Course Logistics

Course logistics can be found on the website

---

Transition: From 429 to 439

What kind of architecture is it?

What else can we say?

You switch it on, and then?

---

Instruction Pipeline

ARM7TDMI

Fetch: Fetch instruction from memory (at PC)

Decode: Decode opcode, operands (reg or imm), etc.

Execute: Read registers from regfile, shift, ALU, write regs back to regfile
Boot Sequence (ARM)

- Processor is in an uninitialized state.
  - Basic features like clock setup needs to be performed in a system-specific way before the processor can execute complex code
- Boot-ROM
  - Typically hard-wired, provided by the SoC manufacturer
  - Programs the clocks, basic interrupt setup, etc.
  - Locate software bootloader and load it
- Software Bootloader
  - Load OS kernel into memory
  - Pass execution to the kernel

Course Overview

- We will start right above the bootloader.
- We will have several projects in the course.
  - Write a non-trivial piece of system software.
  - Write parts of an Operating System.
    - On ARM
    - Can run on the Raspberry Pi!
- We will hopefully have a lot of fun!

Boot Sequence Raspberry Pi

- Systems starts with GPU on, ARM core off, SDRAM disabled.
- GPU executes first stage bootloader from ROM
  - Reads the SD card, loads second stage bootloader (bootcode.bin) from SD card into L2 and runs it.
- GPU executes second stage bootloader from L2
  - Enables SDRAM, MMU, reads the third stage bootloader (loader.bin) from SD card into RAM and runs it.
- GPU loads kernel.img into SDRAM at physical address 0x0 and releases reset on the ARM core

Project 1

- Compiling the Linux Kernel
- Optimizing your kernel
Backup Slides

Instruction Set

<table>
<thead>
<tr>
<th>Cond</th>
<th>Rb</th>
<th>Rj</th>
<th>OpCode</th>
<th>S</th>
<th>Rm</th>
<th>Rd</th>
<th>Operand 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Shift</td>
<td>Rm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Rotate</td>
<td>Immediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1110 00 1 0100 0 0000 0000 0000 0000001

Barrel Shifter
- 8 bit Barrel Shifter
- $n \cdot \log(n)$ MUX

- Shifts and rotates in single cycle
  - [Link to Barrel Shifter](http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/10-gates/60-barrel/shifter8.html)

von Neumann vs. Harvard Architecture
- **Von Neumann Architecture**
  - Shared bus system for instructions and data loads
  - CPU cannot load instruction and data at the same time

- **Harvard Architecture**
  - Separate storage and bus system for instructions and data (pure)
  - Can load instruction and data at the same time