CS 341
Computer Architecture and Organization

Lecturer: Bob Wilson
Office: S-3-176
Phone: 617-287-6475
Email: bobw@cs.umb.edu
Welcome to CS341

• This course teaches computer architecture and organization from the embedded system POV

• Textbooks are:
  – Professional Assembly Language
  – Shaum’s Outline Series, Digital Principles

• The course syllabus is on my CS341 website: www.cs.umb.edu/~bobw/CS341/index.html

• Go over syllabus
Getting Started

• UNIX Account
  – Apply for CS341 ASAP! (Science 3rd Floor UNIX Lab)

• Remote Internet access from your own PC
  – Connect to and login onto “ulab.cs.umb.edu” directly
  – MUST use Secure Shell 2 for remote access!

• Edit your .cshrc file to add “module ulab”
  – Defines environment variables
  – Makes it possible to compile and run programs, fetch examples, get necessary files

• Don’t use chown, chgrp, or chmod on any of your mp directories or files! That could enable cheating.
Homework

• Familiarize yourself with the course syllabus
• Locate and start reading course references:
  – Web Based / Linked from syllabus web page
• First reading assignment:
  – Professional Assembly Language Programming, pp 1-16
  – Access/Print and read (using links from my web page):
    • User guide to UNIX Systems, Patrick O’Neil
    • The Microprocessor Lab
    • $pcex/pc.handout (using link in the above)
    • $pcex/test.c (using link in the above)
Textbook Commentary

• This textbook is one of the few books that uses the GNU (aka ATT) syntax for i386 assembly language instructions
• We’ll be using the GNU syntax in this course
• Most other books (e.g. the previous textbook) require learning and using a translation for the syntax which was always a pain to deal with
• However, it has a couple disadvantages
Textbook Commentary

• The textbook teaches Intel i386 assembly language programming and development tools from the point of view where code is being run on the same machine as where it is being developed

• The text material on the development tools must be adapted for cross compilation and cross assembly to run on SAPC’s which are simulating embedded systems
The book assumes that you will write your assembly language programs to run on the development host with a Linux O/S.

It shows you how to use Linux system calls for I/O operations that we will not be using.

We will run our assembly language programs on an SAPC under Tutor – not Linux.

We need to use direct I/O instructions (e.g. `in` and `out`) that are not covered in the book.
Embedded Systems

• Embedded processors are inside a system that is not usually thought of as a computer
• There are many more processors in embedded systems than there are in IT applications such as laptops, desktops and servers (90% / 10%)
• There are complex challenges for embedded system programming and these programmers need specialized knowledge and are well-paid
Embedded Systems

• Examples:
  – Automobile engine controls, GPS systems, etc.
  – Appliances such as microwave ovens, TVs, etc.
  – Manufacturing plants / robotic machines
  – Chemical and petroleum plants - control systems
  – Aircraft/Ships/Rockets – commercial and military
  – Building HVAC, elevator, and security systems
  – Communications – routers, switches, and modems
  – Medical instrumentation and implanted devices
Embedded Systems

• Considerations for embedded system software
  – The processors are NOT powerful Pentiums with Gigabytes of memory / disk and Gigabit networks
  – Cost may be critical for high volume products
  – Code/data space and bandwidth are critical resources
  – There may be hard real time constraints so the SW efficiency and timing are usually critical factors
  – The cost of a bug in the SW may include injury or death for a person or even worse many people
Embedded System Example

SCUBA Diving “Computer”
Embedded System Example

SCUBA Diving Computer Block Diagram

- Water Sensor
- Depth Sensor
- Temp Sensor
- Push Button
- Program Memory
- CPU
- LCD Display
- RT Clock
- LOG Memory
- Memory
Embedded System Example

• Diving Computer Software Functionality
  – Automatically turn on when in the water
  – Track/display times and depths during the dive
  – Calculate the amount of Nitrogen absorbed
  – Display the safe remaining dive time
  – Pace the diver’s ascent, e.g. deco/safety stops
  – Record dive data in non-volatile log memory
  – Keep track of surface interval before next dive
  – Keep track of safe time before flying
Embedded System Example

• Team 1: How would you develop and install the software in the diving computer?

• Team 2: How would you test the software in the diving computer?
Embedded System Example

• What are the possible consequences of bugs in the diving computer software?
  – Diver gets the “bends”
  – Diver dies

• A bad example of a corporate cover up:

• YouTube video:
  http://www.youtube.com/watch?v=eNqPTOb31S8&NR=1
Embedded Systems Lab

- Introduction to CS341 lab equipment
- Arduino microcomputer system / breadboard
Embedded Systems Lab

• Microprocessor – AtMega328P
  – 8 bit AVR processor @ up to 20 MHz clock
  – 32 Kbytes program memory plus 1 Kbyte RAM

• Programmed using a language like C/C++ with a custom I/O support library

• Development system cost $36.00!

• Buying the processor alone in quantity would probably be about the price of a candy bar
Embedded System Lab “Big” Picture

- To use embedded software development tools, a development host is attached to a target host via:
  - RS-232
  - Ethernet
  - USB
  - Other?