ECE 271 – Microcomputer Architecture and Applications Lecture 4

Vince Weaver

http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

31 January 2019

Announcements

- Read Chapter #1 of the book.
- We have a grad TA, Colin Leary, who will be having office hours on Wednesday at 2pm.

Gitlab Update

- I have posted the initial gitlab directions to the course website
- This was delayed due to the course website (the whole ECE website) being down yesterday. Hopefully that won't happen again.

LCD stuff for Lab

- Chapter 17. A bit confusing at times. 17.4 and after is just informational, not useful for this lab.
- For a detailed view see Microchip AN658.
- What is a Liquid Crystal Display? How is it different from LED?

  Polarized light, crystals that change polarization when apply power
- Fairly low power (compared to LED at least)
- Need some sort of backlight

- Applying DC voltage for too long can damage, so circuit must provide an AC voltage centered on zero.

Static LCD

- Common and segment. Square wave to common, square to segment
- If in phase, voltage across is zero, off
- If out of phase, voltage across is $+/$-V, which has display on

Multiplexed

- Static vs Multiplexed: static each pin drives one segment. Would take 96 GPIOs
- Duty Ratio vs Bias
- Duty Ratio of 1/3 means three segments, driving any given one 1/3 of time
- Our board has duty ratio of 1/4, so can drive with 28 pins (24 + 4 common)
- It is less bright with higher duty ratio.
- Complex series of alternating voltage levels needed for

this. See the textbook for full details.

LCD on our board

- Static vs Multiplexed: static each pin drives one segment. Would take 96 GPIOs
- Our board has duty ratio of 1/4, so can drive with 28 pins (24 + 4 common)
- 14-segment chars*6, colon, decimal point, 4 bars
- We don't have to drive the raw voltages (thankfully) but we do have 28 GPIOs to drive, and they are scattered about somewhat randomly :(

Steps to Program Them

- LCD Clock Initialization
  - Disable RTC clock protection (LCD+RTC share same clock)
    Write "secret code" 0xCA and 0x53 to register
  - Enable LSI clock
  - Select LSI as LCD clock source
  - Enable LCD/RTC clock
- Configure LCD GPIO Pins to be LCD (Alternative mode 11 0xd)

- PortA: 6,7,8,9,10,15
- PortB: 0,1,4,5,9,12,13,14,15
- PortC: 3,4,5,6,7,8,11
- PortD: 8,9,10,11,12,13,14,15

- LCD Config
  - Set BIAS to 1/3
  - Set Duty to 1/4
  - set contrast to max
  - Set pulse on value
  - Disable MUX_SEG
  - Select interval voltage

- Wait for FCRSF
- Enable LCD
- Wait to see if enabled
- Wait for LCD booster to be ready

- LCD loop
  - Spin waiting for LCD_RAM to not be protected
  - Once available, update the LCD_RAM memory

    7 bytes corresponding to the pins we want to turn on
  - Set the UDR flag which says we want to update the display with our value
  - Wait for update to finish, then loop

Steps to Program Them

- Work out what you want to display. Which segments
- Then look at huge lookup table to see what pins correspond to this
- Then set this in the RAM
- You can make a function/lookup that automates this.

Why double-buffering

- Write to one buffer, display to another. Then when ready, swap (display first, write to second). Repeat.
- Avoids tearing – when you are displaying partially old and partially new data
- Avoids flicker

Intro to Computer Architecture

Parameters in an Architecture

- CISC vs RISC
- 8/16/32/64 bits
- Endianess
- Load/Store
- Instruction Size (fixed/variable)
- Number of commands to opcode
- Weird: Branch Delay slot / Zero register
- Number of registers, special registers
- Flags

Memory/Code

- Harvard vs von Neuman Architecture
- Harvard – instruction stream completely separate from data
- von Neuman – instructions are in general memory

Cotex-M4

- Like M3 but some DSP and floating point instructions
- Hardware multiply/divide and saturating instructions
- In-order, 3-stage pipeline, branch speculation, no caches
- Thumb-2 architecture

ARM Instruction Set Encodings

- ARM – 32 bit encoding
- THUMB – 16 bit encoding
- THUMB-2 – THUMB extended with 32-bit instructions
  - STM32L *only* has THUMB2
  - Original Raspberry Pis *do not* have THUMB2
  - Raspberry Pi 2/3 *does* have THUMB2
- THUMB-EE – extensions for running in JIT runtime
- AARCH64 – 64 bit. Relatively new. Completely different from ARM32

Thumb-2 encoding

```
ADD{S}<c>.W <Rd>,<Rn>,<Rm>{,<shift>}
```

15-11: 11101

10-9: 01

8-5: 1001

4: S

3-0: Rn

15: 0

14-12: imm3

11-8: Rd

```
7-6: imm2
5-4: type
3-0: Rm
```