Introduction

COMP 3361: Operating Systems I

Winter 2015

http://www.cs.du.edu/3361

1 COMP3361 Winter 2015

- **Operating Systems I**
- Instructor
  - RINKU DEWRI
  - rdewri@cs.du.edu
  - http://cs.du.edu/~rdewri
  - Office Hours:
    - Aspen North 102C :: WR Noon - 1:30 PM
- GTA
  - Thomas Hamill
  - thomasha@cs.du.edu
  - Office Hours:
    - Aspen North 300C :: M 2:00 - 3:00 PM, R 9:30 - 11:30 AM

Website

http://www.cs.du.edu/3361

3 What You Should Be Familiar With

- Computer Organization
- Systems Programming using C/C++ in Unix

```
int is_valid_address(long addr) {
    long *base;
    long limit;

    base  = (long *)(0x400);
    limit = 0x200;

    if ((long)base+limit <= addr)
        return 0;
    else
        return 1;
}
```

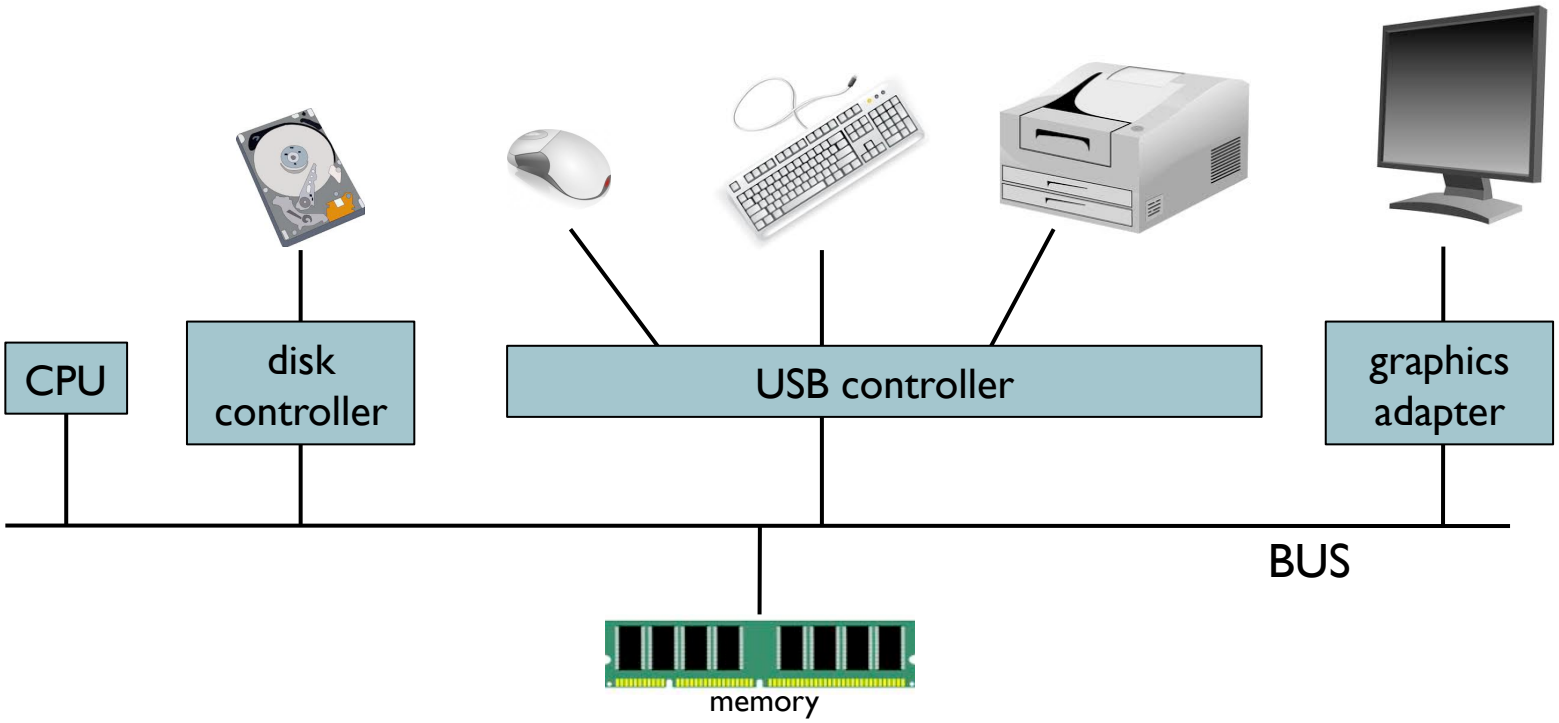
Will this code compile?

No. Why?

Yes. What does it do?

4 Organization of a Computer System

- One or more CPUs and a number of device controllers
  - connected through a common bus
  - the bus provides access to shared memory



CPU

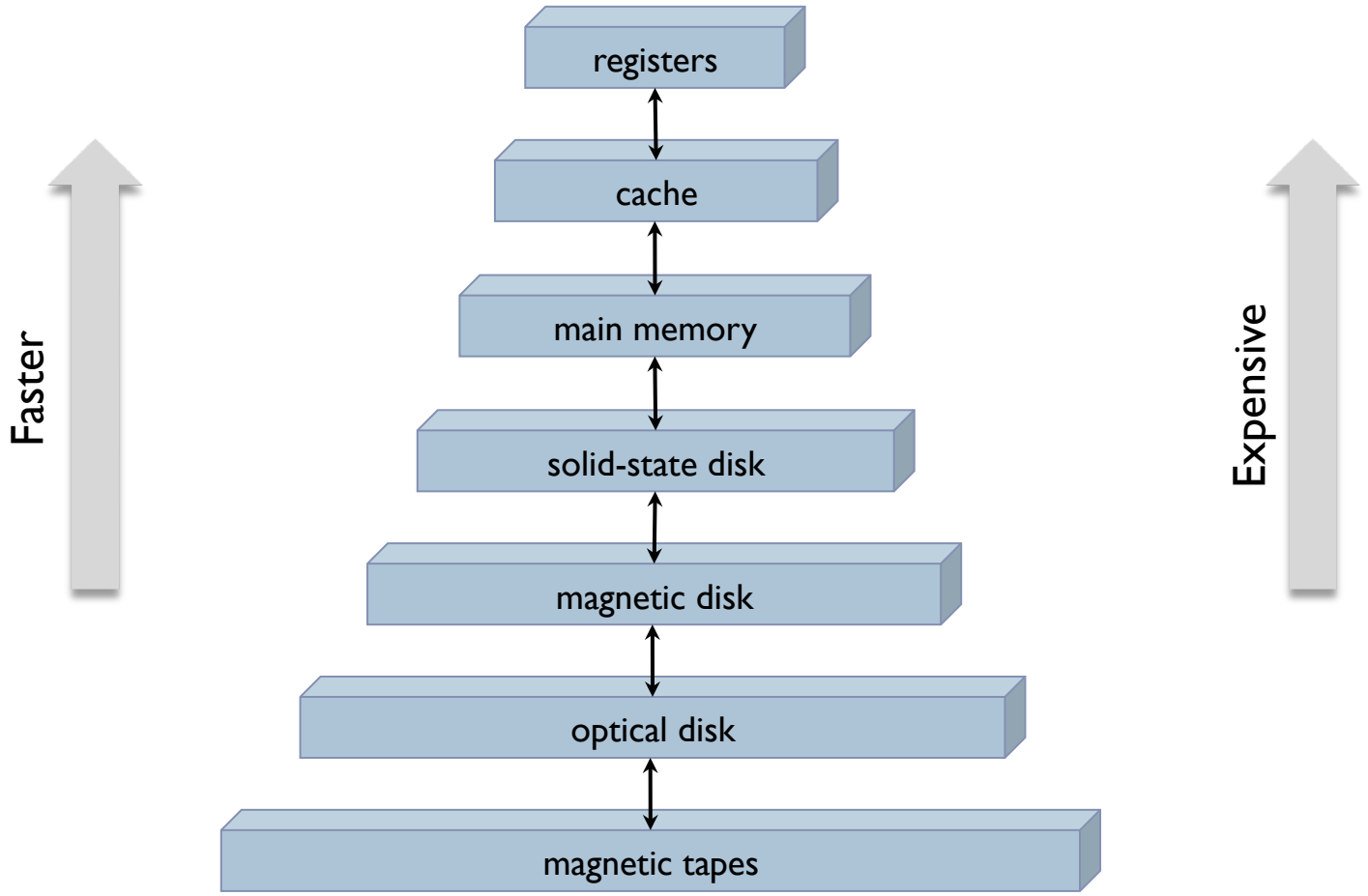
- Can execute a fixed set of instructions
  - e.g. ADD, MOV, CLI, BTS, BSWAP, LIDT and hundreds more
- Basic cycle: fetch → decode → execute
  - has *pipeline* organization (each unit can run in parallel)
- Works with a
  - few general purpose registers (EAX, EBX, ECX, ...), and
  - few special purpose registers (CS, SS, SP, EIP, EFLAGS, ...)
- Can support hyperthreading and have multiple cores

Kernel Mode and User Mode

- Modern CPUs provide multiple modes of operations, also called **rings**
- Kernel (or supervior) mode: **Ring 0**
  - CPU can execute any supported instruction and access every feature of the hardware
- User mode: **Ring 3**
  - certain privileged instructions and direct access to hardware are not allowed
- Other modes (rings) may be available in the CPU, but not necessarily used

Memory

- A hierarchy of storage options



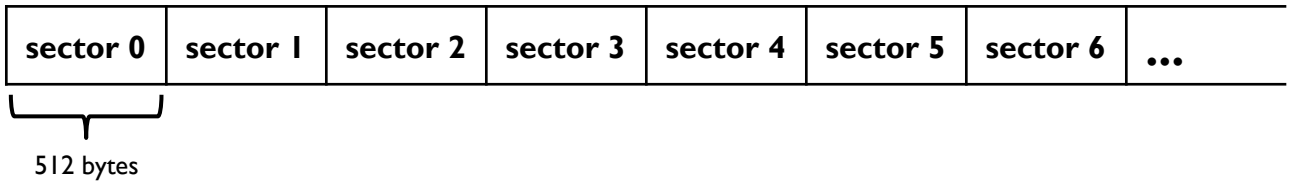
Main Memory (RAM)

- A fast and cheap (yet volatile) storage option for the CPU
  - CPU can address each and every byte of main memory individually
- CPU flags determine how addressing is performed
  - flat mode: 0x1000 (= the 4096th byte in RAM)
  - real mode: 0x0100:0x0000 (= 0x1000)
  - protected mode: 0x001B:0x0100 (= segment 2, offset 0x100)
- From now on, when we say *memory*, we mean *main memory*

9 Disk

- Large non-volatile storage option
- Can be made of mechanical or electronic parts
- Addressing locations on the disk
  - complex: cylinder, head, sector
  - easy: logical block addressing (LBA)

- Our abstraction: a disk is an array where each index can hold 512 bytes (called a **sector**)
  - easily implemented using LBA, which moderns disks support
  - cannot write less than 512 bytes!



I/O Devices

- Input/Output devices
  - keyboard, mouse, printer, display, ethernet, ...
- Consists of the device controller and the device itself
  - the controller accepts commands and performs the necessary hardware operations on the device
- Each device also comes with a **device driver**
  - the software that knows what commands to send to the controller for specific operations
  - provided by the device manufacturer
- A device driver talks to a controller by writing appropriate values to certain registers in the controller

11 CPU and I/O Device Communication

- CPU commands I/O device to perform a certain task. How does it know when the task has finished?
- **Busy Waiting (Polling)**: CPU repeatedly requests the status of the device (are you done?)
- **Interrupt**: Device generates a signal (an interrupt) for the CPU when the command execution finishes
  - every interrupt is associated with an *interrupt number*
  - an **interrupt vector (descriptor) table** stores the memory locations of service routines to run
  - a special instruction (LIDT) is used to tell the CPU where the table itself is stored in memory

What happens on an interrupt?

On an Interrupt

- Few special purpose registers, including the program counter (EIP) and the program status word (EFLAGS), are stored
- CPU is switched to kernel mode
- The interrupt vector table is consulted to determine the location of the code to execute for the interrupt number
- Control is transferred to this location

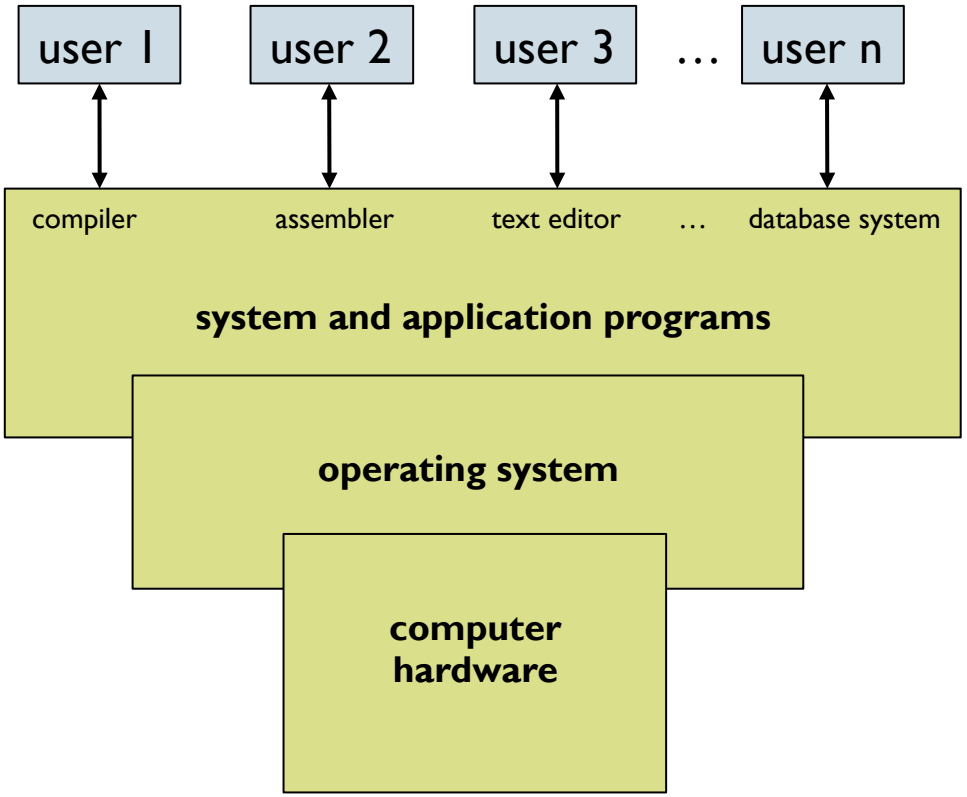
What happens at computer startup?

15 What Happens at Computer Startup?

- Every PC has a **BIOS** (Basic Input/Output System)
  - contains low-level routines to read from the keyboard and the disk, and write to the disk and the display
  - typically stored in ROM or EEPROM
- The CPU executes a special module in the BIOS during power-up or reboot
  - initializes all aspects of the system (Power-On Self Test)
    - CPU registers, device controllers, memory
  - determines the first bootable device (from a stored list)
  - loads the first sector from the boot device (using the routines available in the BIOS) into memory locations **0x7C00 to 0x7DFF** (512 bytes)
  - sets the program counter to 0x7C00 and begins execution

What is an Operating System?

Components of a Computer System



What is an Operating System?

- OS acts as an **intermediary** between application programs and the computer hardware

- A programmer wants to write to the disk
  - at the hardware level: write code to talk directly to the disk controller and send the commands to store the data
  - at the software level: write code to talk to the device driver, and use its interface to send commands to the controller
  - with an operating system: able to wrap the data as a *file*, and use the operating system services to *save a file* to disk

19 Another Operating System Definition

- OS is a **resource manager**
  - which program will run on the CPU?
  - which sections of the memory will be used by a program?
  - which sectors of the disk will be used to store a file?
  - decides between requests for efficient and fair resource use
- One commonly followed definition
  - a program running at all times on the computer, usually called the **kernel**

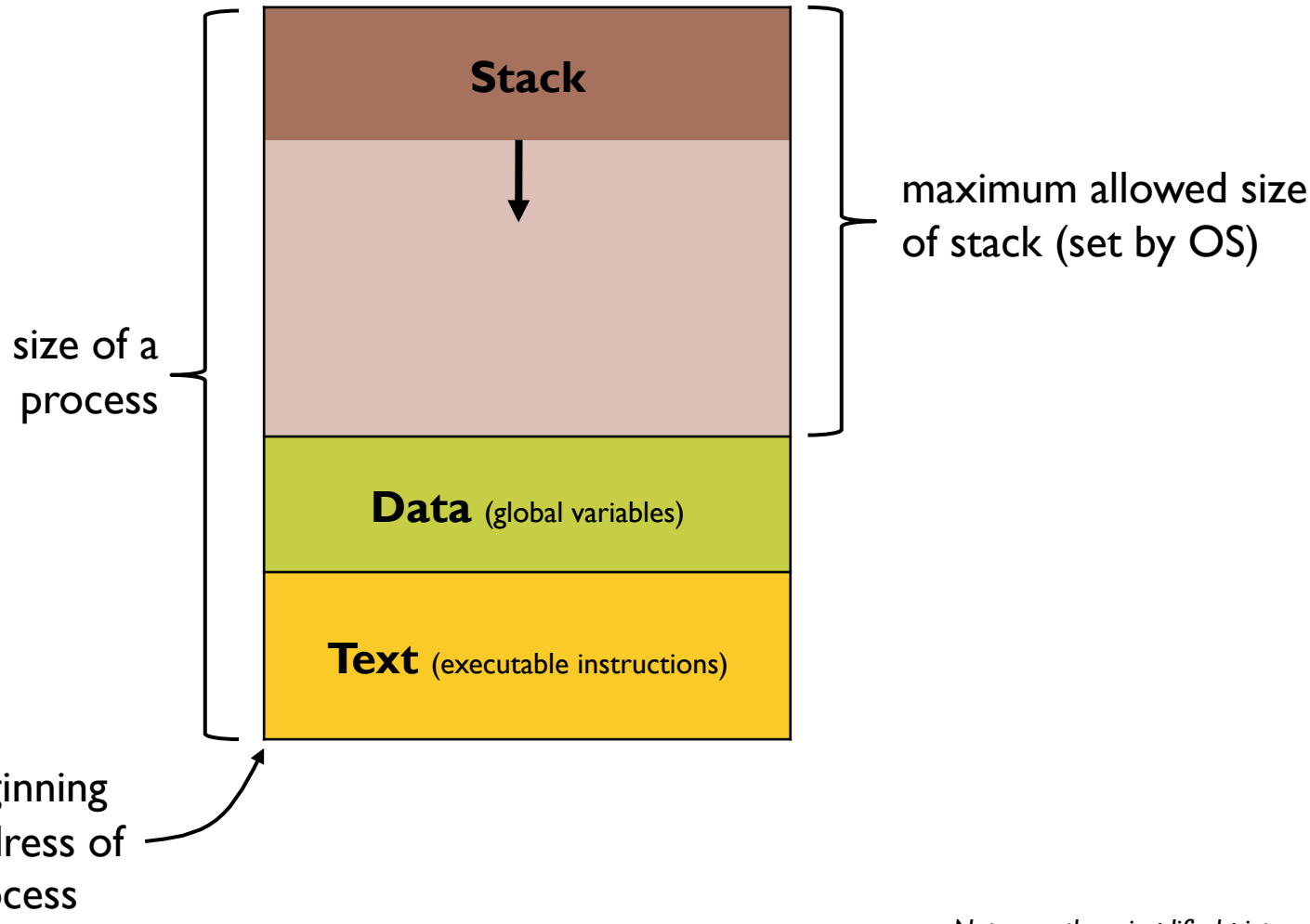
What is a process?

Processes

- A process is a program in execution
  - programs are *passive entities*
  - processes are their *active* counterparts
- Processes need resources, and have states
  - what is there in the CPU registers?
  - which memory locations belong to it?
  - which files are currently being opened by it?
  - which other processes are linked to it?
  - what is it currently doing (running, waiting, sleeping)?
- The OS maintains all such information about all processes in a data structure called a **process table**

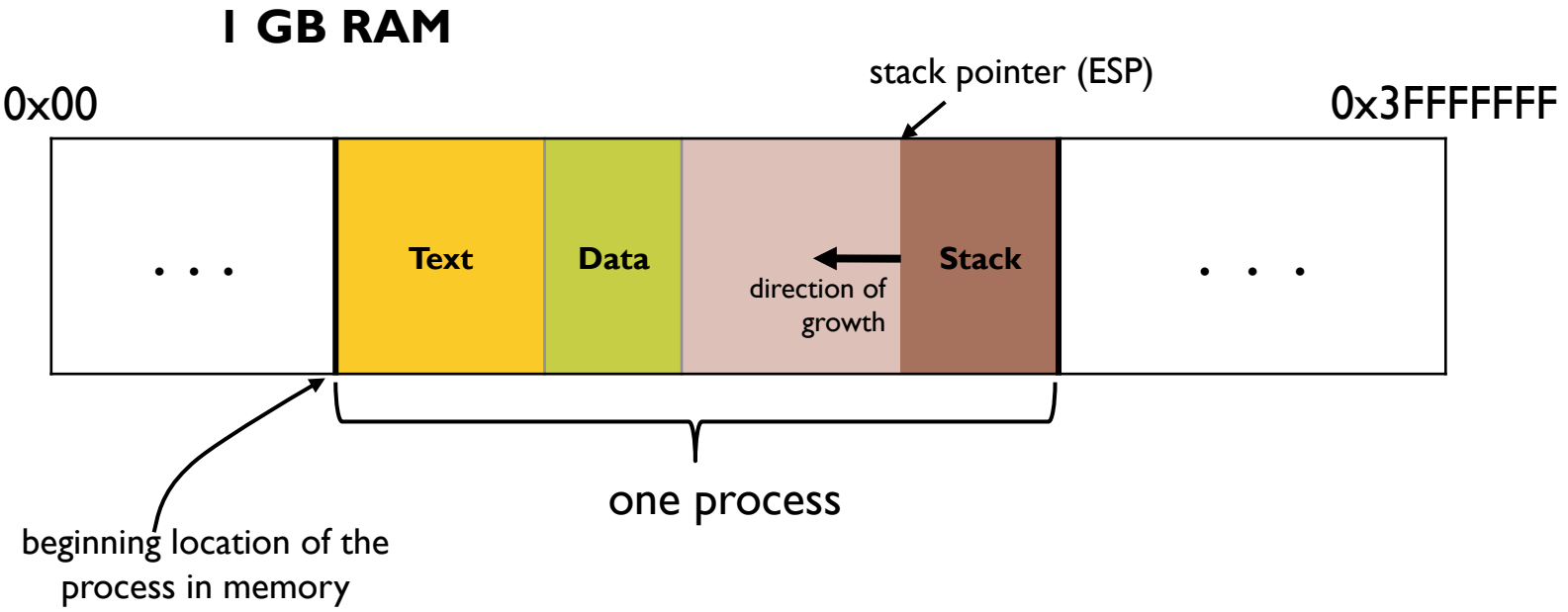
Process Address Space

- How does a process look like in memory?



Note: a rather simplified picture

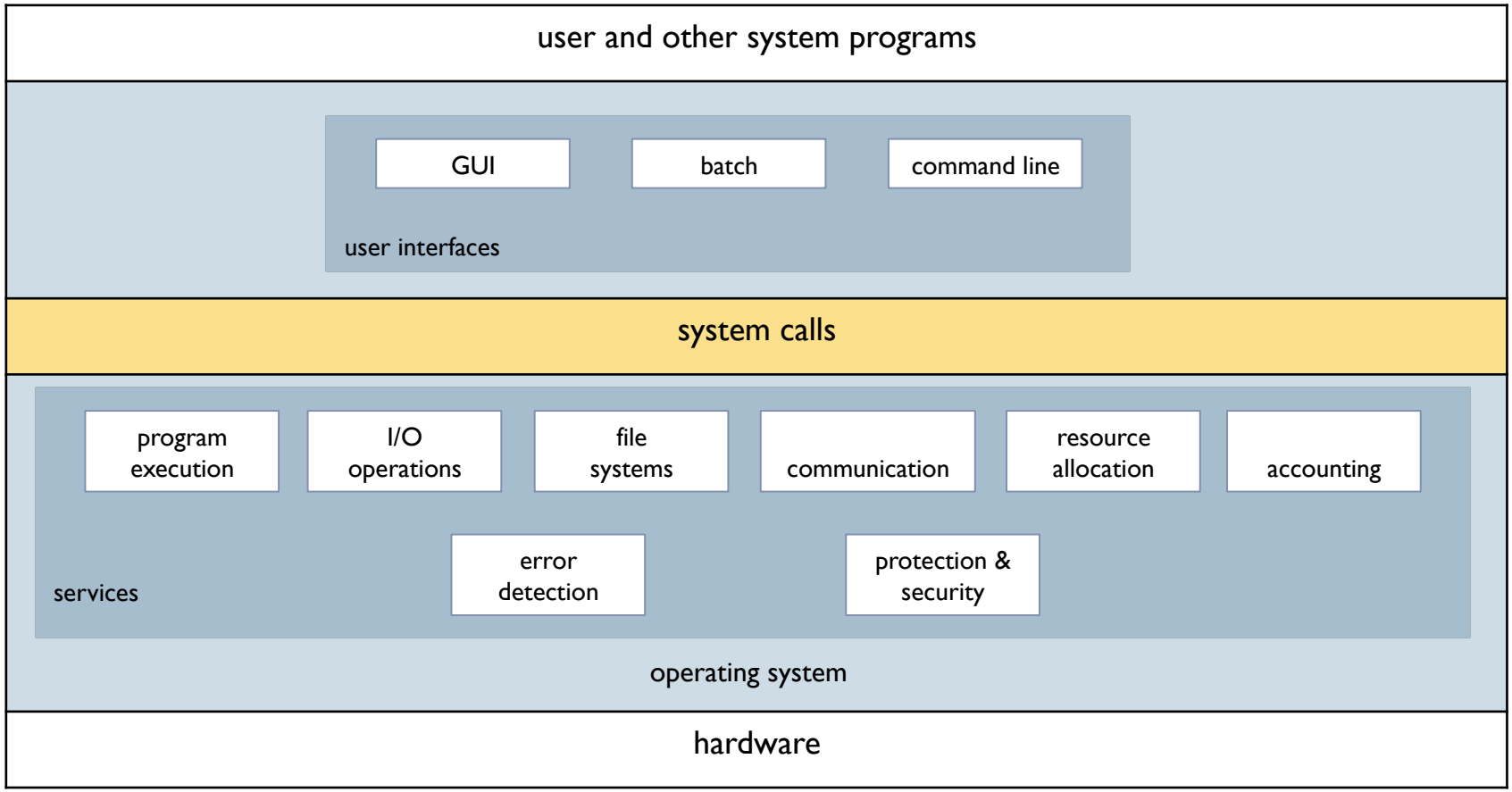
A Process in Memory



- What should the beginning location be?
- Should the beginning address be same for all processes?
- Can parts of the process be in non-contiguous memory locations?
- Can the size of the process be larger than available memory (1 GB in this case)?

What is a System Call?

View of Operating System Services



System Call

- Programming method to obtain the services provided by the OS
- Accessed by programs via high-level wrapper functions, rather than direct system call use
- Switch to OS done by issuing software interrupts
  - e.g. INT instruction
- Most common APIs
  - Win32 API for Windows
  - POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X)
  - Java API for the Java virtual machine (JVM)

Example of System Calls

Example System Call Sequence

Acquire input file name
- Write prompt to screen
- Accept input

Acquire output file name
- Write prompt to screen
- Accept input

Open the input file
- if file does not exist, abort

Create output file
- if file exists, abort

Loop
- Read from input file
- Write to output file

Until read fails

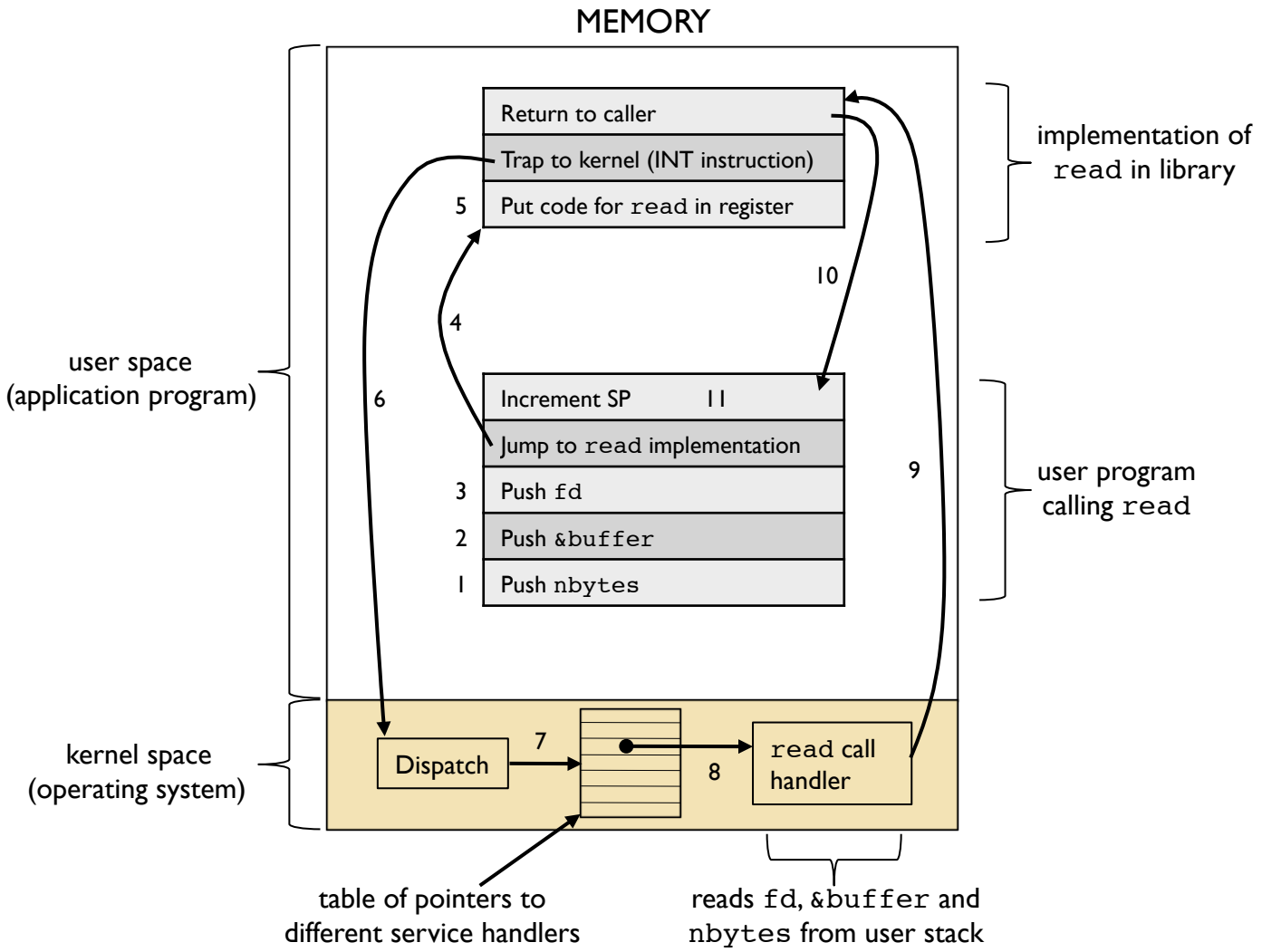
Close output file

Write completion message to screen

Terminate normally

What Happens During a System Call?

```
count=read(fd,buffer,nbytes);
```



System Call Parameter Passing

- At least one parameter: system call number (to identify the requested service)
- More possible depending on the service

- Three general methods used to pass parameters to the OS
  - simplest: pass the parameters in *registers*
    - may have more parameters than registers
  - parameters stored in a *block*, or table, in memory, and address of block passed as a parameter in a register
  - parameters *pushed* onto the *stack* by the program and *read* from the stack by the operating system (system call handler)

How can the different parts of an OS be structured?

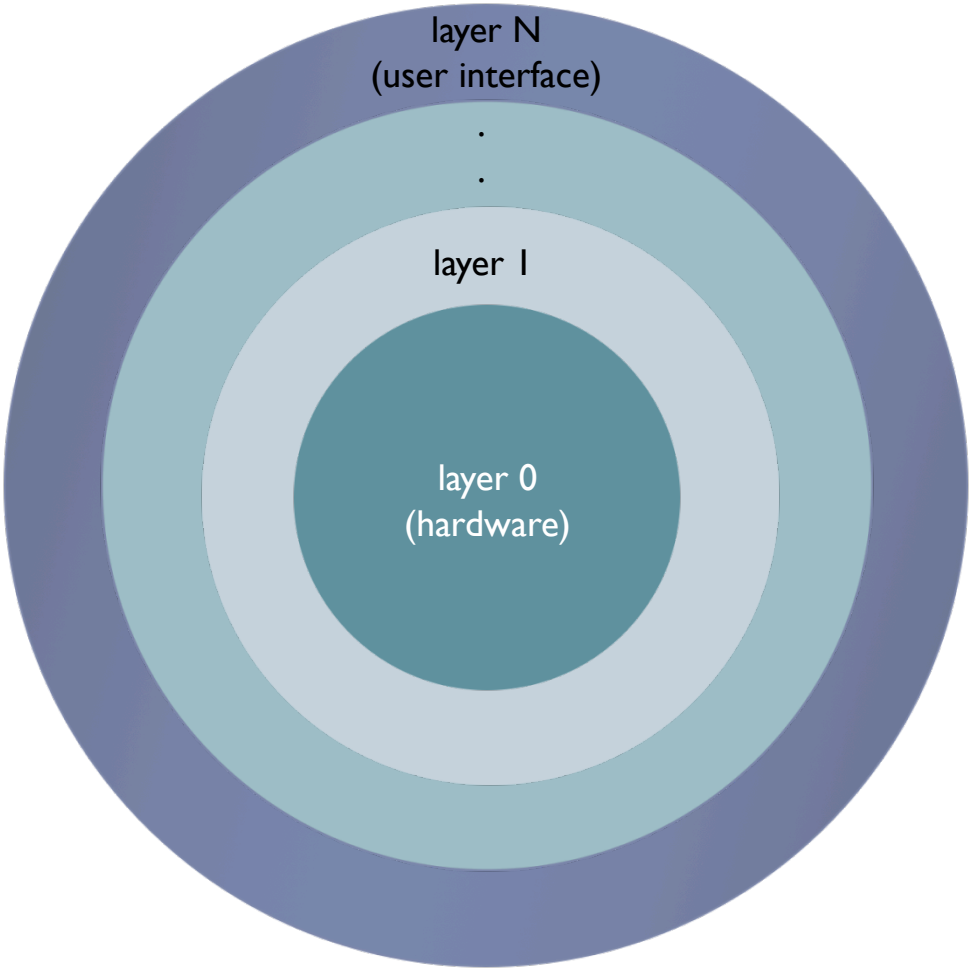
Monolithic Operating Systems

- A single large program that runs in kernel mode
- Basic structure
  - a main program that invokes the requested service procedure
  - a set of service procedures that carry out the system calls
  - a set of utility procedures that help the service procedures
- Loadable extensions (e.g. device drivers) are typically supported in the form of shared libraries or DLLs

OS Layered Structure

- The operating system is divided into a number of layers (levels), each built on top of lower layers
- The bottom layer (layer 0) is the hardware; the highest (layer N) is the user interface
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers

Layered Operating System



OS Microkernel Structure

- Remove all nonessential components from the kernel and implement them as user mode programs
- Communication takes place between user modules using message passing
- Benefits
  - easier to extend a microkernel
  - easier to port the operating system to new architectures
  - more reliable and secure
- Detriments
  - overhead of user space to kernel space communication

References

- Chapter 1, Modern Operating Systems, A. Tanenbaum and H. Bos, 4th Edition.