CS 2461
Computer Architecture 1
i.e., Introduction to Computer Systems

http://www.seas.gwu.edu/~bhaqiweb/cs2461
Fall 2018
Instructor: Dr. Bhagi Narahari

What is CS 2461 about?!

◊ Look ‘under the hood’ to see how a computer works
- Explore the interface between hardware and software
- Bottom up approach: from transistors on up to algorithm design

◊ With this knowledge you can
- Write better, more efficient software
- Design better hardware
  ➢ Link between hardware and software
- Appreciate the abstractions that are built on top of these foundations

Course Objectives

◊ To understand the structure and operation of a modern computer system from the ground up.
  • Understand basic hardware concepts and build simple circuits
    ➢ digital circuits - gates, bits, bytes, number representation
  • Understand the Von Neumann architecture/computing model
    ➢ structure and operation, (assembly language)
  • Basic “system” concepts
    ➢ runtime stack, simple I/O devices, Unix OS
◊ How high level languages are implemented on the machine (using the C language)
  • How are C programs translated to assembly and implemented on a machine
  • Proficiency in the C programming language
◊ Understand how software/program performance is linked to program and machine properties

A Computer

The computer is composed of input devices, a central processing unit, a memory unit and output devices.
An Important Idea: what are Computers meant to do ?

- Solve problems that are described in English (or Greek or French or Hindi or Chinese or ...) and use a box filled with electrons and magnetism to accomplish the task.*
  - This is accomplished using a system of well defined (sometimes) transformations that have been developed over the last 50+ years.
  - As a whole the process is complex, examined individually the steps are simple and straightforward
  - Definition from the textbook
  - So how do you get the electrons to run around and do our task ?

Two recurring themes in Computer Sci.

- Abstraction
  - The notion that we can concentrate on one “level” of the big picture at a time, with confidence that we can then connect effectively with the levels above and below.
  - Framing the levels of abstraction appropriately is one of the most important skills in any undertaking.
    - THIS is the whole point of mathematical modeling in Engineering/CS fields
- Hardware vs. Software
  - On the other hand, abstraction does not mean being clueless about the neighboring levels.
  - In particular, hardware and software are inseparably connected, especially at the level we will be studying.

Two pillars of Computing

- Universal Computational Devices
  - Given enough time and memory, all computers are capable of computing exactly the same things (irrespective of speed, size or cost).
    - Turing’s Thesis: every computation can be performed by some “Turing Machine” - a theoretical universal computational device
    - You will see this in the Foundations course in junior year
- Problem Transformation
  - The ultimate objective is to transform a problem expressed in natural language into electrons running around a circuit!
    - That’s what Computer Science and Computer Engineering are all about: a continuum that embraces software & hardware.
    - Note the role of compilers/ translators

Problem Transformation – Making the Electrons Work

- Problems to be solved
  - Send an email from person A to person B
  - CS terminology: How to route a message from network node A to node B ?
- Algorithms to solve the problem
  - Model as a graph and design Shortest path algorithm
- Write the program using some prog. Language (Java, C,...)
  - C program, with required data structures
- Machine (ISA) Architecture used to implement program
  - Intel IA-32(x86)/ Intel processor
- Microarchitecture to implement the machine instructions
  - Intel Core2 Duo ...
- Circuits
- Devices
Problem Transformation: Levels of Abstraction

- These levels do not necessarily correspond to discrete components, but to well defined standard interfaces.

- Standard interfaces provide
  - portability
  - third party software/hardware
  - wider usage

- These levels are to some extent arbitrary - there are other ways to draw the lines.

The Machine/ Hardware Level

- Machine Architecture
  - This is the formal specification of all the functions a particular machine can carry out, known as the Instruction Set Architecture (ISA).
  - We will study the ISA, and Assembly Language programming of a simple computer LC3 – why select a simple “unrealistic” computer?

- Microarchitecture
  - The implementation of the ISA in a specific CPU - i.e. the way in which the specifications of the ISA are actually carried out.
  - We give an overview of the microarchitecture; CS 3462 covers this topic

The Machine Level - 2

- Logic Circuits
  - Each functional component of the microarchitecture is built up of circuits that make “decisions” based on simple rules
  - We will study the basic building blocks of logic circuits
  - You will learn to implement hardware logic circuits in the labs

- Devices
  - Finally, each logic circuit is actually built of electronic devices such as CMOS or NMOS or GaAs (etc.) transistors.
  - Device electronics – not in this course

Approach

- Bottom-up: from bits to C
  - Establish link between hardware and software
  - Learn C in context of hardware
    - C is not a high level language!
    - C is a relatively cross-platform compatible assembly!
    - Used for systems programming
    - Misused by many!
    - Learn what actually happens when your programs run
    - If you understand the hardware the language loses a lot of its mystery!

- Pre-requisites/Corequisites
  - CS1112 is a pre-req (data structures & prog)
  - CS1311 (discrete math) is a pre-req: will need this knowledge in a couple of topics we cover
  - CS 2113 is a co-requisite
    - I will be synchronizing with instructor
Why C?

C is the ultimate high-level, low-level language
- High level enough that you can write large scale programs in It
  - Low level enough that you can reach down and twiddle the bits.

"C is quirky, flawed, and an enormous success." Dennis M. Ritchie, creator of the C programming language.

"A C program is like a fast dance on a newly waxed dance floor by people carrying razors." - Waldi Ravens.

Course Structure

- Part 1 (8 weeks) of the course spent on hardware
  - From transistors to the design of a simple 16 bit processor
    - Implementation of a simple processor ISA
    - Assembly programming
  - Part 2 of the course (6 weeks) spent on C Prog Lang. and translation to Assembly
    - Quick review of C
    - How are C constructs compiled into LC3 assembly language
  - Managing Memory
    - Stack
    - Heap
  - How to make programs run faster

Course Outline – Check my course website for details

- How to represent information (Chap 2)
- The building blocks of computers: logic gates (Chap 3, Online notes)
- The basic computer: the von Neumann model (Chap 4)
- An example: the LC-3 structure and language (Chap 5)
- Programming the machine: assembly language (Chap 6-10)
- Working in a higher-level language: C (Chap 11-19)
  - How do the lower level design issues relate to your C program performance and correctness?
  - Performance, Correctness and Safety of programs (Online notes)
    - Buffer overflow attacks, speeding up program, debugging
Course: Instruction team

- Lab Instructor/TA:
  - Karl Preisner, CS BS+MS Program (BS 2017)
  - John Shepherd, CS BS+MS Program (BS 2018)
  - (grader: TBD)
  - Office hours TBA
  - Labs are in SEH 4th Floor Comp. Lab

- Undergraduate TA (UTA) and Learning Assistants (LA):
  - UTA: Aaron Coplan, CS Senior
  - LA: Henry Jaensch, CS junior
  - Will be helping with some in-class exercises during lecture sessions; Aaron will be teaching one lab section
  - Will be helping with CS Study Hall – (Sundays?) - and review sessions
    - Go there to review/tutoring any CS sophomore course

In-class exercises

- You will learn through in-class exercises most Thursdays, and some Tuesdays and labs
  - Must read the material and come to class
- You need to complete the exercises while working as a group/team
  - Each team is assigned to a table
  - We may ask a team to present solutions to class
- In-class questions – using clickers
  - Counts towards your in-class grade
  - Will provide the clickers next class

If you do not come prepared to class (by reading the materials before class), it is not going to work out too well…

Textbooks/Software

- Intro. to Computing Systems, by Patt and Patel
  - Will use software that comes with book: LC3 assembler and Simulator
    - SW is free – so install on your laptops (works well on Windows)
- Other useful books
  - Unix for Programmers
  - C Programming language
  - LC3 simulator and C to LC3 compiler
    - Installed in 4th Floor SEH Computer Lab
  - C compiler – gcc (need to use cygwin/Linux/Mac or SEASCF)
  - Basic Unix Programming
  - Hardware Basics – you will given a HW kit for your assignments
  - HW Simulators – CedarLogic (free S/W for Windows)
    - Atanua simulates the logic chips used in the lab…so you can design and simulate your circuit before hardwiring it

Course Materials & Resources

- Course webpage – will have links to syllabus, lecture notes, and inclass exercises
  - www.seas.gwu.edu/~bhagiweb/cs2461
  - (or follow link from my homepage)
- Github: for code submission, team SW projects
- Piazza – for discussions
  - Start using this…get your questions answered, see what questions others have, clarify doubts on assignments…..
Requirements and Grading: Read website for details on how grade is computed

✧ 42% Exams
  • Two exams
  • Makeup (Optional) final (total = average of 3 exams)
✧ 20% Homework and Lab assignments
  • Lab assignments may require completion within lab time
    ➢ No late submissions…except a “one time pass” of 36 hours
✧ 8% Quizzes
  • 10 quizzes (in-class, start of class – you miss if you are late!)
  • Start of class – if you come late, you miss the quiz
  • Will drop 2 lowest scores
✧ 30% Projects and Team activities:
  • about 6 projects
    ➢ First three will be team based; last three will be individual
    ➢ You will be asked to demo some of your projects – be prepared to be examined on any random aspect of your solution (code, design, etc.)
  • In-class activities 2%
    ➢ In groups

Course workload: What you've signed up for....

Academic Integrity
✧ You are here to learn – so keep that in mind
✧ Strictly enforced!
  • “no collaboration” means none of any kind
  • No asking friends
  • No searching on web for answers
✧ Violations will lead to at least a zero on the work and a grade lower than final grade
✧ Stay on top of your work – come ask me/TAs questions!
✧ For team projects (and team based lab assignments), you can work within your team but not across teams
✧ PDT: Plagiarism detection software tool
  • I will be running code submissions through software tool
  • Any pair of submissions with more than 25% similarity will be closely examined

Piazza
✧ Online discussion forum
  • The purpose of this:
    ➢ to encourage students to ask well formed questions
    ➢ To encourage students to answer each others questions
      ➢ Most of the time, you do this better than we do!
      ➢ Be very careful not to border on plagiarism!
      ➢ Don't post your HW solution to the world,
  • Signup email has been sent…check your piazza account
  • Do not expect instant response or substitute Piazza for TA office hours!
    ➢ Piazza is not manned 24 hours/7 days a week
    ➢ sometimes answer may take couple days!
    ➢ Mainly a way for students to help one another with common questions/misunderstandings
      ➢ Not a substitute for office hours
  • NO TA can excuse you from anything/or give any extensions
  • Posting on piazza, not the same as telling instructor things
    ➢ E.g.: I’m going to miss the exam! (cannot do this)
Expectations
✧ Come prepared to class
  • High expectations of preparation for class
✧ In-class expectations – don’t want to sit in class then better to leave the room than disturb others
✧ You will need to spend at least 6 hours outside class time each week
  • If you do not, then you will encounter rough seas
✧ This is not a hard course…it is a VERY hard course
  • But very easy to handle if you spend the required time
  • Practice, practice, practice...especially your programming skills
  • As a CS major, I am assuming you spend 6-10 hours each week programming outside class
✧ You will be expected to learn some materials on your own…
  • This is only the beginning...things get more demanding when you get to your junior year....ask the TA team (they have been through this 'journey')!

Your Initial To-do List
1. Get your “reading resources”
2. Read course web pages.
   1. Lecture notes will be posted
   2. Notes on Blackboard have already been posted
3. Read Chapter 1 as background and complete Chapter 2, sections 2.1 to 2.6 for next class
   • We will complete Chapter 2 next class, so read and come for class
   • There are going to be in-class exercises next class!
4. Attend lab sections and get passwords for machines.

Lab Section Logistics
✧ Most lab sessions will cover ‘new’ material not covered in the lecture
  • If you do not attend the labs, you will miss this
✧ Lab section materials are focused on practical skills
  • Ranging from HW circuit design/testing to using Unix/Compiler tools
  • Concepts are covered in the lectures but we want you to learn both skills & concepts!
✧ Lab Kit – ‘rent’ logistics
  • You can rent out a complete lab kit for the semester
  • You need to pay a refundable deposit of $50
    ➢ Bring Cash to the Lab section and pick up the kit
    ➢ Your deposit will be returned when you return the kit
      – You MUST return the kit
      – You can return (and get deposit back) soon after the first project is submitted
  • You will need to use most of the kit in CS3410-Systems Programming

Next....Let’s get started
✧ Computers solve problems....
✧ They work with information/data
✧ First question: how do you represent/store data?