**CS 250: Computer Organization and Design**

Dr. Andrew (Drew) Hilton

---

**General Information**

Professor: Andrew Hilton  
Office: Hudson 211  
email: adhilton@ee.duke.edu  
Office Hours: Monday 11am—noon, Thursday 1pm—3pm

Teaching Assistants:  
- Puneet Jain (GTA)  
- Kiron Lebeck  
- Niel Lebeck  
- Akshatha Kommalapati  
- Alexa Murphy  
- Parker Kuivila  
- Kevin Esoda

---

**More Information**

- TA office hours: TBD  
  - Will be held in the Link  
- Recitation Fridays  
  Run by GTA (Puneet) with help from UTAs  
  Learn about C, UNIX, SPIM in hands on fashion  
  Work problems, etc…

  - Bring laptops!

---

**A bit about me**

- Teaching is my primary job  
  - Don’t be afraid to come to my office hours!  
  - Don’t be afraid to ask me to setup some other office hours time!

- Please, feel free to call me “Drew”  
  - If you are uncomfortable with that, Dr. Hilton or Prof. Hilton are ok

---

**A bit about this class**

- Hardware (plus C and MIPS) for CS majors  
  - Why hardware?

---

**A bit about this class**

- Hardware (plus C and MIPS) for CS majors  
  - Why hardware?  
  - Software runs on hardware: write better software  
    - Faster!  
  - Low-level software needs to know hardware details  
    - Operating Systems  
    - Compilers  
  - Might decide you like it and go into comp. architecture  
    - Work at Intel, IBM, AMD, Nvidia,…  
    - Or go into academia
A bit about you

• Before we get too much further, I’d like you all to introduce yourselves
  – I will try really hard to learn everyone’s names quickly…
  – This class is very big, so it may take a while—I will probably learn your name faster if you come to office hours
  – Really quickly, let’s go around the room and give me your name

What we will learn: 10K feet

• Numbers
• C/Unix
• ISA (Hardware/software contract)
• MIPS Assembly
• Logic Design: gates, muxes, adders, flip-flops
• Finite State Machines
• Datapaths
• Memory hierarchy: DRAM, caches, virtual memory
• IO/Exceptions/Interrupts/System calls
• Multicore/pipelining

How we will learn it

• Must “learn by doing”:
  • 2/6 homeworks: Implement something in C
    – Write C, compile it, run it, test it
  • 1/6 homework: Write MIPS assembly
    – Run it in SPIM
  • 2/6 homeworks: Design in LogiSim
    – Tool for drawing digital logic (easier than writing VHDL though)
  • 1/6 homeworks: just pencil and paper (last one)

Homeworks

• Homeworks: 6 of them over semester
  – Work in groups of 2 or 3
  – Fixed groups for semester (except unusual circumstances)
  – Last ~5—10 min of recitation this week/next week to form/meet with groups

Homeworks cont’d

• Homeworks will be turned in and the C/MIPS/Logisim portion demoed
  – Due date: turn electronically in to Sakai
    • Note: tar, tar.gz, and zip only acceptable archives. No Word documents for written portions!
  – Demo: sometime in the next week, setup with a TA

• During demo:
  – Show TA that it works
  – Answer questions about/discuss what you did
  – ALL group members MUST know how it all works
    – TA will ask a particular group member to explain something
    – “I don’t know. Joe worked on that part” will lose you points

Grading

• Grade breakdown
  – Class Attendance/Quizzes (10 of 12) 10%
  – Midterm Exams (2) 36%
  – Final Exam 30%
  – Homeworks (6) 24%

• Late homework policy
  – 5 late days per group total for the semester
    – Does not change demo deadline, only submission deadline
    – Days, not classes
    – Used in entire days at a time: 10 min late = on next day.
    – After used up, no credit for late work.
    – No, really. You can’t ask for more later

• This course takes time, start assignments early.
  – Debugging C and assembly can take time
Homework grading: Non-compiling code

- Non-compiling code = almost no points on that part
  - "But it was really close to compiling" = "it was really far from working"
  - If your code does not compile:
    - You did not test it at all
    - You did not debug it at all
    - It is probably very far from working

Please avoid silly mistakes:
- Turn in assignment
- Download what you turned in
- Unzip in fresh directory
- Compile
- Re-run/re-test it

Class Attendance/Quizzes

- I expect you to attend class...
  - But understand that sometimes things come up and you need to miss
- We will have 12 "quizzes" throughout the semester in lecture
  - You get 50% for being there, 50% for correct answers
  - Drop 2 lowest quiz grades (count 10 out of 12)
  - Questions mostly check if paying attention/understanding discussion
    - Recommend asking questions if you are unclear on things
    - Reading in advance of lecture
    - May try out a few different quiz formats
- Drop quizzes: account for needing to miss lecture
  - Long term circumstances, please talk to me

Academic Integrity

- Academic Integrity Expectations
  - I take academic integrity VERY seriously, and you should too
    - Basic principles for Duke in general:
      - I will not lie, cheat, or steal in my academic endeavors, nor will I accept the actions of those who do.
      - I will conduct myself responsibly and honorably in all my activities as a Duke student.
  - If I suspect academic misconduct in my class...
    - Reported to the Office of Student Conduct
    - Then either
      - Faculty Student Resolution
      - Undergraduate Conduct Board

- Outside resources for general information:
  - Can reference webpages/texts/etc for general information
    - Examples: general C syntax, SPIM reference manual, ...
  - If you do so, cite the resource in your homework (comment in code)
  - Note that you may not download code and re-use it, even if you cite it

Academic Misconduct: The Process

- Everything is fine
- UCB determines sanctions, reports outcome to OSC

Academic Integrity: Homeworks

- Expectations for homeworks
  - You will work in groups of 2 or 3, so collaborate freely within group
  - Should not discuss specifics of homework solutions with other students outside your group
    - May freely discuss general class material, study for exams etc
    - Do not exchange (or look at/share) code with other groups
    - Do not get someone else to help you write it or debug it
    - Exceptions: TAs and myself
  - Outside resources for general information:
    - Can reference webpages/texts/etc for general information
      - Examples: general C syntax, SPIM reference manual, ...
    - If you do so, cite the resource in your homework (comment in code)
    - Note that you may not download code and re-use it, even if you cite it

Academic Integrity: Exams

- Exams in this class are individual effort
  - No outside resources/help except 1 page of notes
    - No textbook
    - No talking to friends/group members
    - No text messages/cell phones/laptops/calculators/smart phones
  - Before I return your exams, I will photocopy and keep a random subset
    - If you request a re-grade, I will compare your solutions to my photocopy. If they have been changed, I will report the incident directly.
  - Related exam policies:
    - Questions? Raise hand, TA or I will come to you (don’t get up)
    - Need restroom? Raise hand, we will let you go one at a time
    - No calculators/smart phones: too easy to use to chat
**Academic Integrity: Mini-Quizzes**

- You may use your book, notes, and/or slide print outs
  - I.e., any hard-copy written material
- You may not discuss with other class mates,
  - Talking/text messages/note passing etc.
- You may not use electronic/interactive resources
  - No internet, no computers, no smartphones, no calculators

**Academic Integrity: General**

- Some general guidelines
  - If you don’t know if something is OK, please ask me.
  - If you think “I don’t want to ask, you will probably say no” that is a good sign its NOT acceptable.
  - If you do something wrong, and regret it, please come forward—I recognize the value and learning benefit of admitting your mistakes. (Note: this does NOT mean there will be no consequences if you come forward).
  - If you are aware of someone else’s misconduct, you should report it to me or another appropriate authority.
    - Within your homework group, this becomes even stronger: if you are aware that one of your group members has committed misconduct on a homework submission for your group, you are complicit in it if you do not report it.

**Course Problems**

- Can’t make midterms / final, other conflicts
  - Tell us early and we will schedule alternate time
- Irresolvable group problems
  - Come see me. Will allow group changes in extreme circumstances
  - Prefer you try to resolve issues first.
- Other problems:
  - Feel free to talk to me, I’m generally understanding and will try to work with you
  - Some problems may extend well beyond my course
    - Talk to your Director of Undergraduate Studies (probably Owen Astrachan)
    - Talk to your Academic Dean

**Lecture**

- You all will get more out of this if you participate than if I just talk for 75 minutes
  - Please ask questions, discuss things you are unclear on, etc..
  - I will ask you all to answer questions
    - Don’t be afraid of this, I’ll ask everyone
    - If you are wrong, its not the end of the world, we’ll stop and make sure you get it
    - Typically, I’ll work my way around the room, so nobody will be surprised that they are next to be called on
    - This class is large, so I may not get everyone in one lecture

**Resources**

- Sakai
  - Turn in assignments
- Piazza
  - Discussions, questions, etc
  - Announcements I make: required reading
  - Other discussions: strongly recommended reading
- Course Web Page

**Textbook, etc.**

  - (Patterson & Hennessy)
    - You are expected to complete the assigned readings
    - Some material on the CD (e.g., Appendix)
    - Note: get revised 4th edition, but not any of the unusual variants (e.g., ARM version)
- We are going to skip around a bit relative to the book
  - Digital logic in appendices
  - Pipelining later, less focus
- Read
  - Start reading Chapter 1 now
Question for you all

• What is the first rule of Computer Science?

Everything is a number? Really?

• Two questions bouncing around in your heads
  - Seriously? My computer has all sorts of things that aren’t numbers
    • Pictures
    • Movies (primary use of computers: videos of cats, right?)
    • Music
    • Text
    • …
    - Why is this so important?

Why is this so important

• Computers can do a handful of things
  - Simple math on numbers: add, subtract, multiply,…
  - Copy numbers around
  - Compare numbers for equality/inequality
    - And decide what to do next based on the comparison
  - Later in this course (~October): how computation works
    - Digital Logic
    - Datapaths
    - Memory
    - And how to make it fast
      - Pipelining, caches, multicore

I still don’t believe everything is a number..

• Computers seem like they operate on non-numbers
  - Pictures
  - Videos
  - Sounds
  - Letters

  • Key: they are represented as numbers
    - E.g., A=1, B=2, C=3, ..., Z=26
    - Actually, A=65, B=66, a=97, b=98,…
    - ASCII encoding covers all characters
      • Letters, numbers, spaces, punctuation, new lines, ….

Representations: Continued

• Words:
  - Sequences of characters
    • Either length + chars (c0,c1,c2,…,cN)
    • Or chars with special terminator: c0,c1,c2,…,cN, ‘\’

  • How about other things:
    - Images = Length, Width and a bunch of colors…
      - So how to represent colors as numbers?
Representations: Continued

• Words:
  – Sequences of characters
    • Either length + chars (c0,c1,c2,...cN)
    • Or chars with special terminator: c0,c1,c2,...cN, ‘0’

• How about other things:
  – Images?
    • Or put a different way...

Images: a bunch of numbers

Images = Length, Width and a bunch of colors...
So how to represent colors as numbers?
Many ways, most common: RGB
  Three numbers: red, green, blue
Note: many image formats (pg, png) are compressed
Basically have numbers required to do math to get colors

Lolcat = many numbers

Images = Length, Width and a bunch of colors...
So how to represent colors as numbers?
Many ways, most common: RGB
  Three numbers: red, green, blue
Note: many image formats (pg, png) are compressed
Basically have numbers required to do math to get colors

Everything is a (finite sized, integer) number

• Math question: How many numbers are there?

• CS question: How many numbers are there?

Everything is a (finite sized, integer) number

• Math question: How many numbers are there?
  – An infinite number

• CS question: How many numbers are there?
  – 256
  – 65,536
  – 4,294,967,296
  – or (roughly) 18,446,744,000,000,000,000
Depending on the type of number you use: how many bits it has

P.S. The answer to most CS questions: “it depends”
…but you have to elaborate on what it depends on, and how...
Numbers for computers

- Computers used fixed-size binary numbers
  - Previous slide: 8, 16, 32, 64 bits (most common sizes)
- Binary = base 2 system
  - We usually use base 10:
    - $12345 = 1 \times 10^4 + 2 \times 10^3 + 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$
  - Recall from third grade: 1's place, 10's place, 100's place...
  - Yes, we are going to re-cover 3rd grade math, but in binary
  - What is the biggest digit that can go in any place?
    - Base 2:
      - 1's place, 2's place, 4's place, 8's place, ...
      - What is the biggest digit that can go in any place?

Binary continued:

- Binary Number Example: 101101
  - Take a second and figure out what number this is
  - 1 in 32's place = 32
  - 0 in 16's place
  - 1 in 8's place = 8
  - 1 in 4's place = 4
  - 0 in 2's place
  - 1 in 1's place = 1
  - Answer = 45

Decimal to Binary

- Suppose I want to write the decimal number 97 in binary
  - Find largest power of 2 smaller than 97: 64
  - We will start in the 64's place, we'll call the current place P.
  - Set a variable X = the number (so X = 97)
  - Repeat the following steps until X is 0
    - See if P is less than X
      - If so, write down a 1 in the P's place, and do X = X – P
      - If not, write down a 0 in the P's place, and leave X unchanged
    - Do P = P / 2
    - X = 97
    - P = 64
    - Answer =
Decimal to Binary

- Suppose I want to write the decimal number 97 in binary
  - Find largest power of 2 smaller than 97: 64
  - We will start in the 64's place, we'll call the current place P.
  - Set a variable X = the number (so X = 97)
  - Repeat the following steps until X is 0
    - See if P is less than X
      - If so, write down a 1 in the P's place, and do X = X – P
      - If not, write down a 0 in the P's place, and leave X unchanged
    - Do P = P / 2
    - X = 1
    - P = 16
    - Answer = 11

- Suppose I want to write the decimal number 97 in binary
  - Find largest power of 2 smaller than 97: 64
  - We will start in the 64's place, we'll call the current place P.
  - Set a variable X = the number (so X = 97)
  - Repeat the following steps until X is 0
    - See if P is less than X
      - If so, write down a 1 in the P's place, and do X = X – P
      - If not, write down a 0 in the P's place, and leave X unchanged
    - Do P = P / 2
    - X = 1
    - P = 8
    - Answer = 110

- Suppose I want to write the decimal number 97 in binary
  - Find largest power of 2 smaller than 97: 64
  - We will start in the 64's place, we'll call the current place P.
  - Set a variable X = the number (so X = 97)
  - Repeat the following steps until X is 0
    - See if P is less than X
      - If so, write down a 1 in the P's place, and do X = X – P
      - If not, write down a 0 in the P's place, and leave X unchanged
    - Do P = P / 2
    - X = 1
    - P = 4
    - Answer = 1100

- Suppose I want to write the decimal number 97 in binary
  - Find largest power of 2 smaller than 97: 64
  - We will start in the 64's place, we'll call the current place P.
  - Set a variable X = the number (so X = 97)
  - Repeat the following steps until you have written the 1's place
    - See if P is less than X
      - If so, write down a 1 in the P's place, and do X = X – P
      - If not, write down a 0 in the P's place, and leave X unchanged
    - Do P = P / 2
    - X = 0
    - P = 2
    - Answer = 11000

Note: 64 + 32 + 1 = 97
Why Binary?

- Binary matches well with digital logic
  - Two values in circuits: high voltage or low voltage
  - 0 = low voltage
  - 1 = high voltage
  - We’ll see how all this works in October

Hexadecimal: Convenient shorthand for Binary

- Binary is unwieldy to write
  - 425,000 decimal = 110011110000101000 binary
  - Generally about 3x as many binary digits as decimal
  - Converting (by hand) takes some work and thought
- Hexadecimal (aka “hex”)—base 16—is convenient:
  - Easy mapping to/from binary
  - Same or fewer digits than decimal
  - 425,000 decimal = 0x67C28
  - Generally write “0x” on front to make clear “this is hex”
  - Digits from 0 to 15, so use A—F for 10—15.

Binary ↔ Hex

- Binary ↔ Hex conversion is straightforward
  - Every 4 binary bits = 1 hex digit
  - If # of bits not a multiple of 4, add implicit 0s on left as needed
  - 0000 ↔ 0
  - 0001 ↔ 1
  - 0010 ↔ 2
  - 0011 ↔ 3
  - 0100 ↔ 4
  - 0101 ↔ 5
  - 0110 ↔ 6
  - 0111 ↔ 7
  - 1000 ↔ 8
  - 1001 ↔ 9
  - 1010 ↔ A
  - 1011 ↔ B
  - 1100 ↔ C
  - 1101 ↔ D
  - 1110 ↔ E
  - 1111 ↔ F

Next time...

- Wrap up there for now
  - Will take questions that need individual answering up front
- Next time:
  - What about negative numbers?
  - ...and decimal numbers (2.75, 3.14159)?
  - How do we add and subtract binary numbers?
  - If these numbers are fixed size, what happens when we try to go beyond that range?
- Start reading Chapter 1