CS 341
Computer Architecture and Organization

Lecturer: Bob Wilson
Cell Phone: 508-577-9895
Email: robert.wilson@umb.edu
or 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

• Linux Account
  – Apply for CS341 ASAP! (Science 3rd Floor Linux/PC Lab)
  – Upload mp files from your tutor-vserver VM for grading
  – Don’t use chown, chgrp, or chmod on any of your mp directories or files! That could enable cheating.

• Remote Internet access from your own PC
  – Use Secure Shell 2 or putty for secure remote access
  – Login into “users.cs.umb.edu” with Linux ID and password

• Remote Internet access from your tutor-vserver VM
  – Use SCP to download starting mp files
  – Use SCP to upload finished mp files
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)
  – Familiarize yourself with my CS341 site linked references

• Read the instructions on the syllabus for installing the VMWare tutor environment on your own PC
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 the code is being run on the Linux O/S where it is being developed

• The text material on the development tools must be adapted for cross compilation and cross assembly to run under Tutor which is simulating an embedded system
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 embedded system virtual machine 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 (99% / 1%)
• 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 - guidance and control
  – 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
  – Code/data space and bandwidth are critical resources
  – Cost may be critical for high volume products
  – 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
- LCD Display
- RT Clock
- LOG Memory
- CPU
- LOG Memory
- Program Memory
- RT Clock
- LCD Display
- Push Button
- CPU
- Water Sensor
- Depth Sensor
- Temp Sensor
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 (to avoid DCS)
  – 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

• Implement algorithms based on diving tables
Embedded System Example

- Implement algorithms based on diving tables
Embedded System Example

• Diving Computer HW/SW Constraints
  – Small hockey puck sized device (wrist mounted)
  – Waterproof case (no fans for cooling with air)
  – Battery powered (no replacement under water)
  – Software in ROM (no hard drive to boot from)
  – Self-test code so diver can rely on its operation
  – Timing constraints (must provide real time information to the diver – not just a message to head for “the chamber” after the dive is over!)
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” or an air embolism in blood
  – Diver’s buddies get injured/killed trying to rescue
  – Diver dies a painful death

• A bad example of a corporate cover up:

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