Class 1 Overview

- course details
- review of computer architecture
- OS as service provider
- The Process

CS571: Where and When

- 'live' classroom presentation in STI:126, Wednesdays 16:30 to 19:10
- live over the Internet using GMU’s NEW software for distance learning: http://netlab.gmu.edu/disted
- recorded as given in class, may be played back anytime

NEW: Network Education Ware

- provides, live, in real-time:
  - overheads of class presentation with
  - instructor annotations and
  - instructor voice, synchronized to overheads
  - live video of classroom
- lets students ask questions:
  - by typing them in
  - by speaking them (if client does audio)
- entire lecture recorded for later Internet playback (excluding live video)
Using NEW

- need client software; runs only on Windows platforms
- minimal machine configuration:
  - P200 or faster
  - IE 4.0 or later, Netscape (6.0 or later)
  - standard sound card
  - ≥ 56 kbps connection to Internet
- see http://netlab.gmu.edu/disted to try it and/or sign up (need password to get courses)

Resources: Systems & Software

- students must have IT&E UNIX system accounts
- students will need ssh clients for remote access
- programming assignments are to be written in C unless otherwise noted

Course Resources

- required text: Applied Operating System Concepts
  Silberschatz, Galvin and Gagné
- course web site: http://cs.gmu.edu/~csnow/cs571/2003A
  - contains printed copies of slides (pdf)
  - other course related info
  - background material

Students must:

- course pre-requisites:
  - CS310 Computer Science III,
  - CS365 Computer Systems Architecture
- be able to program in C

Resources: Instructor + TA

- Instructor: Charles Snow
  - office hours: Wed 14:00 to 15:30
  - ST2, room 435 (across from CS dept office)
  - email: csnow@cs.gmu.edu
- TA: TBA (see course website for update)

CS571 Grading:

- Assigned work: 35%
- Midterm: 25%
- Final exam: 40%
- assignments due by end-of-day on due date
  - assignments handed via http://cs.gmu.edu/course-upload
- grades are proficiency based
CS571 Grading, cont’d:

- important that you demonstrate:
  - understanding of material
  - ability to reason with it
- final exam
  - is comprehensive
  - scheduled for ?? Dec 2003, 16:30 to 19:15
- basic rule: if we cover it in class, it can be on an exam
  - includes reading material suggested but not necessarily explicitly covered in class

What is a computer?

- what’s a computer?
  - has one (or more) processing units
  - has memory where instructions and data reside

Class 1 Overview

- course details
- review of computer architecture
- OS as service provider
- The Process

What is a computer? What’s an OS?

- what’s a computer?
  - has one (or more) processing units
  - has memory where instructions and data reside
  - has one (or more) peripheral units to allow for:
    - interaction with other components
    - persistence of data

What is a computer?

- what’s a computer?
  - has one (or more) processing units
  - has memory where instructions and data reside
  - has one (or more) peripheral units to allow for:
    - interaction with other components
    - persistence of data
What is a computer?

- what’s a computer?
  - has one (or more) processing units
  - has memory where instructions and data reside
  - has one (or more) peripheral units to allow for:
    - interaction with other components
    - persistence of data
  - has necessary electrical and mechanical infrastructure needed to operate its components

A basic computer:

- CPU:
  - executes instructions retrieved from memory,
  - has registers and status word(s),
  - is bus master most of the time,
  - may surrender bus to other devices for brief intervals,
  - can generate traps for failure conditions

- MEMORY:
  - stores bit patterns (instr, data),
  - organized as flat, linear sequence of bytes,
  - is byte addressable (usually),
  - may generate traps for error conditions

- BUS:
  - provides data and control signal connection between units,
  - transfers data between a source and a destination as requested by bus master,
  - can generate traps for certain conditions,
  - “creates” the address space: most addresses are in memory, some are I/O devices (memory mapped I/O)

Memories…

- place to hold a value (data or instruction) while it is worked on or until it is needed to be worked on
Memories...

- place to hold a value (data or instruction) while it is worked on or until it is needed to be worked on
- may require electrical power to hold a value: volatile memory
- may not require electrical power: non-volatile memory
- earliest computer memories, magnetic core, were non-volatile

Memories...

- not all memories are the same:
  - smaller access time $\Rightarrow$ higher cost per bit
  - bigger capacity $\Rightarrow$ smaller cost per bit
  - higher capacity $\Rightarrow$ higher access time

Memories...

- REGISTERS:
  - contained within CPU
  - GPRs: general purpose registers
  - PC: program counter
  - PSW: program status word
  - access at CPU speed
  - volatile

Memories...

- DISKS (magnetic, optical):
  - are peripheral devices
  - interact through interfaces/controllers
  - magnetics: now in 100s of Gb
  - opticals: now in 10s of Gb
  - mechanical access method (e.g., read/write head must be moved to correct position)
  - non-volatile
  - magnetics subject to magnetic fields

Memories...

- CACHE:
  - small amount of high-speed mem between CPU and bus
  - may be multi-staged
  - volatile

Memories...

- MAIN MEMORY:
  - byte addressable
  - flat, linear address space
  - typical sizes: 256 Mb to 2 Gb
  - volatile

Memories...

- REGISTERS:
  - contained within CPU
  - GPRs: general purpose registers
  - PC: program counter
  - PSW: program status word
  - access at CPU speed
  - volatile

Memories...

- DISKS (magnetic, optical):
  - are peripheral devices
  - interact through interfaces/controllers
  - magnetics: now in 100s of Gb
  - opticals: now in 10s of Gb
  - mechanical access method (e.g., read/write head must be moved to correct position)
  - non-volatile
  - magnetics subject to magnetic fields
Memories...

- Memory types:
  - Reg
  - Cache
  - Main memory
  - Magnetic disks
  - Optical disks
  - Magnetic tape

- Magnetic tape:
  - Peripheral devices
  - Interact through interfaces/controllers
  - Now in 100s of Gb
  - Mechanical access method: tape must be moved to correct position
  - Sequential access only
  - Non-volatile
  - Magnetics subject to magnetic fields

- Decreasing size of data-chunk transferred:
  - Lower cost per bit
  - Higher capacity
  - Higher access time
  - Lower frequency of access by CPU

- Small, expensive, fast memory supplemented by larger, cheaper, slower memory

About caches

- Cache hit ratio: \[
\frac{\text{number of accesses content in cache}}{\text{total number of accesses}}
\]

- How high are typically cache hit ratios?
  - 95% or better

- How is this possible?
  - Locality: how close together addresses are that are accessed
    - E.g., if PC points to a word at address k, there is high probability that next PC access is word at address k+4, i.e., next word in memory
    - Operands often show similar local clustering
    - But operands and instructions may not be local w.r.t. each other ⇒ separate caches for I and D
    - Example: loop to zero an array...
About caches

#define SIZE 500
main()
{
    int i,j;
    int fobble[SIZE];
    for (i = 0; i < SIZE; i++) {
        fobble[i] = 0;
    }
}

Input/Output Devices: Polling

- may be polled:
  - e.g., keyboard:
    - while (no new data to read);
    - loop takes ~15 ms (> 65,000 iterations/sec)
    - how fast do you type?
    - what else is CPU doing when it’s doing this?

Our improved computer:

- why do input/output devices have bi-directional connection to bus?
  - data
  - control/status

Input/Output Devices: Using Interrupts

- may be polled
- may be interrupt-driven:
  - when a device needs attention it uses an interrupt to assert the need for service
  - CPU:
    - suspends current processing
    - executes a special routine that performs the service (an interrupt handler)
    - can then resume suspended work
  - are all interrupts equally important?

Input/Output Devices

- may be polled:
  - check device to see if it needs attention
    - yes: do what needs to be done and go back to checking
    - no: go back to checking

- interrupts have different priorities:
  - are processed by priority
  - an interrupt can interrupt an interrupt handler
## Input/Output Devices: Using Interrupts

- Interrupts have different priorities:
  - Are processed by priority
  - An interrupt can interrupt an interrupt handler
- Keyboard example:
  - When data is ready, device asserts interrupt
  - CPU switches from one context to another
  - That of the interrupt handler
  - Switching designed to be fast (e.g., 40 ns)
  - Keyboard handler is short (moves 1 byte: e.g., 10 ns)
  - Return to previous context (e.g., 20 ns)

## Fast Input/Output Devices: DMA

- Requires:
  - “Smart” devices that are capable of interacting directly with bus (becoming bus master)
  - Setup for transfer: device is given instructions
    - E.g., head/sector/cyl, count, read (or write)
    - DMA xfer begins some time later, when device is ready
  - Such transfers are usually small in size
  - I.e., short duration
  - Why?

## Fast Input/Output Devices

- E.g., a hard disk (read)
  - Delivers data at, e.g., 25 Mb/sec
  - If use interrupts, can get a byte every \( \approx 50 \) ns
  - Bottleneck is context switch & CPU intervention every single byte

## Fast Input/Output Devices: DMA

- Requires:
  - “Smart” devices that are capable of interacting directly with bus (becoming bus master)
  - Setup for transfer: device is given instructions
    - E.g., head/sector/cyl, count, read (or write)
    - DMA xfer begins some time later, when device is ready
  - Such transfers are usually small in size
  - I.e., short duration
  - Why?

## What the CPU does:

- CPU repeatedly performs this loop:
  1. Fetch word from memory pointed at by PC
  2. Decode word as instruction
  3. Fetch any operand(s) instruction needs
  4. Execute the instruction
  5. Update status flags as consequence (e.g., NZVC)
  6. Store any result(s)

## What could possibly go wrong?

- CPU repeatedly performs this loop:
  1. Fetch word from memory pointed at by PC
  2. Decode word as instruction
  3. Fetch any operand(s) instruction needs
  4. Execute the instruction
  5. Update status flags as consequence (e.g., NZVC)
  6. Store any result(s)
What could possibly go wrong?

• CPU repeatedly performs this loop:
  1. fetch word from memory pointed at by PC
  2. decode word as instruction
  3. fetch instruction
  4. execute instruction
  5. update status flags as consequence (e.g., NZVC)
  6. store result

⇒ illegal instruction error

What could possibly go wrong?

• CPU repeatedly performs this loop:
  1. fetch word from memory pointed at by PC
  2. decode word as instruction
  3. fetch instruction
  4. execute instruction
  5. update status flags as consequence (e.g., NZVC)
  6. store result

⇒ illegal instruction error

What could possibly go wrong?

• CPU repeatedly performs this loop:
  1. fetch word from memory pointed at by PC
  2. decode word as instruction
  3. fetch instruction
  4. execute instruction
  5. update status flags as consequence (e.g., NZVC)
  6. store result

⇒ illegal instruction error

What could possibly go wrong?

• CPU repeatedly performs this loop:
  1. fetch word from memory pointed at by PC
  2. decode word as instruction
  3. fetch instruction
  4. execute instruction
  5. update status flags as consequence (e.g., NZVC)
  6. store result

⇒ illegal instruction error

Dealing with errors:

• any of these error conditions cause a trap
• upon a trap happening, CPU:
  • suspends current processing
  • executes a special routine that performs the service (a trap handler)
  • could resume suspended work, but usually doesn’t because it usually doesn’t make sense to

Dealing with errors:

• upon a trap happening, CPU:
  • suspends current processing
  • executes a special routine that performs the service (a trap handler)
  • could resume suspended work, but usually doesn’t because it usually doesn’t make sense to

• sounds just like interrupts, except:
  • traps don’t have priorities
  • traps are handled instantly as they occur
What the operating system operates:

- basic hardware configuration
  - memory
  - CPU
  - cache
  - input device
  - output device

Different OS for different settings:

- OS can make resource collection apt for one or another kind of use, e.g.,
  - batch: run programs, often from beginning to end, in their entirety (e.g., VISA month-end billing)
  - typically what “mainframe” computers did
  - interactive: run programs of many different concurrent users, usually logged in on terminals
  - real-time: run program(s) subject to the constraint that some events must occur within certain time, e.g., autopilot
  - generally can do any of these with ‘plain’ hardware: the OS makes the difference

Class 1 Overview

- course details
- review of computer architecture
- OS as service provider
- The Process

OS as supervisor:

- computers provide (in hardware) for different levels of user:
  - not all instructions are available at all levels
- OS uses higher privilege level instructions to manage machine’s resources
- ‘user’ programs usually have least privilege level

What the OS gives us:

- a set of resources created, maintained and controlled by the OS
- presented as a set of services for use

OS as supervisor:

- e.g., when running a user program (lowest privilege level) an interrupt occurs:
  - suspend work on user’s stuff (so that it can be resumed)
  - get address of interrupt handler
  - raise priority level to ‘supervisor’ level to perform handler
  - often handled simply through PSW (holds priority and privilege level)

© Charles M Snow 10
Ask the OS...

- user program (low privilege) cannot, e.g.,
  - directly control an I/O device like a disk
  - so how does a user get disk input to program?

OS: Service Provider

5. Error Detection & Recovery
   - traps described earlier: program errors
   - device errors/failure: disk errors, network down,...

6. Resource Allocation:
   - brokers access to resources to programs
   - some allocated at start-up: cpu, memory
   - some allocated dynamically: memory, disk space

7. Accounting: who did what, when, & for how long

8. Protection: of resources under OS management
   against accidental or intentional attack

Ask the OS...

- user program (low privilege) cannot, e.g.,
  - directly control an I/O device like a disk

- user program asks the OS to do the I/O work on
  the user's behalf: it makes a supervisor call
  to invoke OS services

- what kinds of services does the OS provide?

OS: Service Provider

5. Error Detection & Recovery
   - traps described earlier: program errors
   - device errors/failure: disk errors, network down,...

6. Resource Allocation:
   - brokers access to resources to programs
   - some allocated at start-up: cpu, memory
   - some allocated dynamically: memory, disk space

7. Accounting: who did what, when, & for how long

8. Protection: of resources under OS management
   against accidental or intentional attack

OS: Service Provider

5. Error Detection & Recovery
   - traps described earlier: program errors
   - device errors/failure: disk errors, network down,...

6. Resource Allocation:
   - brokers access to resources to programs
   - some allocated at start-up: cpu, memory
   - some allocated dynamically: memory, disk space

7. Accounting: who did what, when, & for how long

8. Protection: of resources under OS management
   against accidental or intentional attack

Ask the OS...

- user program (low privilege) cannot, e.g.,
  - directly control an I/O device like a disk

- user program asks the OS to do the I/O work on
  the user's behalf: it makes a supervisor call
  to invoke OS services

- what kinds of services does the OS provide?

OS: Service Provider

5. Error Detection & Recovery
   - traps described earlier: program errors
   - device errors/failure: disk errors, network down,...

6. Resource Allocation:
   - brokers access to resources to programs
   - some allocated at start-up: cpu, memory
   - some allocated dynamically: memory, disk space

7. Accounting: who did what, when, & for how long

8. Protection: of resources under OS management
   against accidental or intentional attack

OS: Service Provider

5. Error Detection & Recovery
   - traps described earlier: program errors
   - device errors/failure: disk errors, network down,...

6. Resource Allocation:
   - brokers access to resources to programs
   - some allocated at start-up: cpu, memory
   - some allocated dynamically: memory, disk space

7. Accounting: who did what, when, & for how long

8. Protection: of resources under OS management
   against accidental or intentional attack

OS: Service Provider

5. Error Detection & Recovery
   - traps described earlier: program errors
   - device errors/failure: disk errors, network down,...

6. Resource Allocation:
   - brokers access to resources to programs
   - some allocated at start-up: cpu, memory
   - some allocated dynamically: memory, disk space

7. Accounting: who did what, when, & for how long

8. Protection: of resources under OS management
   against accidental or intentional attack

Ask the OS...

- user program (low privilege) cannot, e.g.,
  - directly control an I/O device like a disk

- user program asks the OS to do the I/O work on
  the user's behalf: it makes a supervisor call
  to invoke OS services

- what kinds of services does the OS provide?

OS: Service Provider

5. Error Detection & Recovery
   - traps described earlier: program errors
   - device errors/failure: disk errors, network down,...

6. Resource Allocation:
   - brokers access to resources to programs
   - some allocated at start-up: cpu, memory
   - some allocated dynamically: memory, disk space

7. Accounting: who did what, when, & for how long

8. Protection: of resources under OS management
   against accidental or intentional attack

OS: Where it Lives

- the OS is a collection of instructions and data items
OS: Where it Lives

- the OS is a collection of instructions and data items that must live in memory
- how much of it has to be present at all times?
  - some basic OS code
  - interrupt and trap handlers (and their vectors)
- the part of the OS always resident in memory is the kernel

The Process

- standard model of user of OS services/resources
- is a program, hence uses:
  - cpu
  - memory
  - possibly, other resources
    - e.g., disk, graphics display, network
- has a status
  - e.g., running, waiting
- because it’s standard, OS has standard way to handle

Class 1 Overview

- course details
- review of computer architecture
- OS as service provider
- The Process

The Process

- each new process gets:
  - a unique identifier (PID: process ID)
  - an amount of memory to work with (apart from its instruction space)
  - a privilege level (can do some things, can’t do others)
  - possibly, some or all of resources requested up-front

The Consumer

- who uses the resources managed by the OS?
- services are intended for the running of programs
- we introduce a model for such a ‘consumer’ of these services: the process