4. Quiz 1, Lab 1, Gumstix Introduction

18-349: Embedded Real-Time Systems

Costas Akrivoulis
Carnegie Mellon University
September 14, 2012
Agenda

• Quiz 1 solutions
• Lab 1
• Hardware (Gumstix)
• Software
  – Toolchain, workflows
• Hand out quizzes, hardware kits
Quiz 1 Statistics

Quiz 1 Histogram

- Frequency
- Bin
- N: 64
- Min: 56
- Max: 95
- Mean: 76.13
- Std Dev: 10.48
Quiz 1 Solutions

• Review solutions to Quiz 1

• If there are questions, please contact the TAs
  – Email, office hours
Lab 1 – Introduction to Gumstix and ARM

- Out: Tonight! (September 14, 2012)
- Due: Saturday, September 29, 2012 @ 11:59pm
- Official handout on course website

- You’ll be able to practice the ARM you’ve just learned by writing some code!
Lab 1 – Lab Contents (1/2)

• Version Control
• Getting acquainted with the Gumstix
  – Making microSD bootable, installing system files
  – Setting up serial communication
    • Used for interactive session, file transfer
    • Bonus: Bluetooth
  – U-Boot, Gumstix OS
Lab 1 – Lab Contents (2/2)

- Writing some ARM code
  - Hello world
  - Low-level optimizations (assembly)
  - High-level optimizations (C)

- Tools for working with ARM code
  - make, objdump / readelf, gdb
  - Bonus: cross compilation, emulation, workflow
Lab 1 – Version Control

• Use version control to keep track of your code
  – You’ll have to use SVN for part of the lab
    • Basic tutorial in lab writeup
  – Feel free to use whatever you’re comfortable with
    • I strongly recommend using Git 😊
    • http://www.progit.org/book/
Lab 1 – Gumstix Hardware

• Gumstix Verdex-Pro 400xm-bt
  – Intel PXA270 processor (ARMv5), 16MB flash mem
• Robostix expansion board (mostly unused)
  – Atmel ATmega128 microcontroller
• 5V power adapter
• Acroname USB serial interface connector
  – Also serial extension cable + USB cable
• microSD card + SD card adapter
• Bluetooth antenna (optional)
Lab 1 – Gumstix Hardware

- USB A-B cable to USB port on host computer
- Acroname USB-to-serial connector
- Serial extension cable
- PXA-AVR UART jumper (not provided & not needed)
- 5V power adapter to power outlet
- Micro-SD card
- verdex-pro + robostix
- Bluetooth antenna
Lab 1 – Gumstix Operating System

• Separate handout describes how to set up OS
  – Link in lab write-up

• Use Linux machine to format and partition microSD card (with adapter)
  – Need root privileges to run disk tools... can’t use CMU machines to do this

• We have a file system you can just drop in
  – And some boot scripts too
Lab 1 – Serial Communication

• Used to get an interactive prompt, and to transfer files to the gumstix
  • PuTTY, HyperTerminal, minicom, screen
    – kermit, zmodem to transfer files

• Use Bluetooth for a faster, wireless connection
  – completely optional
Lab 1 – U-Boot and Gumstix OS

• You’ll likely spend most of your time in the Gumstix OS for Lab 1
  – You’re writing user-mode code right now
  – Gumstix OS is a barebones Linux installation with compiler and related build tools

• U-Boot is the Universal Bootloader
  – Will allow us to run our kernel in later labs
  – Provides a basic Exports API for us to leverage later
    • Implements essential services like printing to the screen
Gumstix Demo

- Connecting to Gumstix (serial)
- Looking at U-Boot prompt, booting into Gumstix OS
- Sending a file
- Running some ARM code
Lab 1 – Writing ARM Code

• Hello World (ARM assembly)
• Basic calculator and library (C)
• Low-level optimization (assembly)
  – Optimize mysterious strTable function called 1000 times
• High-level optimization (C)
  – Naïve algorithm given to find missing Oddball integer among array of size $2n - 1$
  – Optimize in two different ways at a high level while maintaining correctness
    • Write different algorithms, don’t just tweak the naïve impl.
Lab 1 – Using Developer Tools

• Writing a basic Makefile and using make
  – Tool to automate your build process

• Analyzing binaries with objdump and readelf
  – Will dump ARM assembly of a binary so you can quickly examine what code is being executed
  – Prints binary information (headers, symbols)

• Debugging your code with gdb
  – Single-stepping, breakpoints
Gumstix Workflow (1/5)

• Remember, the Gumstix is a constrained, embedded system
  – Low-power, low-memory, slow serial link
  – Not always the most pleasant to develop on

• Thankfully, we have the tools to work remotely too

• Try out both on-line and off-line development
Gumstix Workflow (2/5)

• We **highly** encourage you to stick to off-line development
  – Much better tools
    • vim instead of vi (glorious undo), git
  – You and your teammates can work concurrently
    • Having one Gumstix can sometimes constrain the team
  – Faster iteration
    • Compilation takes less time
    • Less time wasted waiting for Gumstix to reboot
  – Less chance for data loss should disaster strike
Gumstix Workflow (3/5)

• How do we facilitate off-line development?
  – x86 and ARM are different architectures
    • You can’t just run ARM code on your laptop!
• **Cross-compiling**: there *IS* a better way!
  – Toolchains targeting ARM, but running on x86
    • gcc, gdb, objdump, readelf, and friends
  – Cross-compiler available for CMU machines
    • Check out cross-compilation.pdf
    • Look for *gnueab*i* and *arm-linux* in your favorite Linux distribution’s package manager
Gumstix Workflow (4/5)

• Take it one step further and emulate!
  – Using QEMU, a machine emulator, you can emulate the Gumstix as a virtual ARM system
  – Debug your programs using gdb
    • Remote, symbolic debugging of the QEMU instance
    • Invaluable for kernel code debugging later on
  – Many thanks to Alex Crichton for the tutorial!

• Increases your team’s throughput 😊
Gumstix Workflow (5/5)

• **Recommended workflow:**
  – Write code on desktop / laptop
    • Use version control if you’d like
  – Cross-compile and resolve compiler warnings
    • Can try targeting x86 if it’s platform-independent code
  – Transfer to Gumstix once you’re ready to test
  – Keep workflow consistent
    • Avoid editing files on Gumstix – you’ll forget to transfer them back and end up clobbering changes
Demo – Gumstix Workflow

• Cross-compiling a C file on CMU machines
• arm-linux-objdump the resulting binary
Hardware Kits

• Write your team members’ names on the hardware box
• Register with us with what kit you have
• Make sure all equipment is there

• Start Lab 1 early! – we want to make sure the hardware works sooner rather than later
Summary

• Quiz 1 solutions
• Lab 1
  – Technical details, logistics, plan of attack
• Hardware (Gumstix)
• Software (toolchain, workflows)
• Quizzes + hardware kits
Gumstix Hardware Kit Label

• Team number
• Team member names, andrew IDs

Team Number: ___

Member: Person 1 (andrewID)
Member: Person 2 (andrewID)
Member: Person 3 (andrewID)