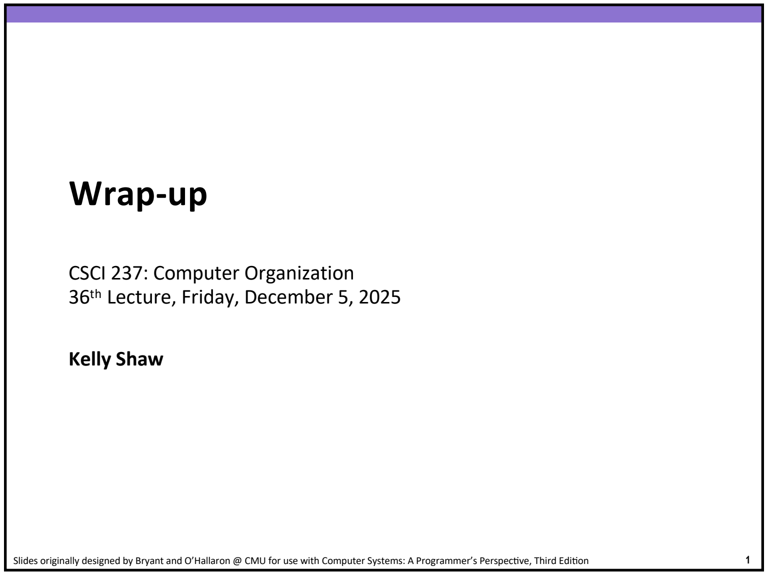
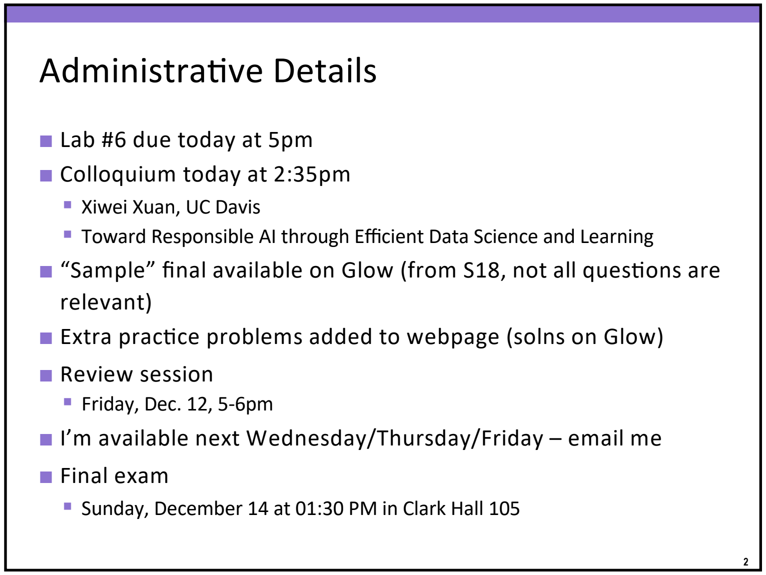


Wrap-up

CSCI 237: Computer Organization
36th Lecture, Friday, December 5, 2025

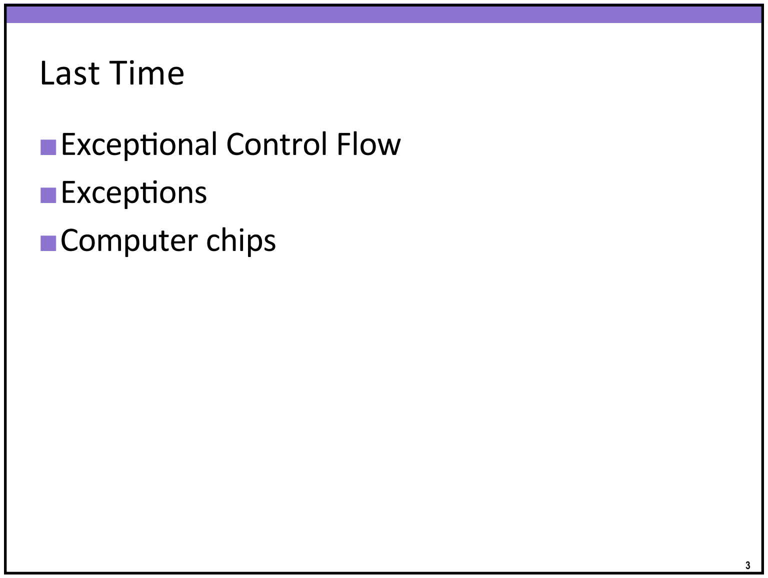
Kelly Shaw

Slides originally designed by Bryant and O'Hallaron @ CMU for use with Computer Systems: A Programmer's Perspective, Third Edition



Administrative Details

- Lab #6 due today at 5pm
- Colloquium today at 2:35pm
  - Xiwei Xuan, UC Davis
  - Toward Responsible AI through Efficient Data Science and Learning
- "Sample" final available on Glow (from S18, not all questions are relevant)
- Extra practice problems added to webpage (solns on Glow)
- Review session
  - Friday, Dec. 12, 5-6pm
- I'm available next Wednesday/Thursday/Friday – email me
- Final exam
  - Sunday, December 14 at 01:30 PM in Clark Hall 105

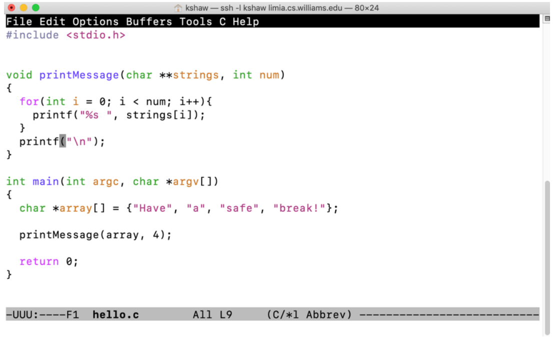


Last Time

- Exceptional Control Flow
- Exceptions
- Computer chips

You've Learned So Much!

Start by Writing a Program in C


kshaw — ssh -l kshaw limia.cs.williams.edu — 80×24
File Edit Options Buffers Tools C Help

```
#include <stdio.h>


void printMessage(char **strings, int num)
{
  for(int i = 0; i < num; i++){
    printf("%s ", strings[i]);
  }
  printf("\n");
}

int main(int argc, char *argv[])
{
  char *array[] = {"Have", "a", "safe", "break!"};

  printMessage(array, 4);

  return 0;
}


-UUU:----F1  hello.c        All L9      (C/*l Abbrev) ----------------------------
```

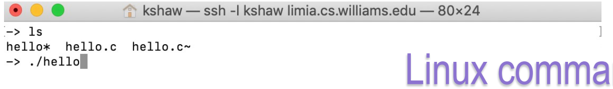
- Learned how to write and debug C code
  - Ah, pointers, we love you! ❤
- Learned how to read and modify someone else's code
- Learned how to use gcc, gdb, make, valgrind, address sanitizer

Compile it to an Executable

gcc! make!

- Learned how to represent data
  - Twos complement
  - Floating point
- Learned how to manipulate data
  - Logical and bitwise manipulation
  - Arithmetic operations (including shifting for ÷ and ×)
- How to convert C code to assembly code
  - Registers/memory
  - `if/else`, `for/while/do-while`, `switch`
  - Function calls
  - Layout of arrays and structs in memory
- How executables are created
  - Text segment holds instructions
    - Encoded in binary (Y86-64)
  - Data segments hold global data

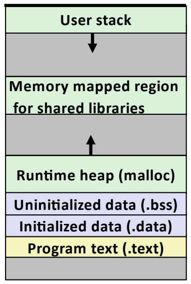
A Process is Created to Run Our Executable



```
-> ls
hello*  hello.c  hello.c~
-> ./hello
```

Linux commands!

- Each process has its instructions and data loaded from the executable into a virtual address space
- Each process thinks it has all of memory to itself
  - Enabled by virtual memory
- Each process thinks it has the CPU to itself
  - Exceptional control flow allows process context switching


User stack

Memory mapped region for shared libraries

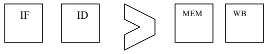
Runtime heap (malloc)

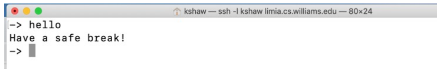
Uninitialized data (.bss)

Initialized data (.data)

Program text (.text)

Process Starts Running (1)

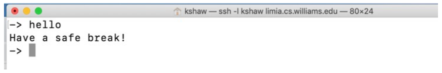




```
-> hello
Have a safe break!
->
```

- PC register is set with address of first instruction
- Register storing page table address is set
- Instructions executed on pipelined datapath
  - Stalling or data forwarding pipeline used depending on processor
  - Branch prediction used to reduce control hazards
  - Every virtual address (instructions and data) translated to physical address and sent to memory hierarchy
    - Access TLB, page table, potentially causes page fault
    - Physical addresses sent to L1/L2/DRAM

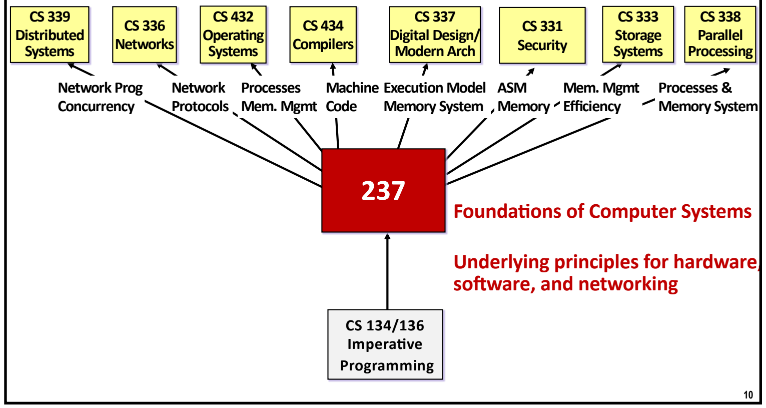
Process Starts Running (2)



```
-> hello
Have a safe break!
->
```

- Operating system invoked on traps/faults/syscalls/interrupts
  - Examples: grow heap, page fault, timer interrupt, divide by 0, read virtual address 0
  - If instruction causes exception, flag is set and carried down pipeline until instruction reaches WB stage
    - Process state (i.e. registers, condition codes) saved into memory
    - Exception table used to find starting address in kernel of exception handler
    - Handler code starts running on processor
- Calls to dynamic memory allocator just normal execution of instructions in library code until system calls to grow heap invoke OS

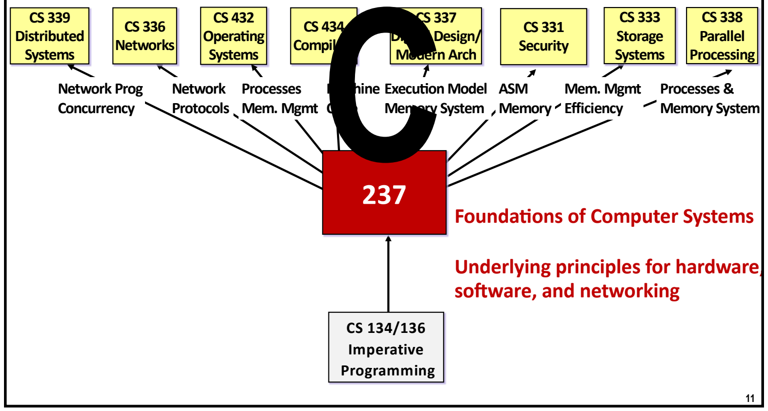
Role within CS Curriculum


CS 339 Distributed Systems
CS 336 Networks
CS 432 Operating Systems
CS 434 Compilers
CS 337 Digital Design/ Modern Arch
CS 331 Security
CS 333 Storage Systems
CS 338 Parallel Processing
Network Prog Concurrency
Network Protocols
Processes Mem. Mgmt
Machine Code
Execution Model Memory System
ASM Memory
Mem. Mgmt Efficiency
Processes & Memory System
237
CS 134/136 Imperative Programming

Foundations of Computer Systems

Underlying principles for hardware, software, and networking

Role within CS Curriculum



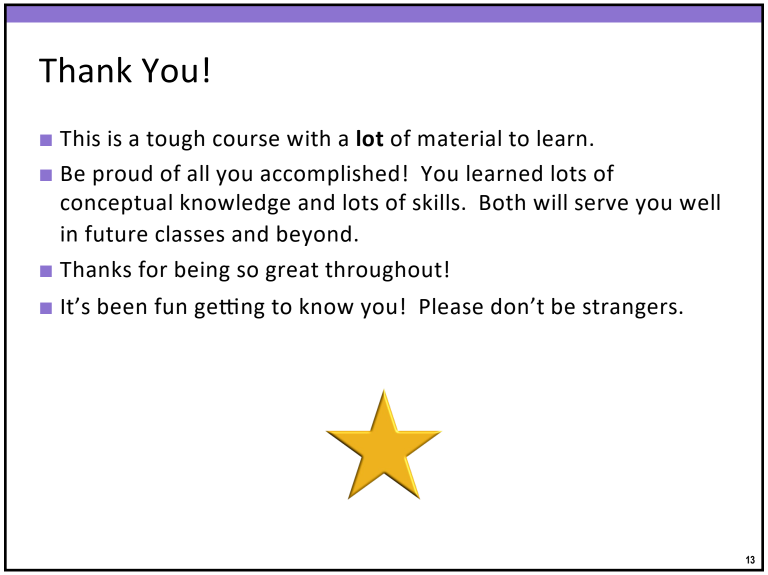
CS 339 Distributed Systems
CS 336 Networks
CS 432 Operating Systems
CS 434 Compil
CS 337 D Design/ M dern Arch
CS 331 Security
CS 333 Storage Systems
CS 338 Parallel Processing
Network Prog Concurrency
Network Protocols
Processes Mem. Mgmt
hine Execution Model Mem y System
ASM Memory
Mem. Mgmt Efficiency
Processes & Memory System
237
CS 134/136 Imperative Programming

Foundations of Computer Systems

Underlying principles for hardware, software, and networking

Final Exam

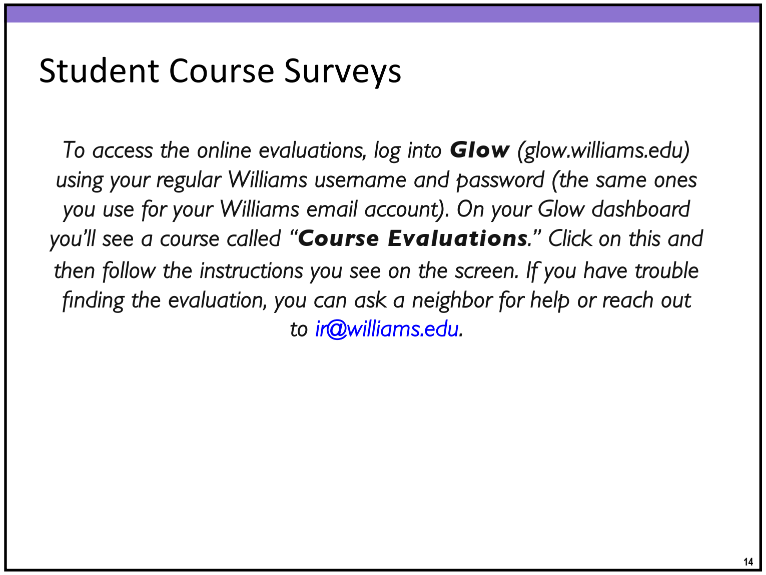
- Closed book, closed notes
- Cumulative with emphasis on material not tested on yet
- Will provide info w/ instructions and registers similar to midterm
- Things to think about
  - Midterm, quizzes, practice problems, in-lecture questions
  - Short answer conceptual questions



Thank You!

- This is a tough course with a **lot** of material to learn.
- Be proud of all you accomplished! You learned lots of conceptual knowledge and lots of skills. Both will serve you well in future classes and beyond.
- Thanks for being so great throughout!
- It's been fun getting to know you! Please don't be strangers.

13



Student Course Surveys

To access the online evaluations, log into ***Glow*** *(glow.williams.edu) using your regular Williams username and password (the same ones you use for your Williams email account). On your Glow dashboard you'll see a course called "****Course Evaluations****." Click on this and then follow the instructions you see on the screen. If you have trouble finding the evaluation, you can ask a neighbor for help or reach out to ir@williams.edu.*

14