CSE 466 – Software for Embedded Systems

- Instructor:
  - Bruce Hemingway
    - CSE 464, Office Hours: 1:30-2:30 p.m., Monday
    - or whenever the door is open
    - bruceh@cs.washington.edu

  - Teaching Assistants:
  - Shuowei Li and Hanchuan Li

CSE 466 – Software for Embedded Systems

- Class Meeting Times and Location:
  - Lectures: EEB 045, MWF 12:30-1:20
  - Labs: CSE 003,   Tuesday, Thursday, 2:30-5:20

- Exams
  - Midterm: Wednesday, October 29, EEB 045, 12:30-1:20
  - Final: Thursday, December 11, 2014, 830-1020, CSE 003

CSE 466 – Software for Embedded Systems

- Grading Policy
  - There will be two exams, as shown on the class schedule.
  - Lab reports: Demo usually required, sometimes questions

- Ratios:
  - Lab: 50%
  - Exams total: 20%
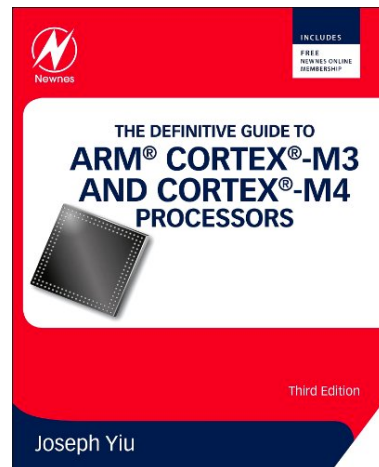 - Homework: 10% (reports)
  - Class Participation: 20%

Recommended Textbook:

The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors, Third Edition

By Joseph Yiu

Newnes; 3 edition (November 1, 2013)

U Bookstore doesn't have it.

Embedded systems

What are Embedded Systems?

■ Anything that uses a microprocessor but isn't a general-purpose computer
– PDAs
– Set-top boxes
– Televisions
– Video Games
– Refrigerators
– Cars
– Planes
– Elevators
– Remote Controls
– Alarm Systems
• The user "sees" a smart (special-purpose) system as opposed to the computer inside the system
   • "how does it do that?"
   • "it has a computer inside it!"
   • "oh! BTW, it does not or cannot run Windows or MacOS!"
  – the end-user typically does not or cannot modify or upgrade the internals

What Are You Going to Learn?

- **Hardware**
  - I/O, memory, busses, devices, control logic, interfacing hardware to software
- **Software**
  - Lots of C and assembly, device drivers, low level real-time issues, scheduling,
  - Concurrency: interrupts
- **Software/Hardware interactions**
  - Where is the best place to put functionality hardware or software?
  - What are the costs:
    - performance,
    - memory requirements (RAM and/or ROM)
- **Integration of hardware and software courses**
  - Programming, logic design, architecture,
  - Algorithms, mathematics and *common sense*

Where Could *You* End Up?

- Automotive systems
  - perhaps designing and developing "drive-by-wire" systems
  - -- self-driving vehicles
- Telecommunications
- Consumer electronics
  - cellular phones, MP3 devices, integrated cellular/tablet/kitchen sink
  - Set-top boxes and HDTV
  - Home appliances
  - Internet appliances
    - your washer will be on the internet more than you are!
- Defense and weapon systems
- Process control
  - gasoline processing, chemical refinement
- Automated manufacturing
  - Supervisory Control and Data Acquisition (SCADA)
- Space applications
  - Satellite communications

Goals of the Course

- **High-Level Goals**

1. Understand the scientific principles and concepts behind embedded systems, and

2. Obtain hands-on experience in programming embedded systems.

By the end of the course, you should be able to

• Understand the "big ideas" in embedded systems

• Obtain direct hands-on experience on both hardware and software elements commonly used in embedded system design.

• Understand the basics of embedded system application concepts such as signal processing and feedback control

• Understand, and be able to discuss and communicate intelligently about

    – embedded processor architecture and programming

    – I/O and device driver interfaces to embedded processors with networks, multimedia cards and disk drives

    – OS primitives for concurrency, timeouts, scheduling, communication and synchronization

---

The Big Ideas

•HW/SW Boundary
• Non processor centric view of architecture
• Bowels of the operating software
       – specifically, basic real-time operation with interrupts
       – Concurrency
• Real-world design
       – performance vs. cost tradeoffs
• Analyzability
       – how do you "know" that your drive-by-wire system will function correctly?
• Application-level techniques
       – signal processing, control theory
       – semaphores, locks, atomic sections

What is an Embedded System?

- Computer purchased as part of some *other* piece of equipment
    - Typically dedicated software (may be user customizable)
    - Often replaces previously electromechanical components
    - Often no "real" keyboard
    - Often limited display or no general purpose display device
- But, every system is unique there are always exceptions
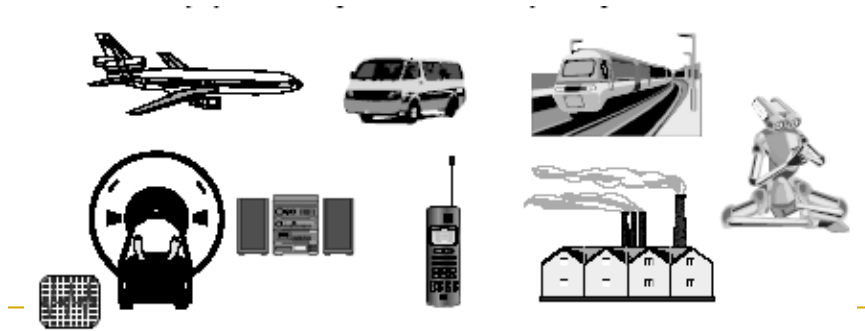
CPU: An All-Too-Common View of Computing

- Measured by:
    - Performance

CPU

An Advanced Computer Engineer's View

- Measured by: Performance
    - Compilers matter too...

| Cache | → | Memory |

CPU
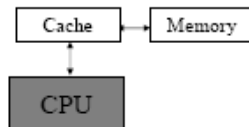
An Enlightened Computer Engineer's View

- Measured by: Performance, Cost
  Compilers & OS matters

| Cache | ↔ | Memory |

CPU ↔ I/O
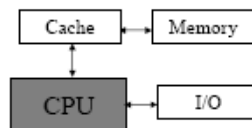
An Embedded Computer Designer's View

- Measured by: Cost, I/O connections, Memory Size, Performance
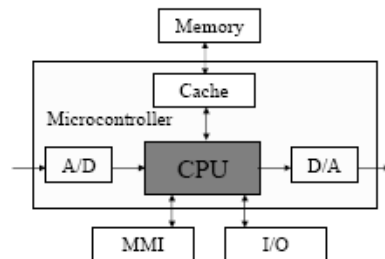
An Embedded Control System Designer's View

- Measured by:
  Cost, Time to market, Cost, Functionality, Cost & Cost.
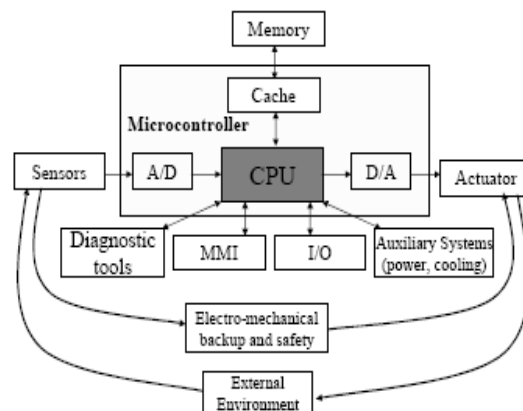
A Customer View

– Reduced Cost
– Increased Functionality
– Improved Performance
– Increased Overall Dependability

Why Are Embedded Systems Different?

Four General Categories of Embedded Systems
• General Computing
    – Applications *similar* to desktop computing, but in an embedded package
    – Video games, settop boxes, wearable computers, automatic tellers
    – Tablets, Phablets
• Control Systems
    – Closed loop feedback control of real time system
    – Vehicle engines, chemical processes, nuclear power, flight control
• Signal Processing
    – Computations involving large data streams
    – Radar, Sonar, video compression
• Communication & Networking
    – Switching and information transmission
    – Telephone system, Internet
    – Wireless everything

Typical Embedded System Constraints

- Small Size, Low Weight
  - Handheld electronics
  - Transportation applications weight costs money
- Low Power
  - Battery power for 8+ hours (laptops often last only 2 hours)
  - Limited cooling may limit power even if AC power available
- Harsh environment
  - Heat, vibration, shock
  - Power fluctuations, RF interference, lightning
  - Water, corrosion, physical abuse
- Safety critical operation
  - Must function correctly
  - Must not function incorrectly
- Extreme cost sensitivity
  - $.05 adds up over 1,000,000 units

Embedded System Design World-View

A complex set of tradeoffs:

- Optimize for **more than just speed**
- Consider **more than just the computer**
- Take into account **more than just initial product design**

Multi-Discipline		**MultiPhase**		**MultiObjective**
• Electronic Hardware		• Requirements		• Dependability
• Software		• Design		• Affordability
• Mechanical Hardware	**X**	• Manufacturing	**X**	• Safety
• Control Algorithms		• Deployment		• Security
• Humans		• Logistics		• Scalability
• Society/Institutions		• Retirement		• Timeliness

Embedded System Designer Skill Set

- Appreciation for multidisciplinary nature of design
  - Both hardware & software skills
  - Understanding of engineering beyond digital logic
  - Ability to take a project from specification through production
- Communication & teamwork skills
  - Work with other disciplines, manufacturing, marketing
  - Work with customers to understand the real problem being solved
  - Make a good presentation; even better write ``trade rag'' articles
- And, by the way, technical skills too...
  - Low-level: Microcontrollers, FPGA/ASIC, assembly language, A/D, D/A
  - High-level: Object oriented Design, C/C++, Real Time Operating Systems
  - Meta-level: Creative solutions to highly constrained problems
  - Likely in the future: **U**nified **M**odeling **L**anguage, embedded networks

Class logistics – see course web

- http://courses.cs.washington.edu/courses/cse466/14au/
- Class structure
- Business matters
- Grading
- Syllabus
- What we'll be doing

11

Class structure

- Lecture
  - Mondays: about the week's lab
  - Closely linked to laboratory assignments
  - Cover main concepts, introduce laboratory problems
- Lab
  - Work leading to implementation of a final project
  - Lab reports due prior with 30 minutes of start of next lab section
- Exams
  - Two, based on lecture, lab, and datasheet reading
  - **Open datasheets, open notes**
- Final demo
  - During last class time – participation required

Business Matters

- Lecture notes will be on line after class (links on Calendar page)

- You pick lab partner assignments

- Sign up for CSE466 mailing list (soon)

Grading

- Lab reports:
  - Demonstration(s) required
  - Brief answers to questions embedded in assignment
  - Sometimes hand-in code
  - Do with your partner
  - Both build hardware
- Distribution:
  - Labs: 50%
  - Exams: 20%
  - Homework: 10%
  - Class Participation: 20%

CSE466 Lab Content

- Arm Cortex M4 processor
  - Begin with basics and build
  - Do with your lab partner; both will build hardware
  - You can work alone at home
- Resources
  - Freescale Arm Processor
  - 320x240 Color LCD display
  - Switch, potentiometer
  - Accelerometer with gyroscope
  - Tri-color LED
  - Learn how to interface various devices
- Final project
  - Heart-rate monitor– a mini ECG
  - LCD display of heart trace
  - Measure heart rate, basic arhythmia detection

Freescale MK20DX256VLH7 processor

Core		System	Memories		Clocks
ARM® Cortex®-M4 50/72/100/120 MHz		Internal and External Watchdogs	Program Flash (32 KB to 1 MB)	SRAM 8 to 256 KB	Phase-Locked Loop
Debug Interfaces	DSP	16-Ch. DMA	Serial Programming Interface (EZ Port)	External Bus Interface (FlexBus)	Frequency-Locked Loop
Interrupt Controller	Floating-Point Unit (FPU)	Low-Leakage Wake-Up Unit	NAND Flash Controller	Cache	Low/High-Frequency Oscillators
		Memory Protection Unit (MPU)	FlexMemory (32 to 512 KB) (2 to 16 KB EE)		Internal Reference Clocks

Security and Integrity	Analog	Timers	Communication Interfaces		HMI
Cyclic Redundancy Check (CRC)	16-bit ADC	FlexTimer	I²C	I²S	GPIO
Cryptographic Acceleration Unit (CAU)	Analog Comparator with 6-bit DAC	Programmable Delay Block	UART (optional ISO 7816)	USB Device Charger Detect (DCD)	Xtrinsic Low-Power Touch-Sensing Interface
HW Tamper Detection Unit	PGA	Periodic Interrupt Timers	SPI	USB On-the-Go (LS/FS)	
Random Number Generator	12-bit DAC	Low-Power Timer	CAN	USB Voltage Regulator	
	Voltage Reference	Independent Real-Time Clock (RTC)	Secure Digital Host Controller (SDHC)	USB On-the-Go (HS)	
		Carrier Modulator Transmitter			

14

CSE466 Labs

- Final Project – Using the Teensy 3.1
- Build a Heart Rate Monitor

Assignment for Friday:

- Review the K20 Sub-Family Reference Manual
MK20DX256 Manual (8.0M PDF), for Teensy 3.1 (This manual has all the useful programing info)

  - *Chapter 2: Introduction and Functional Modules*

  Download here:

  http://courses.cs.washington.edu/courses/cse466/14au/pdfs/K20P64M72SF1RM.pdf

  (link is on the Resources page...)