ECE291

Lecture 0

All about ECE291

Staff info – Me!

- Josh Potts
- Office phone: 244-8032
- Office address: 242 Everitt
- Email: jpotts@uiuc.edu
- Office hours: whenever you see me and I don't blow you off
- Sense of humor: present but very bad

Staff info – the TA's

- Michael Urman (urman@uiuc.edu)
- Ajay Ladsaria (ladsaria@uiuc.edu)
- Ryan Chmiel (rchmiel@uiuc.edu)
- Justin Quek (quek@uiuc.edu)
- David Du (daviddu@uiuc.edu)
- George Lu (georgelu@uiuc.edu)

Lecture outline

- Course overview
- Lab etiquette
- Design flow
- Number systems review
- Assignments

Course overview

What is ECE291?

- Computer Engineering II
- Assembly language programming
- Emphasis on hardware and device programming
- The glue between your logic design classes and your high-level programming classes
- A *lot* of work
- The best class at UIUC

Course goals

- You will become proficient in assembly language programming
- You will learn to organize large programs
- You will learn how to interface with hardware devices in your computer
- You will develop excellent time management skills or you will die

Structure and evaluation

6 Home work sets	125 points
5 Machine problems	300 points
2 Exams	400 points
1 Final project	175 points
Total	**1000 points**

Structure and evaluation 2

- Grading scale
  - Minimum A = 95% x AVG(Top 10%)
  - Minimum B = Minimum A – 100
  - Minimum C = Minimum B – 100
  - Etc.
- If Minimum A as calculated above is > 900 then 900 is the minimum A

Structure and evaluation 3

- The top “few” people will get A+’s unless everyone in the class really sucked
- + and - grades will be dolled out just below letter grade break points.
- So if the minimum A is 900, 890 to 899 will get an A-, 880 to 889 will get a B+.
- I have no quotas on grades. I hope you all get A’s!

Miscellaneous

- The 291 website is your most important source of information

  www.ece.uiuc.edu/ece291

- Home works are completed online at the website
- All lecture notes, schedule changes, handouts, etc, are available on the website.

Miscellaneous 2

- The lab notes may be purchased in the IEEE bookstore in the basement for aproximately $11.
- They have been extensively revised and updated over the summer and should be very useful.
- You may obtain PDF or HTML versions from the 291 Web Site.

Miscellaneous 3

- The 291 newsgroup is used heavily throughout the semester to answer questions

  uiuc.classes.ece291
- Don't post code to the newsgroup
- Post questions and answers to questions, but don't tell people how to code their MP's.
- Be civil. No flaming. Flaming will be punished.
- We also have an announcements newsgroup that only TA's and myself can post to

  uiuc.classes.ece291.announce

Miscellaneous 4

- Review the following documents available on the 291 website
  - Syllabus
  - Class outline and lecture schedule
  - Lab staffing schedule
  - Homework 0
  - Machine problem 0

Lab etiquette

- The 291 lab is in 238 Everitt Lab
- The lab is open 24-7
- Get your keycards activated in 153 EL
- No food or drink allowed in the lab
- Don’t prop the door open
- Don’t let people in the lab who aren’t enrolled in the class

Lab etiquette 2

- The lab has recently been completely renovated.
  - New paint job
  - New tables and chairs
  - Increased number of computers from 30 to 37
  - New computers (1GHz, 256MB ram, 17” flat panel monitors)
  - About $85,000 worth of improvements in all

Lab etiquette 3

- Clean up after yourselves
  - Push your chair back under the table when you leave
  - Don’t leave candy wrappers or pop cans
  - Don’t leave papers laying about
  - Don’t lock screens – log off when you leave
  - **Don’t poke monitors!!!**

Lab etiquette 4

- Computer accounts – reset password at

  http://accounts.ad.uiuc.edu
- Home directories – store your work on the w: drive. The w: drive is your own private work area. It is stored on a server and follows you from computer to computer
- DO NOT STORE ANYTHING ON THE C: DRIVES OF THE LAB COMPUTERS

Lab etiquette 5

- FTP access to home directories

  ftp to elalpha.ece.uiuc.edu

- The 291 lab printer is for printing 291 lab materials.
  - Don’t abuse it
  - 100 page per month quota
  - $0.07 charge per page over quota

Lab etiquette 6

- The lab computers don't belong to you!
  - Don't destroy them
  - Don't install **anything** on them (this includes ICQ, Netscape, Real Player, MIRC, etc.)
  - Don't attempt to move the computers, rearrange the monitors and speakers, etc.
  - ALWAYS log off when you are done

Machine problems

- MP0 – Introduction to assembly language
  - General design flow
  - Debugging process
  - Introduction to the software tools
- MP1 – Basic input and output
  - Procedures
  - Basic instructions
  - Library routines

Machine problems

- MP2 – Interrupts and I/O
  - Advanced keyboard input
  - Serial port communication (probably between two computers in a client-server paradigm)
  - Keyboard interrupts
- MP3 – Text mode video, timing
- MP4 – Protected mode, hi-res graphics, image processing, basic-shape drawing

Final project

- All of the above topics plus sound and networking if you want to try them.
- Groups of 2 to 5 people (we prefer 3 or 4)
- Should be the equivalent of a later MP's work per person
- You propose projects to us, we approve them or make suggestions

Why am I doing this?

- Program speed
- Compact executable
- Common in DSP and embedded systems
- Games, games, games!!!

Design flow

- Coding
  - We will provide you with “skeleton” files for each MP. Copy this skeleton file to your w: drive and work on it from there. For MP0, the file is mp0.asm.
  - Write your assembly code in mp0.asm using your favorite text editor (we suggest vim)
  - Run the make command. This invokes NASM, the Netwide ASseMbler.

Design flow 2

- Making your executable
  - Running make from the command line assembles your code using NASM, links it with any library functions we provide and generates an executable like mp0.exe
  - If there are syntax errors in your code, Nasm will tell you about it

Design flow 3

- Debugging
  - Use feedback from Nasm to eliminate all syntax errors.
  - Use Turbo Debugger (td) to track down run-time and logic errors.
- MP0 takes you through the above process

Number systems review

- Base 10 representation (decimal) (0..9):
  - $d_n10^n + d_{n-1}10^{n-1} + \ldots + d_210^2 + d_110^1 + d_010^0$
  - Example: 3045

    $= 3*10^3 + 4*10^1 + 5*10^0$

    $= 3000 + 40 + 5$

    $= 3045$

Number systems review 2

- Base 2 representation (binary) (0..1):
  - $d_n 2^n + d_{n-1} 2^{n-1} + \ldots + d_2 2^2 + d_1 2^1 + d_0 2^0$
  - Eg: 101101

    $= 1*2^5 + 1*2^3 + 1*2^2 + 1*2^0$

    $= 32 + 8 + 4 + 1$

    $= 45$

Number systems review 3

- Base 16 representation (hex) (0..9,A..F):
  - $d_n16^n + d_{n-1}16^{n-1} + \ldots + d_216^2 + d_116^1 + d_016^0$
  - 3AF

    $= 3*16^2 + 10*16^1 + 15*16^0$

    $= 3*256 + 10*16 + 15$

    $= 943$

Base conversion

- Division/remainder method - Long division by largest power of base.
  - Example: Convert 45 decimal to binary

    45 divides by 32 (2^5) once, leaves 13

    13 divides by 8 (2^3) once, leaves 5

    5 divides by 4 (2^2) once, leaves 1

    1 divides by 1 (2^0) once, leaves nothing [done]

    Thus: 45 (base 10) == 101101 (base 2)

Number representation

- In computers, the size of a number becomes important. An 8-bit number for example always has 8 binary digits.
  - Example: 0001 1010
- Hexadecimal is a convenient shorthand for binary numbers:
  - Example: 1Ah

Number representation

- In an 8-bit system, the largest number we can represent is 1111 1111, or FFh, or 255d
- If we need to represent bigger numbers, we need more bytes.
- 16-bit numbers go up to 65,535d or FFFFh or 1111 1111 1111 1111.

Signed numbers

- Need a way to represent negative numbers
- Let's try using the first bit or most significant bit as a sign bit!
  - Sign bit = 0: positive
  - Sign bit = 1: negative

    Example: **1**000 0010 = -2 decimal

    0000 0010 = +2 decimal

Signed numbers 2

- Trouble/Confusion/Problem
  - How is 0000 0000 different from 1000 0000?
  - One is -0 and one is +0. No different at all!
  - We should intuitively dislike having to different representations for the same thing.
- More problems
  - It becomes difficult to process positive and negative numbers. Special cases for each.

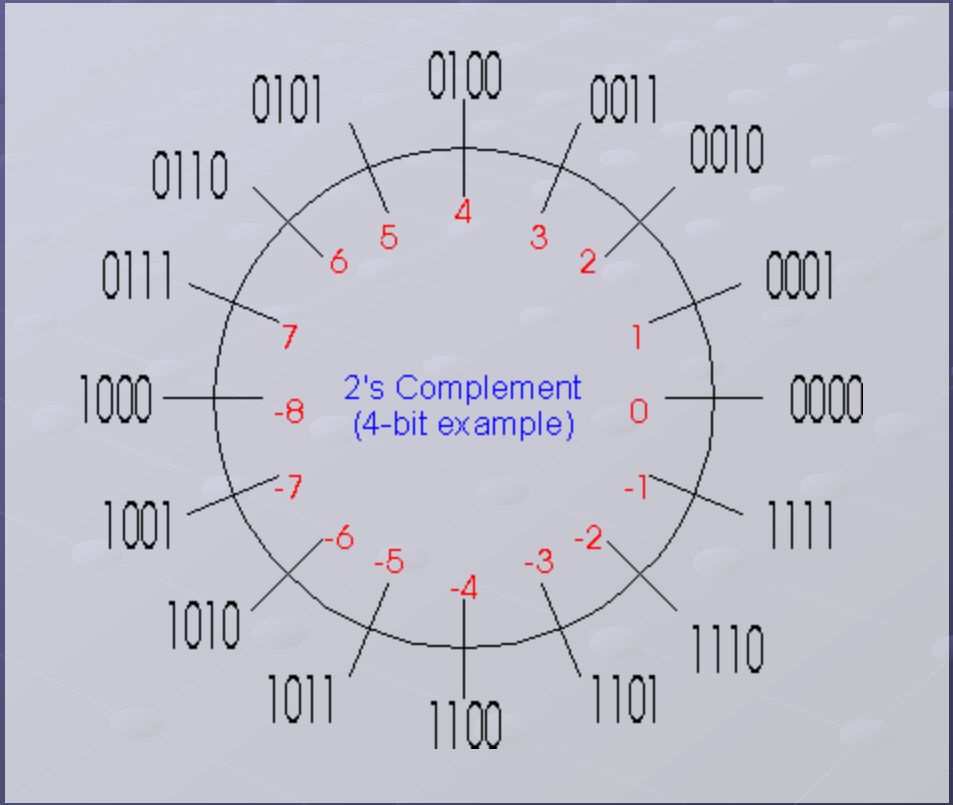
Two's complement

- The solution is to use Two's Complement
  - Numeric Formula:

    $-d_n 2^n + d_{n-1} 2^{n-1} + \ldots + d_2 2^2 + d_1 * 2^1 + d_0 * 2^0$

  - *Notice the negative sign in front of d_n*

Two's complement 2



Two's complement 3

- We are taking advantage of the circular nature of fixed-length binary numbers
- To subtract A-B, perform A+(-B).
  Now the addition operator works for negative numbers
- Notice: First bit *still represents* the sign of the number.
  - s=0: positive (+)
  - s=1: negative (-)

Two's complement 4

- Three methods to compute a negative number (n) (choose one method)
  - Invert bits of *n* then add 1
  - Take largest number (all ones), subtract *n*, add 1
  - Scan *n* from right to left. Copy zeros, copy first one, invert remaining numbers

Two's complement 5

- Examples (8-bit):
  - -1 = 1111 1111
  - -2 = 1111 1110
  - -128 = 1000 0000
  - +1 = 0000 0001
  - +127 = 0111 1111
  - +128 = Invalid

Two's complement 6

- Examples: (16-bit)
  - -1 = 1111 1111 1111 1111
  - -2 = 1111 1111 1111 1110
  - -32,768 = 1000 0000 0000 0000
- Observations
  - When you move to a larger register (increase the number of bits), just extend the sign bit to the left.

Sign and Zero Extension

- Consider the value 64
  - the 8-bit representation is 40h 0100 0000
  - the 16-bit equivalent is 0040h 0000 0000 0100 0000
- Consider the value -64
  - the 8-bit two's complement is C0h 1100 0000
  - the 16-bit equivalent is FFC0h 1111 1111 1100 0000
- The rule: to sign extend a value from some number of bits to a greater number of bits - copy the sign bit into all the additional bits in the new format
- Remember - you must not sign extend unsigned values

Sign Contraction

- Given an n-bit number, you cannot always convert it into an m-bit number if m < n
- Consider the value -448
  - the 16-bit representation is FE40h 1111 1110 0100 0000
  - the magnitude of this number is too great to fit into an 8-bit value
  - this is an example of an overflow condition that occurs upon conversion
- To properly sign contract one value to another the bits that you wish to remove must all contain either 0 or 1
  - FF80h *can be signed contracted to* 80h
  - 0100h cannot be sign contracted to 8-bit representation

Real numbers in binary

- Recall
  - $d_n*2^n + d_{n-1}*2^{n-1} + ... + d_2*2^2 + d_1*2^1 + d_0*2^0$
- Notice the exponents begin from n and decrease to zero.
- For fractional parts, just keep going
  - $d_{-1}*2^{-1} + d_{-2}*2^{-2} + d_{-3}*2^{-3} + \ldots$

Real numbers in binary

- Example: Convert 11010.101 to decimal
  - $2^4 + 2^3 + 2^1 + 2^{-1} + 2^{-3}$
  - 16 + 8 + 2 + .5 + .125
  - 26.625
- For negative numbers, we just stick a sign bit in the front and forget about it.

Assignments

- Get started on MP0.
- Start HW0. Due Wednesday
- Reset your computer account password
- Get your keycard activated
- Go buy the Lab Manual