Software Tools for Low-Level Software and Operating Systems Classes

Max Walter & Sven Karlsson
Technical University of Denmark
Introduction

• I have been involved in a number of classes
  • Real Time and Embedded Systems
  • Introduction to Operating Systems
  • Computer Organization and Assembly
Introduction

- I have been involved in a number of classes
  - Real Time and Embedded Systems
  - Introduction to Operating Systems
  - Computer Organization and Assembly

- The common theme: **Hardware** from a **software** perspective
That Being Said

• These are the first architecture classes students take
  • May be the last

• Introductory computer science becoming increasingly higher level
  • Why not introductory computer science architecture?
Outline

• Motivation
• How to increase intuitive understanding
• Virtualization as a platform
• Our solution
• Examples
The Common Theme

• Some software development requires (some) architectural knowledge
  • Operating systems / Device Drivers
  • Timing / performance critical code
  • Compilers
  • Bare-metal software
  • Etc…
Does Anyone Do That Anymore?

- Yes, and it is becoming increasingly important
Does Anyone Do That Anymore?

• Operating Systems
  • Multi- / Many- Core
  • Accelerators
• “The Machine”
Does Anyone Do That Anymore?

• Embedded Systems
  • Increasingly capable mobile devices
  • Massive proliferation of embedded devices with “Onboard Processing”
• IoT / Cyber Physical Systems
And Yet

- Many universities are moving introductory classes to higher level languages
  - Matlab
  - Java
  - Python

This Isn’t a Bad Thing

- It reflects the current nature of computer science education
- Algorithms and concepts over details

```c
char *x = "hello";
char *y = "world";
char *z = malloc(strlen(x) + strlen(y) + 2);
strcpy(z, x);
strcat(z, " ");
strcat(z, y);
printf("%s
", z);
free(z);
```
This Isn’t a Bad Thing

• It reflects the current trends of commercial programming

  • App market grossed $44.8 billion in 2016¹

¹) new zoo global market report
But Abstractions Matter

- Hardware is treated to be a black box
  - It exists just to run code
  - Sometimes you have to deal with it
But Abstractions Matter

- In these classes we have to lose the abstractions

- Stack, heap, busses, interrupts, traps, assembly, DMA, processes, virtual and physical memory, cache sizes, memory hierarchy, instruction latencies, scheduling, etc…

- They all matter
How to Increase Conceptual Level

- We want to decrease difficulty curve
  - Better tools
  - Better visualization
First Step

• Simply integrating Bochs and Eclipse

• Not unheard of
First Step

- Simply integrating Bochs and Eclipse
  - Not unheard of
  - Need lower level access than what these tools provide
  - No user-space to run the debugger
First Step

---

error: Can’t get controller info...

---

long start_address = copy_ELF(sl Elf.Images[0]);
init_context(&contexts[0], 0, start_address);
set_active_context(&contexts[0]);
go_to_user_mode();

---

long
First Step
First Step

- So simple; trite even
Can We Do Better?

- There are so many tools that could be useful
  - Profilers
  - Tracing tools
  - Visualization tools
- Instrumentation Frameworks
Can We Do Better?

• Virtual execution environments have a lot to offer
  • They expose the entire hardware platform

• Provide higher level access so we don’t have to focus on details
Virtualization is Used Extensively

- But as a tool, not as a platform
A Framework for Exposing Virtualization Internals

- Started out as a way to test our kernel
- Combined with tools and assignments from past classes
A Framework for Exposing Virtualization Internals
Create a Way to Expose the Internals

• What do we want access to?
  • Exceptions and Interrupts
  • System calls
  • Physical / virtual memory, page tables
Create a Way to Expose the Internals

• What do we want access to?
  • Exceptions and Interrupts
  • System calls
  • Physical / virtual memory, page tables

• To start
What Do We Build On?

• Bochs?
• QEMU?
• gem5?
• Xen?
• Our own?
• No good answer
What Can/Have We Done With It

• Very much a work in progress
Samples

• Multitasking
• Cooperative and preemptive
Samples
Samples
Samples
### Samples

<table>
<thead>
<tr>
<th></th>
<th>1700</th>
<th>1800</th>
<th>1900</th>
<th>2000</th>
<th>2100</th>
<th>2200</th>
<th>2300</th>
<th>2400</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>160 (0xa0)</td>
<td>170 (0xaa)</td>
<td>190 (0xb4)</td>
<td>190 (0xbe)</td>
<td>200 (0xc8)</td>
<td>210 (0xd2)</td>
<td>220 (0xdc)</td>
<td>230 (0xe6)</td>
<td></td>
</tr>
<tr>
<td>85 (0x55)</td>
<td>255 (0xff)</td>
<td>180 (0xb4)</td>
<td>115 (0x73)</td>
<td>60 (0x3c)</td>
<td>15 (0xf)</td>
<td>235 (0xeb)</td>
<td>210 (0xd2)</td>
<td></td>
</tr>
<tr>
<td>40 (0x28)</td>
<td>252 (0xfc)</td>
<td>48 (0x30)</td>
<td>73 (0x49)</td>
<td>75 (0x4b)</td>
<td>60 (0x3c)</td>
<td>32 (0x20)</td>
<td>124 (0x7c)</td>
<td></td>
</tr>
<tr>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td></td>
</tr>
<tr>
<td>0x3296fa</td>
<td>0x3296fa</td>
<td>0x3296fa</td>
<td>0x3296fa</td>
<td>0x3296fa</td>
<td>0x3296fa</td>
<td>0x3296fa</td>
<td>0x3296fa</td>
<td></td>
</tr>
<tr>
<td>0x6428e2</td>
<td>0x6428e2</td>
<td>0x6428e2</td>
<td>0x6428e2</td>
<td>0x6428e2</td>
<td>0x6428e2</td>
<td>0x6428e2</td>
<td>0x6428e2</td>
<td></td>
</tr>
<tr>
<td>0x96b42d</td>
<td>0x96b42d</td>
<td>0x96b42d</td>
<td>0x96b42d</td>
<td>0x96b42d</td>
<td>0x96b42d</td>
<td>0x96b42d</td>
<td>0x96b42d</td>
<td></td>
</tr>
</tbody>
</table>
Work In Progress

- Need a proper visualization system
- Wanted to expose AXI bus signals
  - Need a way to record/playback
  - Need custom IP
- Need a better way to find and match symbols
Conclusions

• Introductory computer science is moving to a higher level
  • And will continue to do so
• Introductory architecture can as well
  • Still lots of questions