CS 135: Computer Architecture I

Instructor: Prof. Bhagi Narahari
Dept. of Computer Science
Course URL: www.seas.gwu.edu/~bhagiweb/cs135/
TA: Anikhet Pingley

So, what is this course all about?

This course is about:

- What computers consist of
- How computers work
- How they are organized internally
- What is the link between hardware and software
- Low level programming: how programs really work
- How architecture design affects program performance
- How to fix computers
- How to build one myself real cheap
- Which one to buy
- Knowing all about the Pentium or PowerPC

Course Objectives

- To understand the structure and operation of a modern computer system from the ground up.
  - Understand basic hardware concepts
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
- Understand how software/program performance is linked to program and machine properties
- Working in teams and cooperative learning
  - Learn how to function in a team
  - Team responsibilities, and individual responsibilities
Computing Machines

- Ubiquitous (everywhere)
  - General purpose: servers, desktops, laptops, (PDAs?...)
  - Special purpose: cash registers, ATMs, telephone switches...
  - Embedded: cars, printers, cell phones, industrial machinery, medical equipment...

- Distinguishing Characteristics
  - Speed
  - Cost
  - Ease of use, software support & interface
  - Scalability

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

Making the Electrons Work

- Problems to be solved
  - How to route a message from network node A to node B?
- Algorithms to solve the problem
  - 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 Pentium processor
- Microarchitecture to implement the machine instructions
  - Core2 Duo...
- Circuits
- Devices

Problem Transformation - levels of abstraction

- The desired behavior: the application
- The building blocks: electronic devices
  - Natural Language
  - Algorithm
  - Program
  - Machine Architecture
  - Micro-architecture
  - Logic Circuits
  - Devices
**Computer Organization—Layers**

- Application
- Run-time system
- Operating system
- Register-level h/w
- Gate-level h/w
- Circuit-level h/w

**Interfaces between layers**

**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.

- **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

- **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/interpreters

**A Turing Machine**

Also known as a Universal Computational Device: a theoretical device that accepts both input data and instructions as to how to operate on the data

you will study this in CS152-Theory of Comp
## 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 Program Level

- Most computers run a management program called the *operating system (OS)*.
  - You will learn more about OS in CS 154
- Application programs interface to the machine architecture via the OS.

### An example:

```
This lecture  | Data
PowerPoint   | Application Program
Windows XP   | Operating System
```

## The Machine/Hardware Level - 1

### 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 136 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

### 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
Course Outline

- How to represent information (Chap 2)
- The building blocks of computers: logic gates (Chap 3)
- 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 the lower level design issues relate to your C program performance and correctness ?
  - Performance, Correctness and Safety of programs
  - Buffer overflow attacks, speeding up program, debugging

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
    - CS133 is a pre-req (data structures & prog)
    - CS 143 or C is preferably a co-requisite
      - Don’t worry, I will be synchronizing with cs143 instructor

Why Learn Low-Level Programming?

Interfacing with run-time & operating systems more suited to low-level programming.

Debug all programs more effectively.
  - High-level abstractions have limits, especially in presence of bugs.
  - Need to understand underlying implementations.

Improve program efficiency.

Prepare for later systems classes.

Course Administrivia...
Course Information

- Course materials placed at
  - www.seas.gwu.edu/~bhagiweb/cs135/
  - All lecture notes, homeworks, lab work, and announcements
  - Check info/announcements at least once a week – before class.
- Online submissions using blackboard

Course Organization

- Lecture section
  - Twice a week, TR 2:20–3:35pm
- Recitation & Lab
  - TA: Aniket Pingley
    - Office hours: TBA
  - Tompkins 211
  - What?
    - Tutorials
    - Lab/in-class exercises and programming assignments
    - teamwork

Textbooks/Software

- Introduction to Computing Systems 2nd ed, by Patt and Patel [P&P]
  - Will use software that comes with book
    - Available also on machines in Tomp 211 and Tomp411
    - SW if free – so install on your laptops (works well on Windows)
- Other useful books
  - Unix for Programmers
  - C Programming language
  - Computer Systems: A programmer’s perspective, by Bryant and O’Hallaron [B&O]
- LC3 simulator and C to LC3 compiler
  - Installed in Tomp 211, can be downloaded to your PC
- C compiler – gcc (need to use cygwin/Linux/Mac or Hobbes)
- Basic Unix Programming

Requirements and Grading

- Exams 45%
- Homework and Lab assignments 15%
- Quizzes 10%
- Team activities: 30%
  - In-class activities
  - Group assignments
  - Projects 100%
Exams and Quiz

- Two exams during lecture time slots
  - October, November
- One optional final exam
  - Mainly multiple choice
  - to make up exam grade
- Number of quizzes (7 or 8)
  - Given out at start of lecture or lab
  - Will take the best 5 scores

Labs/Recitations

- Wednesday or Thursday in Tomp 211
- Mix of tutorials, recitations, and prog assignments
  - Schedule will be posted
  - Tutorials on basic Unix, using gcc, debuggers
- You must attend your scheduled lab
  - You cannot attend a lab for which you are not scheduled
  - In some assignments, you will need to turn in assigned work during the lab

Homeworks and Lab Assignments

- Average 1 per week
- Types of assignments
  - Simple programming to emphasize low level concepts learnt in class
  - Assembly programming
  - C programming
- Collaboration is NOT allowed on Homework
  - “practice problems” will be posted for each homework and you can collaborate on the practice problems
- Types of homeworks
  - Paper design of logic circuits
  - Theoretical questions without programming
  - Simple programming

Team Projects

- Require applying concepts learnt in class
  - Some are like solving puzzles
  - At least one using assembly language
  - At least one performance tuning related project
- Projects are not meant to expose you to “real world” projects
  - wait till junior year
  - They are meant to instill basic skills and apply problem solving concepts
  - But they do expose you to real world problems
“Teaching Methodology?”

SCALEUP Model and working in Teams

- Will use a SCALEUP style teaching model in some of the classes
  - Collaborative work with peers
  - Active problem solving
- Come prepared to class
  - High expectations of preparation for class
- You will be assigned to a group/Team
  - Team assignment may be changed at halfway point
- In-class exercises to promote active learning
  - Lecture session will contain in-class exercises
  - Work in the team
  - In some cases, questions will be posted before class and each team will discuss their analysis during class
- Team assignments and projects

Cooperative Learning and Teamwork

- Teamwork is a course objective
- Working in teams is a must
  - Industry expects this!
- working in teams can be beneficial
  - Pull your weight
  - Ask team members for help at right time and context
  - Learn to communicate
**Team Member Roles**

- **Manager**
  - directs the sequence of steps in the problem
  - manages time
  - ensures each group member participates
  - keeps team focused on task
  - summarizes solution and team’s discussion

- **Recorder**
  - writes down actual steps (for example, types code)
  - checks for understanding of all group members
  - makes sure all team members agree on solution
  - submits reports for the team.

- **Discusser/Skeptic**
  - makes sure all possible problem-solving strategies explored
  - suggests alternative approaches or concerns
  - provides reasoning and explanations of steps to team members if necessary.
  - ensures problem and data interpretation is correct.
  - debugs the program, thinks of alternative ways to code or express the code.

**Workload Distribution**

- The team roles will be rotated through the exercises, assignments and projects.
- Only the recorder will submit the final answer.
- Doing everything and not letting other team members do any work constitutes violation of team rules.
- Leadership is not the same as dictatorship!

---

**Team/Group Contracts**

- Each team member will read through the team contract and sign the contract.
  - Details of team roles and class expectations are also provided in the handout.
- Meet with your team members during first week and fix time for weekly team meetings.
  - Attend first week Lab sessions on Sept.1,2 as a default meeting time for first week of team.
- Signed contracts must be submitted by September 7th start of lecture.
  - In-class team activities will start on Sept.7th
- Problems with team members must be brought to instructor’s attention ASAP.

---

**Class participation and in-class exercises**

- Each team will be assigned to a table.
- Usually only one person in each team is allowed to submit the team’s answer (report, code, etc.)
  - This is the only person who should be using the workstation.
- No ‘surfing'/email/chat during lecture
  - I will randomly ask a team to present!
Collaboration

- Collaboration doesn't mean copying.
  - Collaboration is working together so that everyone learns the material.
- No collaboration on homework & in-lab assignments
  - You can collaborate on other questions from the book.
- Team exercises and projects will be assigned to teams
  - Cannot collaborate between teams
- Even though it looks like the homework doesn’t count for many points nothing could be further from the truth!
  - You cannot and will not do well on tests if you do not have a good understanding of the homeworks.

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

Questions?

Please ask at any time!