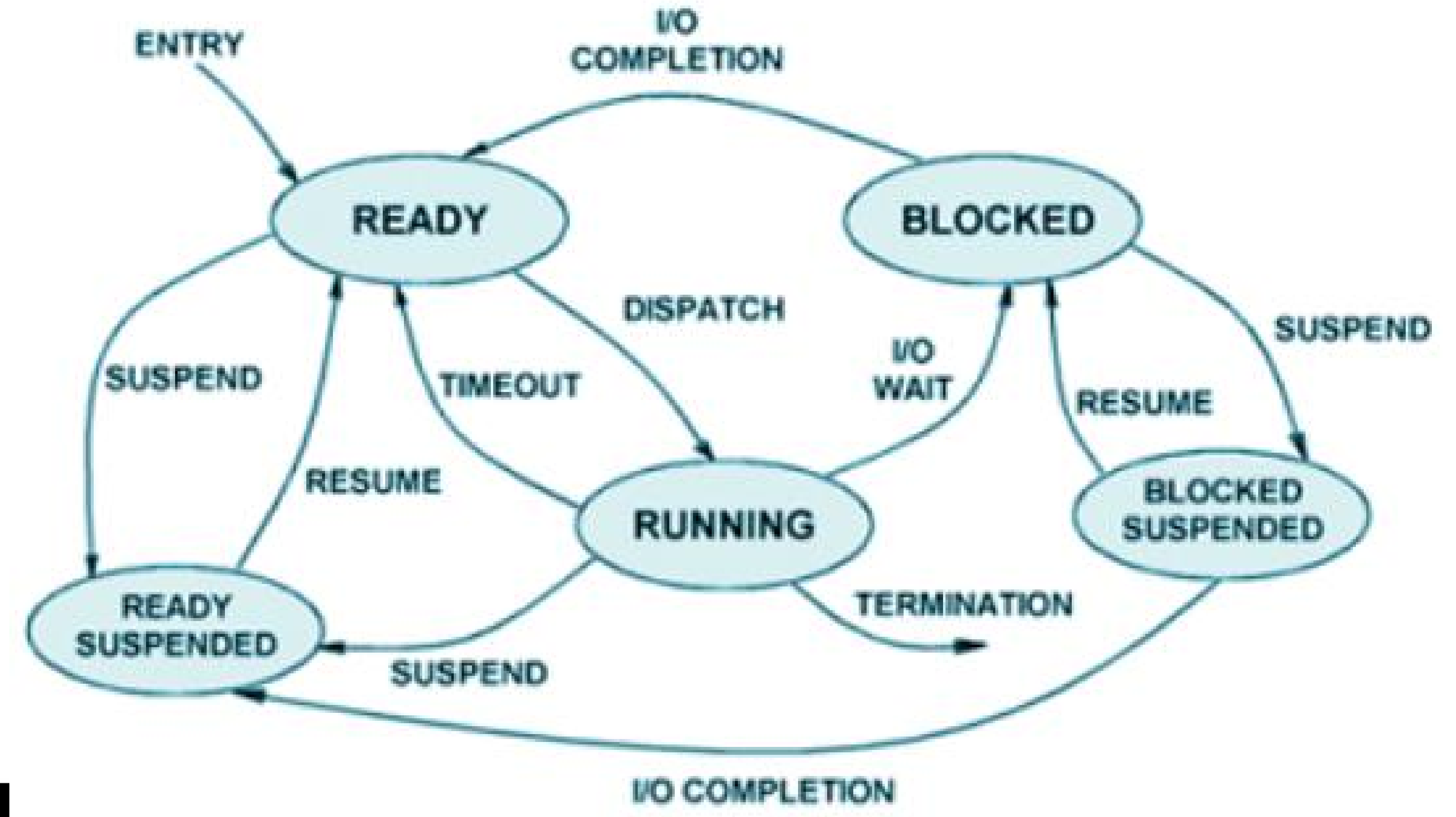
Operating System Concepts (contd.)

- Process States

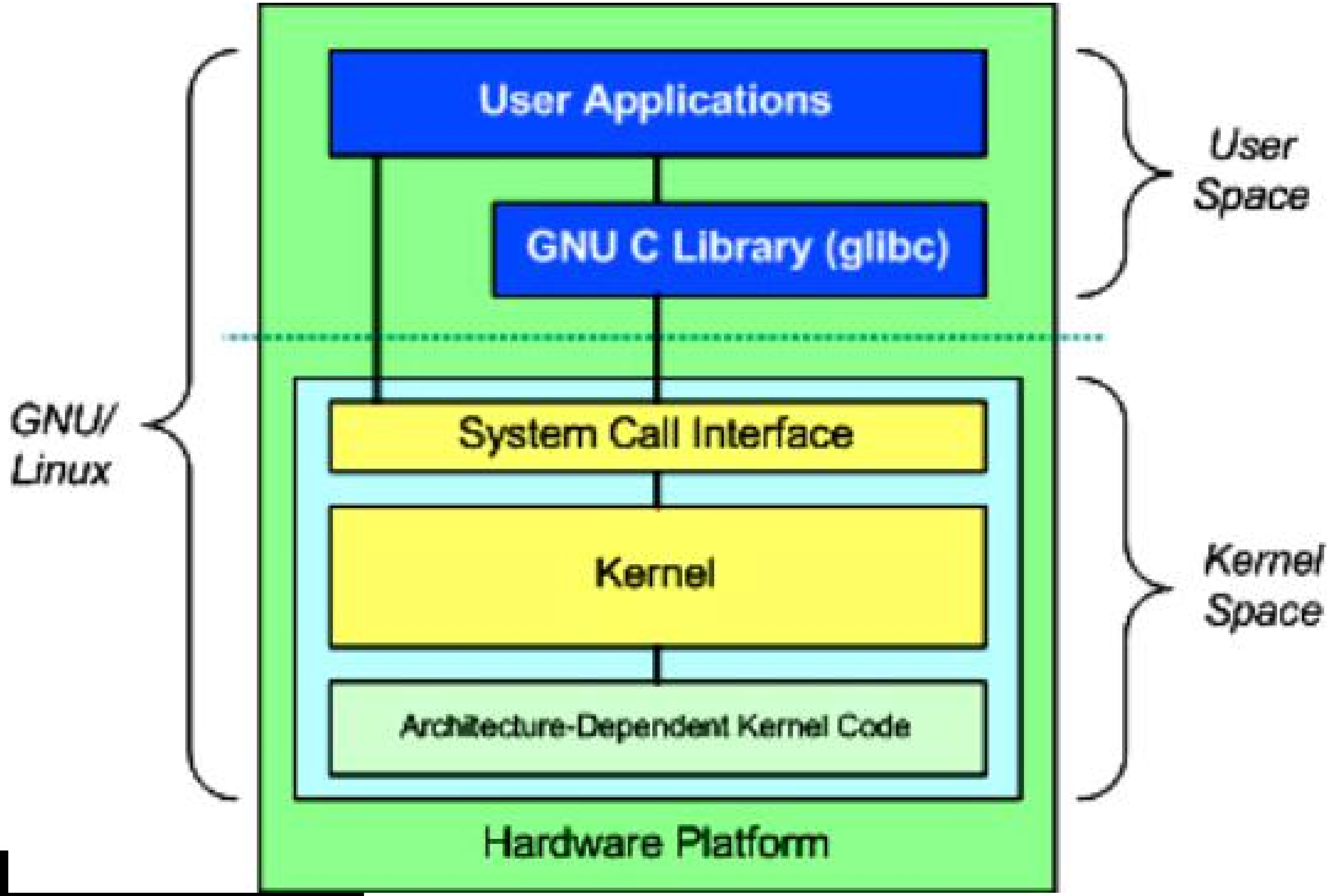


Operating System Concepts (contd.)

- Process States (advanced)

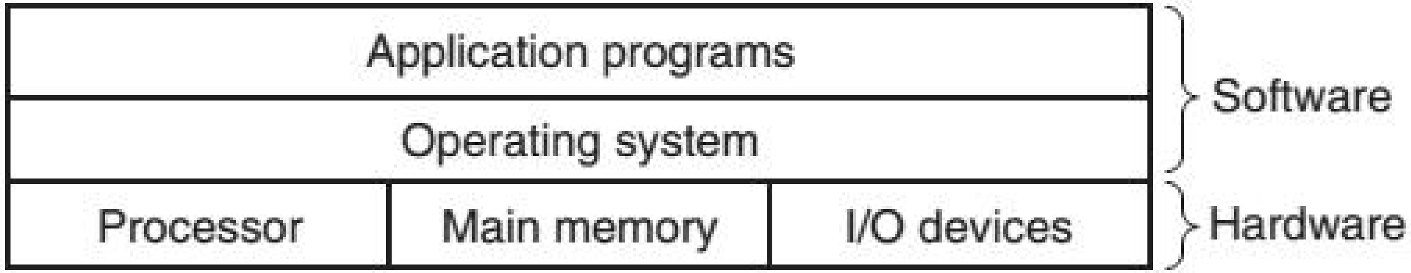


Operating System Concepts (contd.)

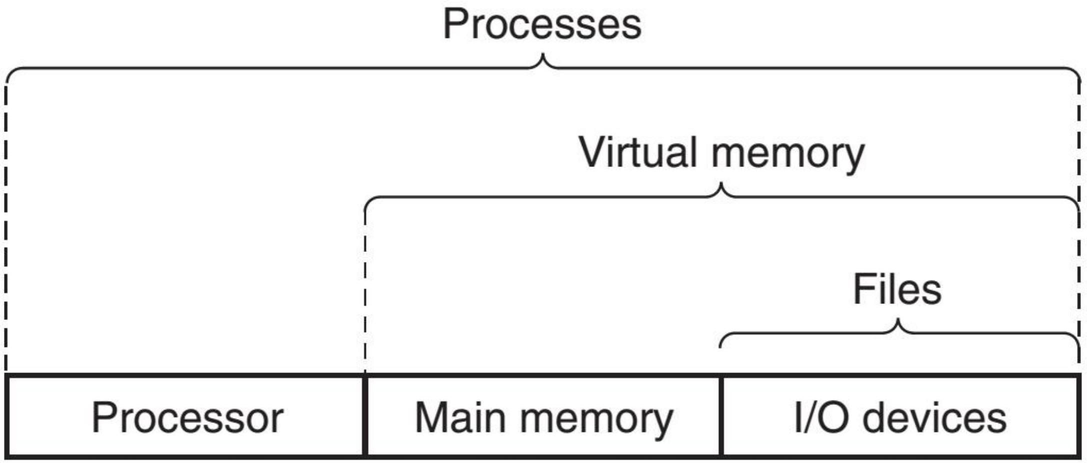


Operating System Concepts (contd.)

- Layered view

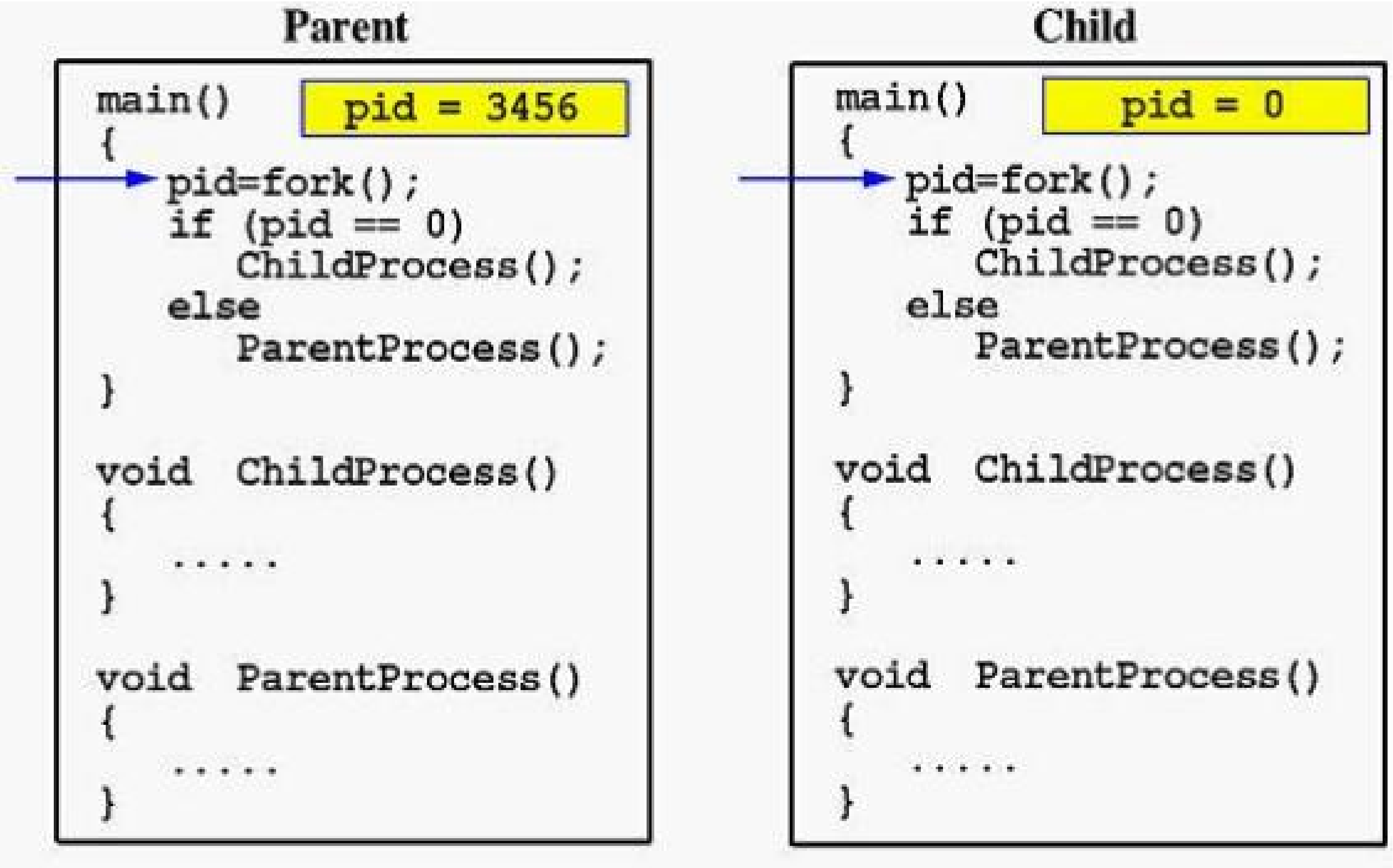


- Abstraction view



Operating System Concepts (contd.)

Process Creation



Parent

```
main()            pid = 3456
{
    pid=fork();
    if (pid == 0)
       ChildProcess();
    else
       ParentProcess();
}

void  ChildProcess()
{
   .....
}

void  ParentProcess()
{
   .....
}
```

Child

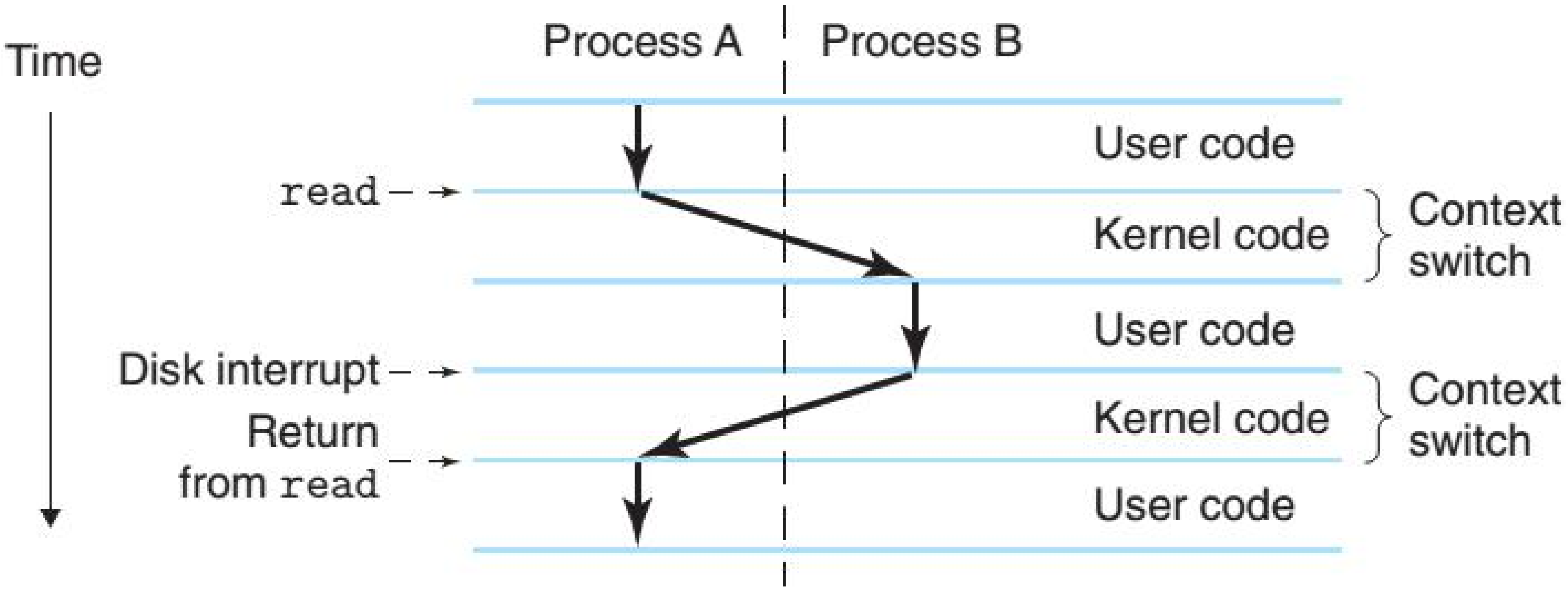
```
main()            pid = 0
{
    pid=fork();
    if (pid == 0)
       ChildProcess();
    else
       ParentProcess();
}

void  ChildProcess()
{
   .....
}

void  ParentProcess()
{
   .....
}
```

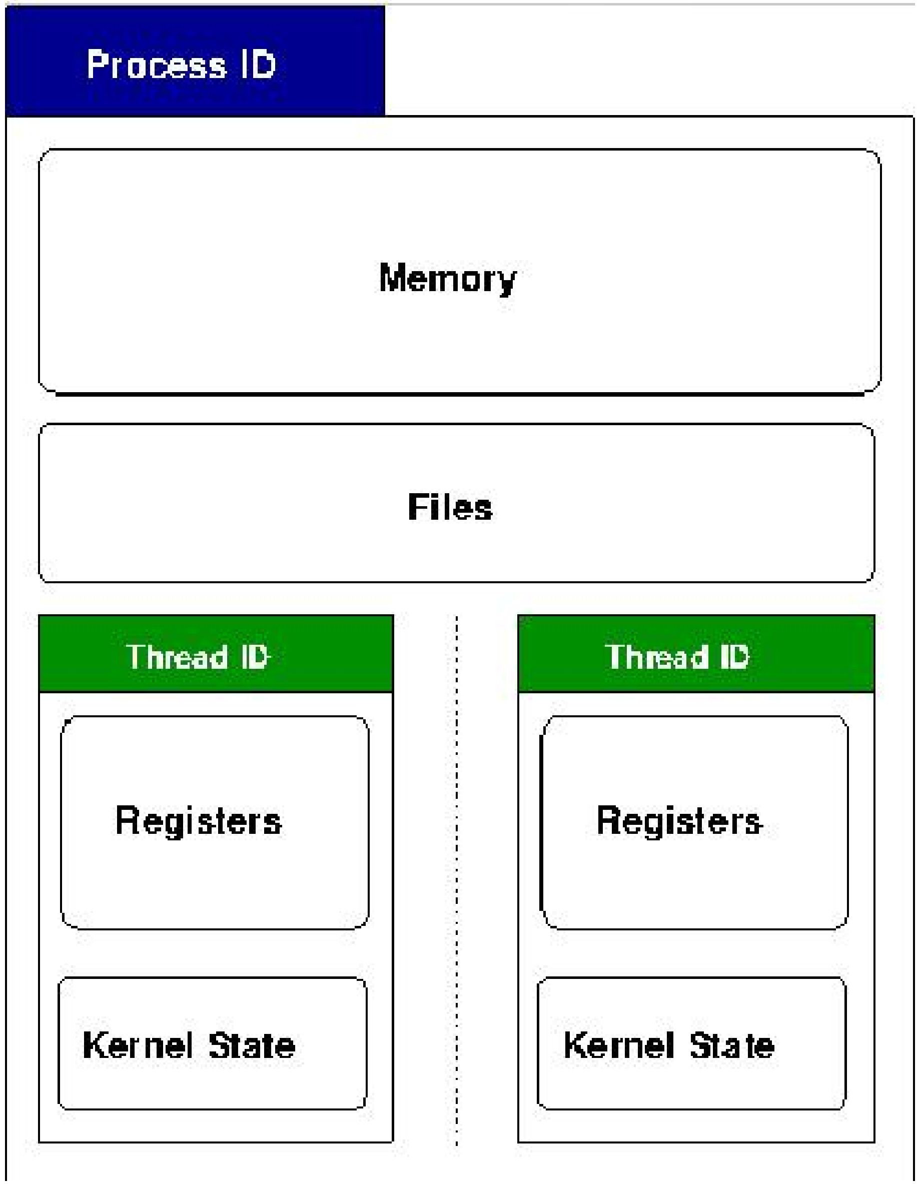
Operating System Concepts (contd.)

Context Switching



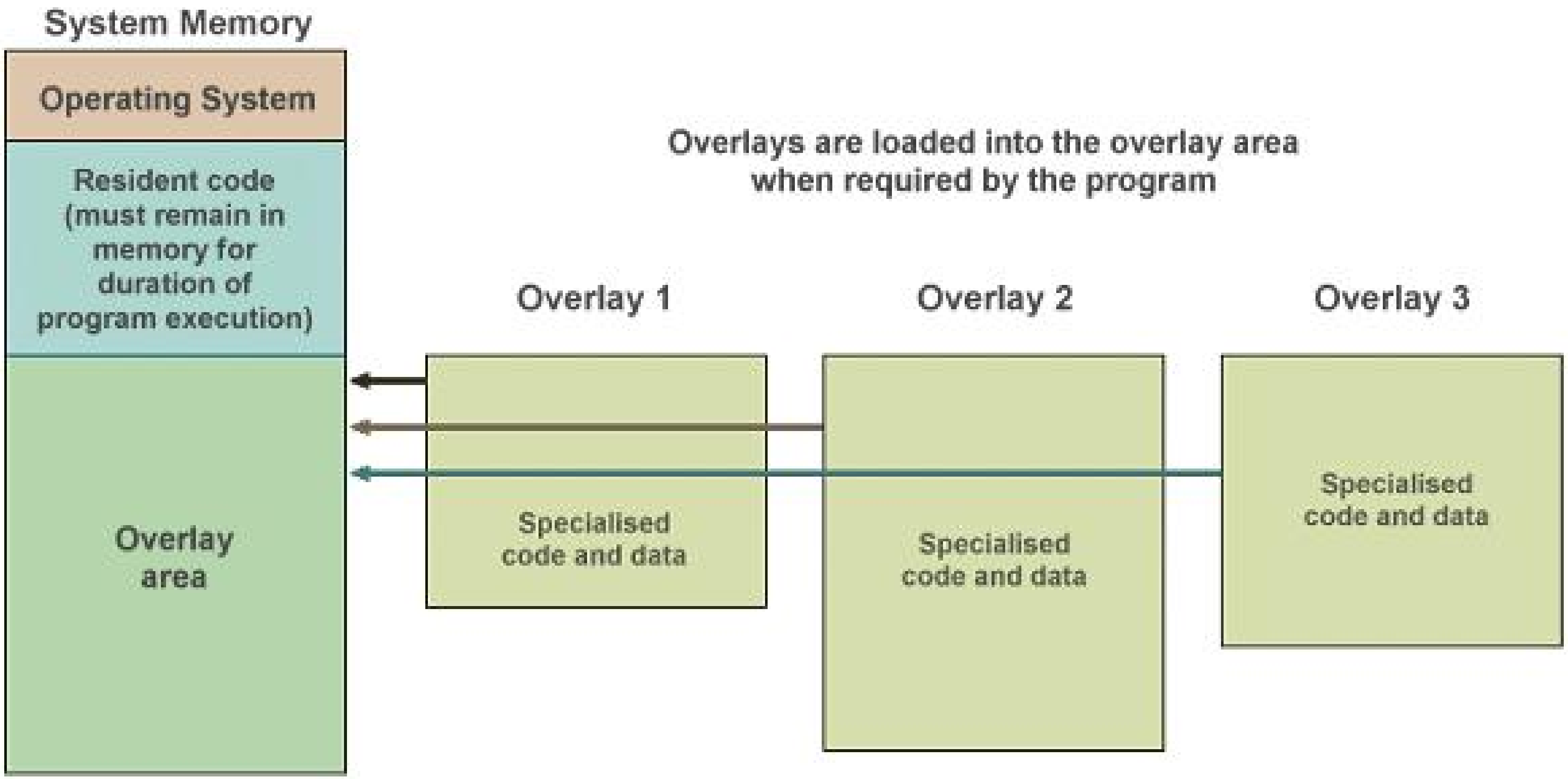
Operating System Concepts (contd.)

- Threads



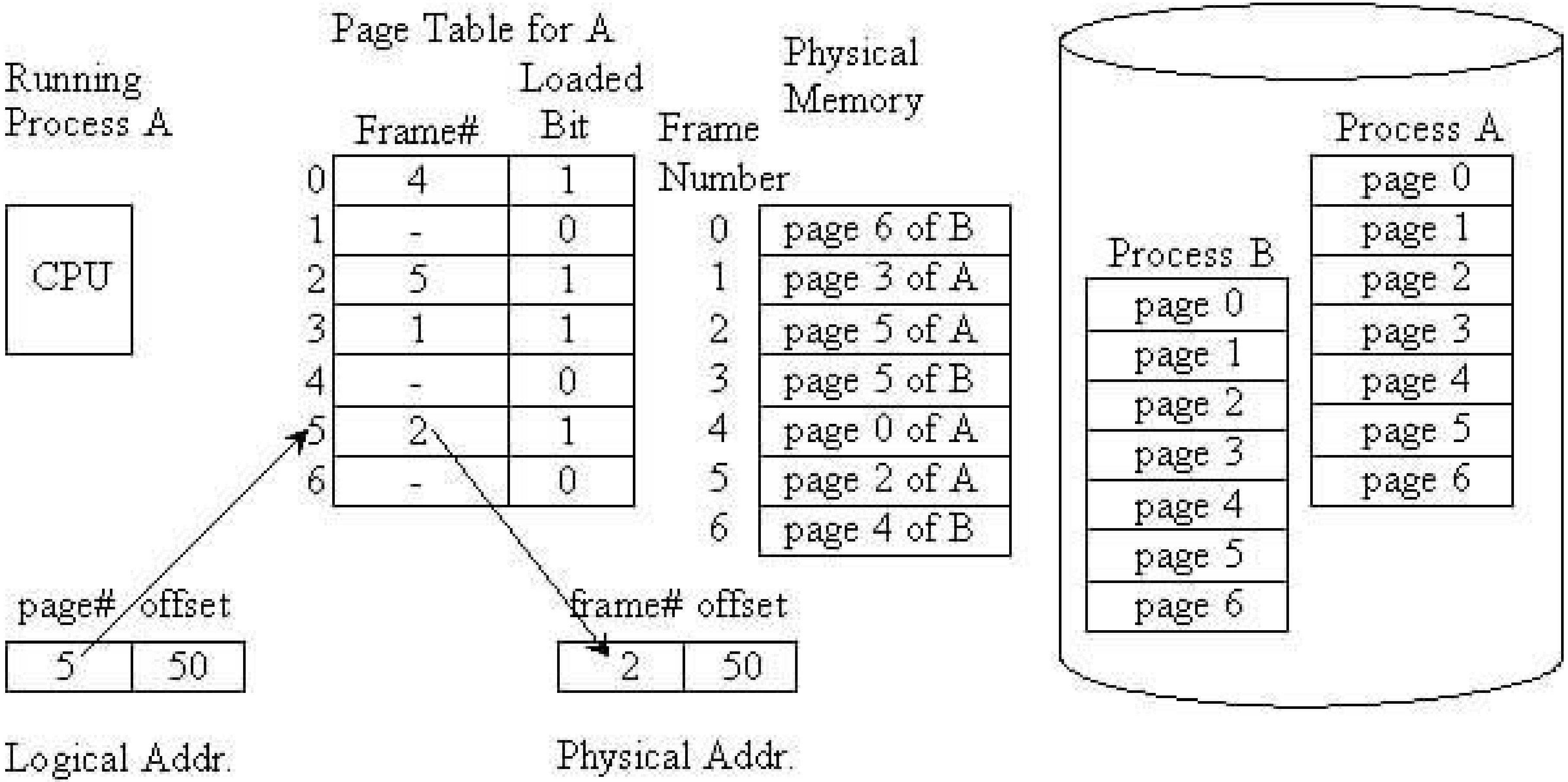
Operating System Concepts (contd.)

Overlays


System Memory
Operating System
Resident code (must remain in memory for duration of program execution)
Overlay area
Overlays are loaded into the overlay area when required by the program
Overlay 1
Overlay 2
Overlay 3
Specialised code and data
Specialised code and data
Specialised code and data

Operating System Concepts (contd.)

Paging



Page Table for A

	Frame#	Loaded Bit
0	4	1
1	-	0
2	5	1
3	1	1
4	-	0
5	2	1
6	-	0

Frame Number	Physical Memory
0	page 6 of B
1	page 3 of A
2	page 5 of A
3	page 5 of B
4	page 0 of A
5	page 2 of A
6	page 4 of B

Process B
page 0
page 1
page 2
page 3
page 4
page 5
page 6

Process A
page 0
page 1
page 2
page 3
page 4
page 5
page 6

page#	offset
5	50

frame#	offset
2	50

Operating System Concepts (contd.)

Virtual Memory

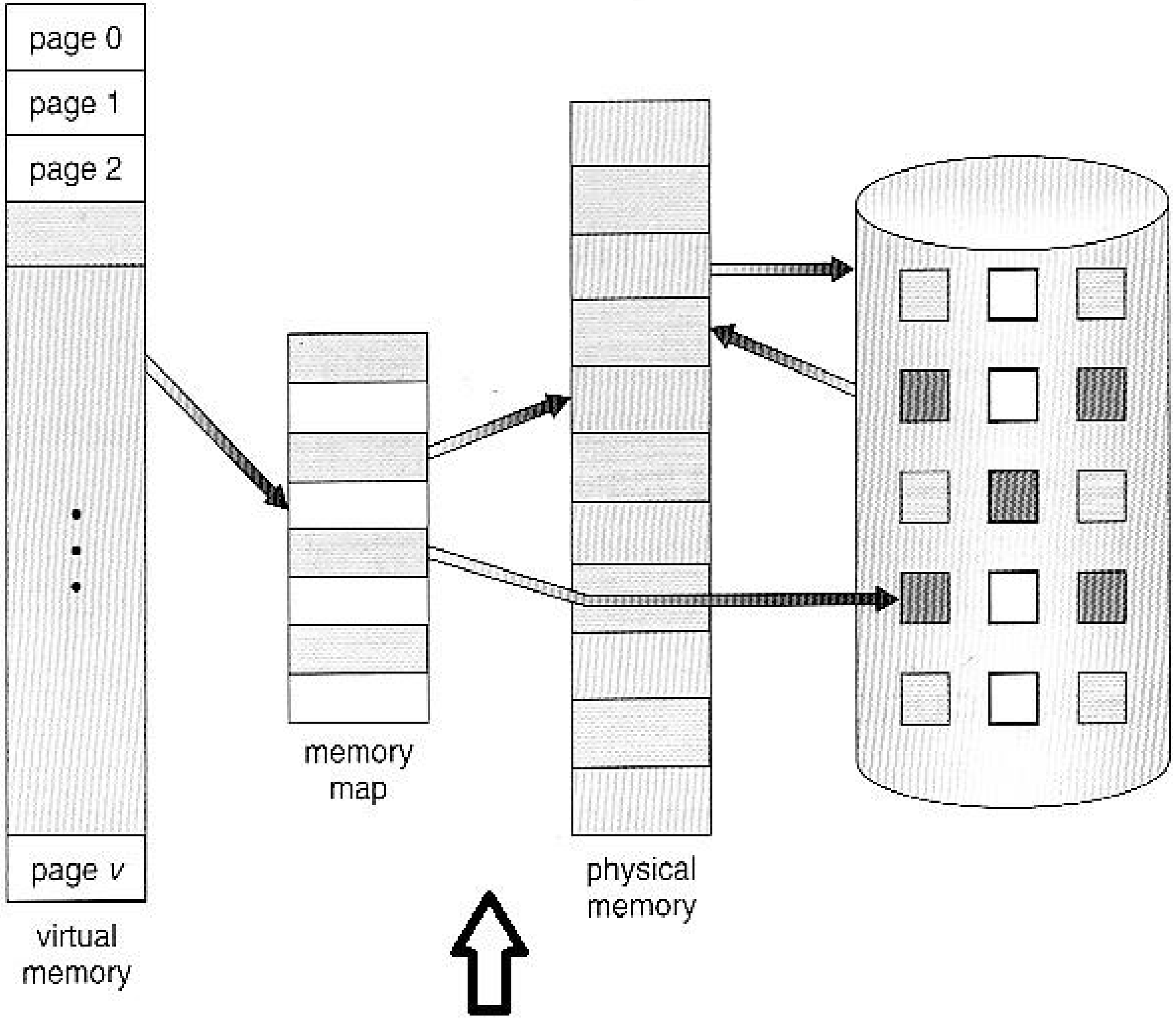
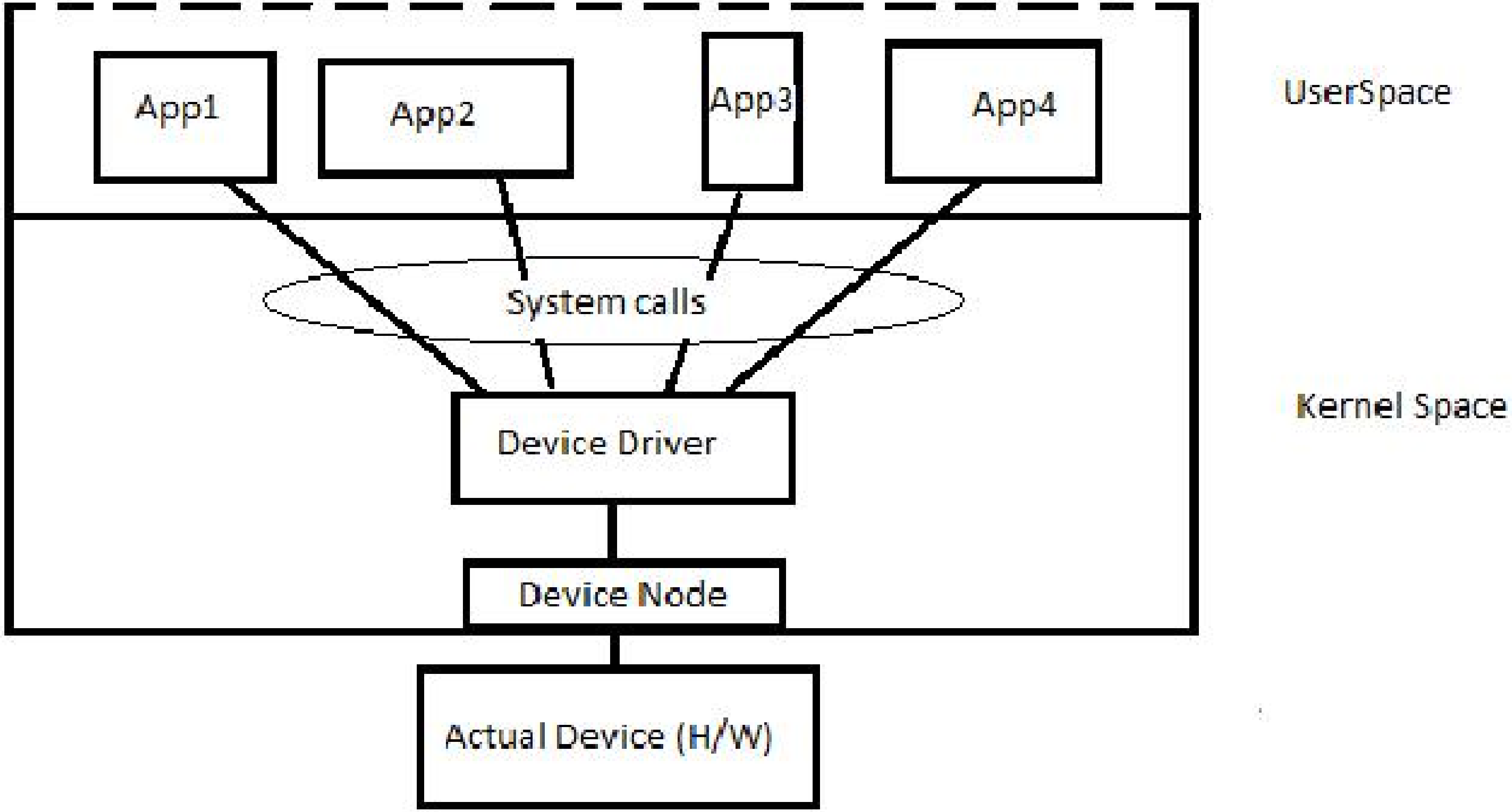


Diagram showing virtual memory that is larger than main memory

Operating System Concepts (contd.)

Files:

- Files are FILES
- Folders are FILES
- Devices are also FILES



Operating System – Tasks

- Memory management
- Device management
- Processor management
- I/O programs
- File systems
- Searching / sorting
- Scheduler
- Libraries

Other System Software

- Compiler-compiler
- Cross compiler
- Cross assembler
- Emulator
- Preprocessor
- Macro-processor
  - MASM, NASM, TASM, VAX

Programming Considerations

- Development and Production environments
- Making Software Portable
- Software over Internet
- Programs as Components
- Quick-and-Dirty Programming
- Dynamic/Flexible/Adaptive Software

Take Away

- Concurrency
  - Multiple simultaneous activities

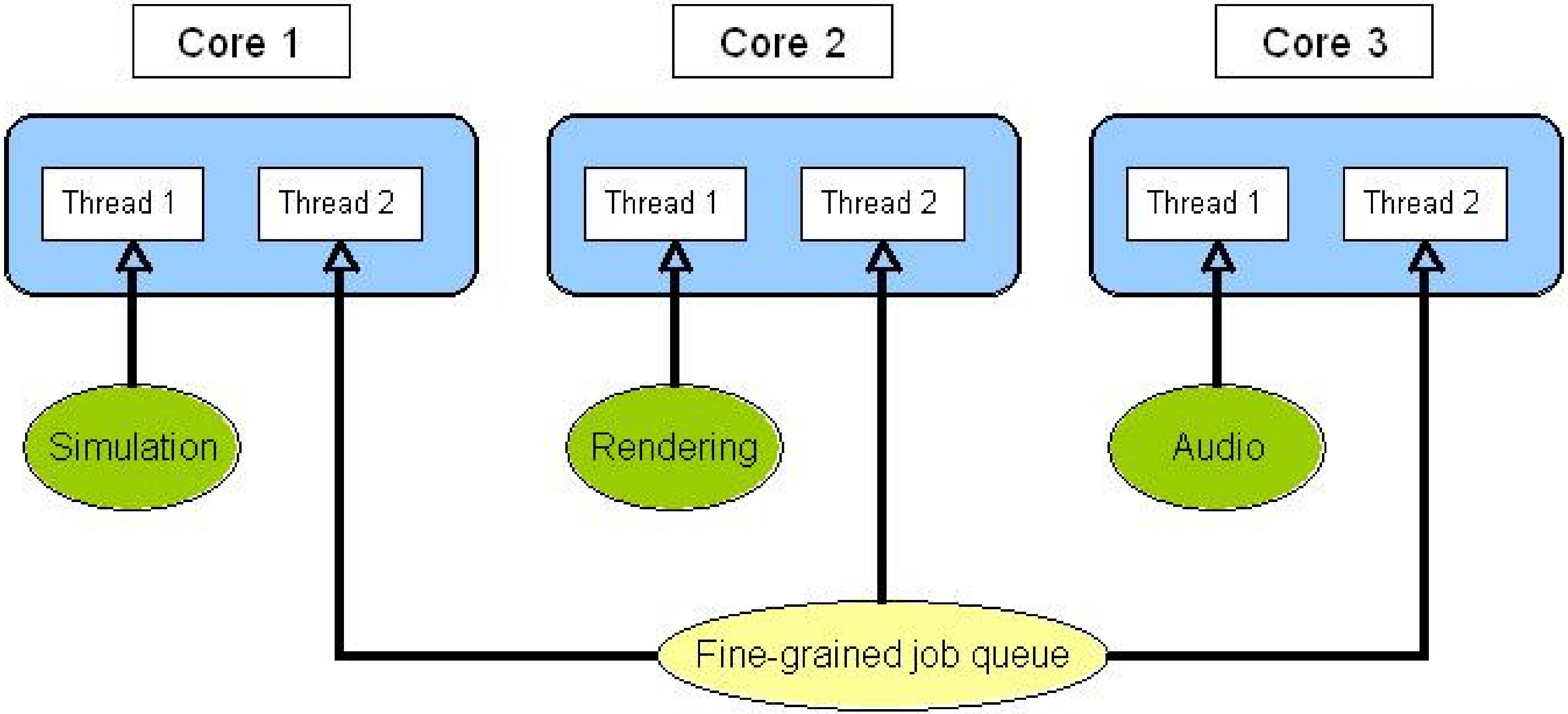
- Parallelism
  - Concurrency to make systems run faster

Take Away (contd.)

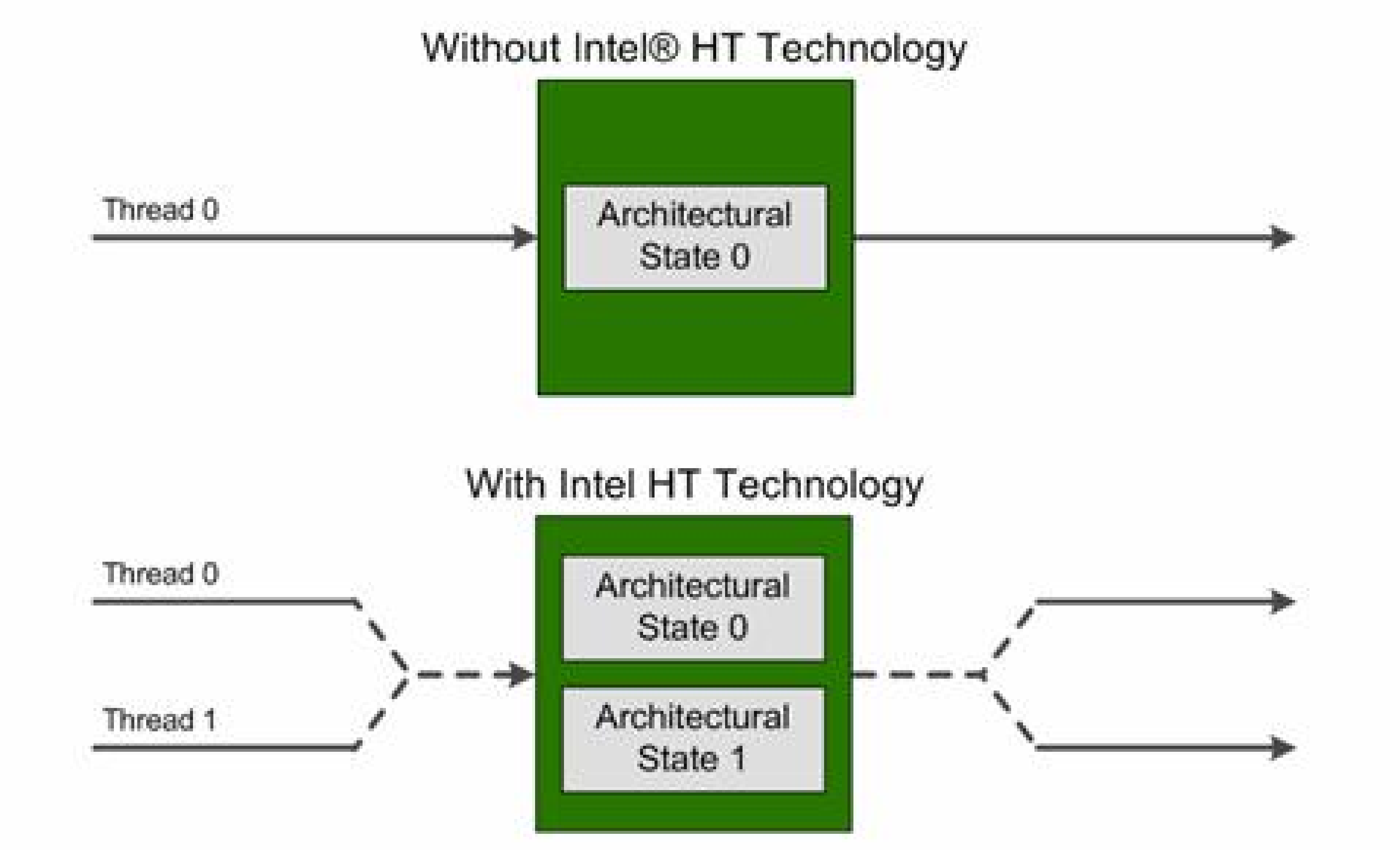
Hardware Threads:

- Thread-level Concurrency
  - Uni-processor
  - Multi-processor
- Hyper-threading
  - Simultaneous multithreading
  - Multiple: PC, other registers
  - Single: ALU, FPU

Take Away (contd.)



Take Away (contd.)



Take Away (contd.)

Instruction-level Concurrency:

- Previous Systems:
  - 1 instruction takes 3-4 Machine Cycles

- Superscalar
  - System that can execute more than ONE instructions per Cycle

GOOGLE CLASSROOM

- You may have received the request, accept it
- All the materials and submissions will be done through this
- In cse you haven't received the request, pl. inform the course coordinator