RTL through OS

B9 (Formerly B10)

Reid Long, Teguh Hofstee

What are we building?

- RISC-V Processor
  - RV32I-MAS
    - Integer
    - Multiplication and Division
    - Atomic
    - Supervisor
- Kernel
  - Pebbles Specification
- Architecture
  - Single Core
  - 512 MB of DRAM
  - 2 Level, Hardware Walked, Page Table for Virtual Memory

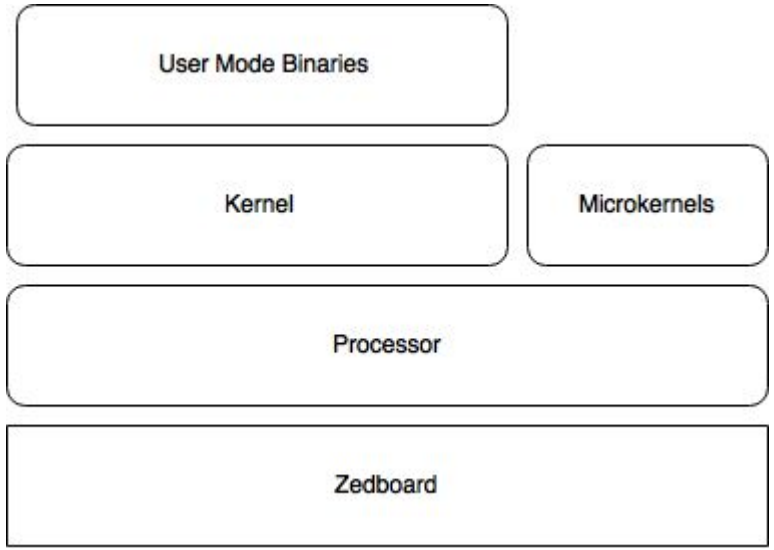
Competition

- There are other CISC and RISC processors
  - MIPS R10000 (outdated ISA)
  - Intel x86 (excessively complex ISA)
- There are other Operating Systems
  - Linux, Windows, macOS
- All require years of development and tens to thousands of engineers
  - We are scaling down to target a scope achievable in a semester
  - Our goal is to develop a unique skill set that spans both kernels and processors
- *In a world ravaged by nuclear fallout and zombies...*
  - Re-create ancient blood rituals performed by Bjarne Stroustrup at Stonehenge[1]
  - In the beginning, man created the computer and the kernel.

[1] https://www.usenix.org/system/files/1311_05-08_mickens.pdf

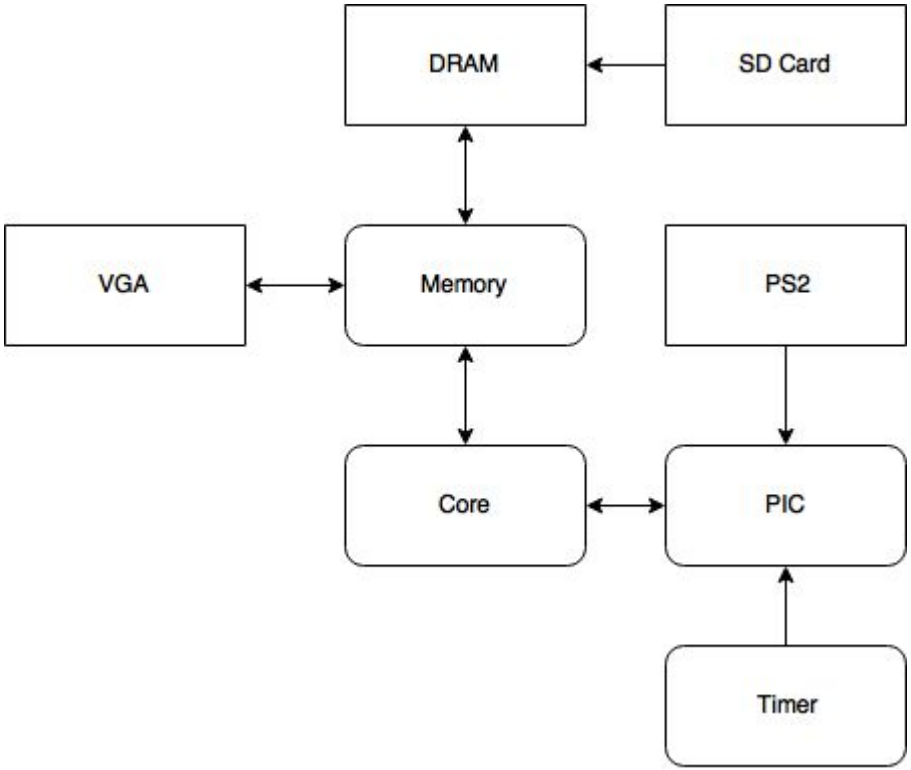
Three Phase Approach

1. Design a processor
2. Develop microkernels
3. Develop a kernel

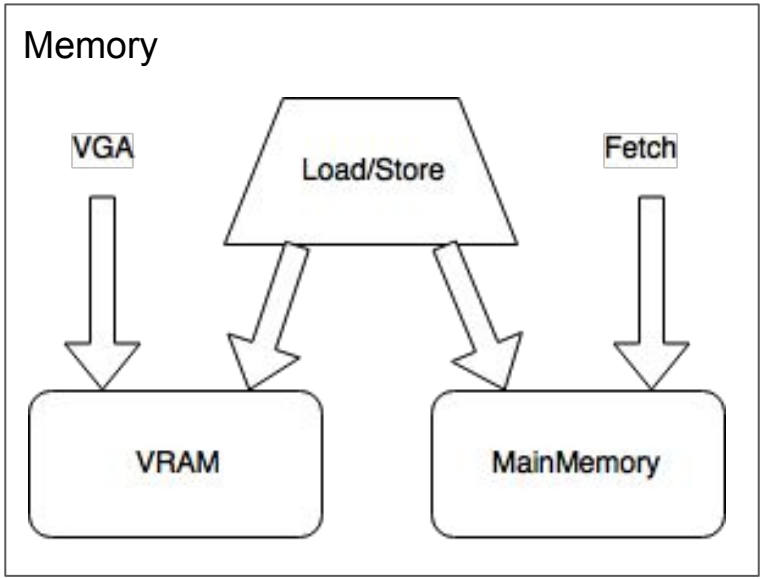


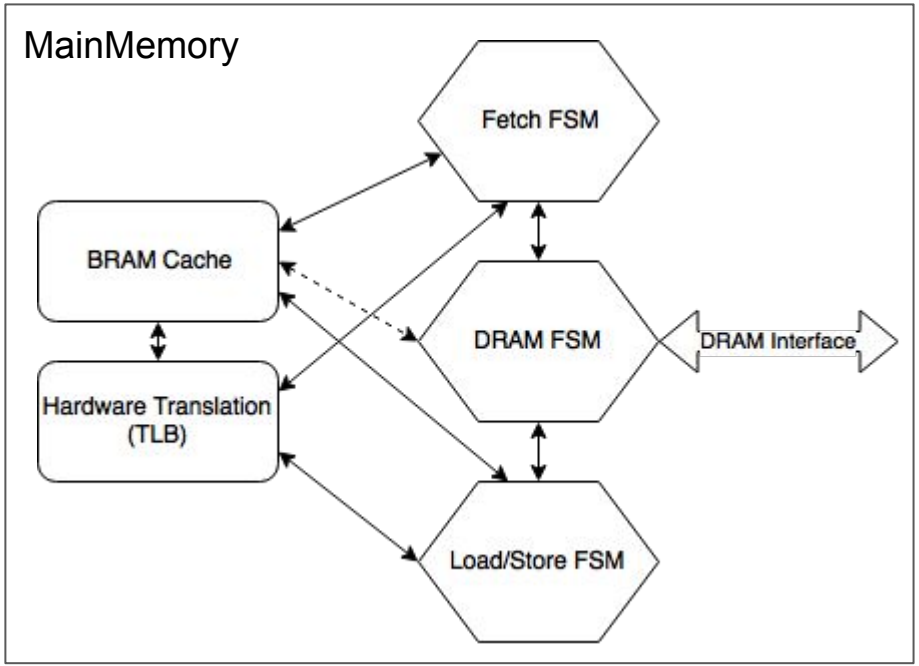
Design a Processor

- Notation
  - Rectangle: Includes External Interface
  - Rounded Rectangle: No external interface
- Custom vs. Off-the-shelf
  - VGA: Custom
  - DRAM: Off-the-shelf aren't working currently
  - SD Card: Desired Off-the-shelf (likely need custom)
  - PS2: Custom
  - Memory: Custom
  - Core: Custom
  - Timer: Trivally Custom
  - PIC: Custom

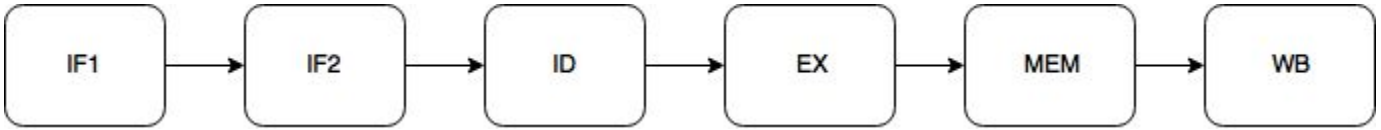


Design a Processor - Memory





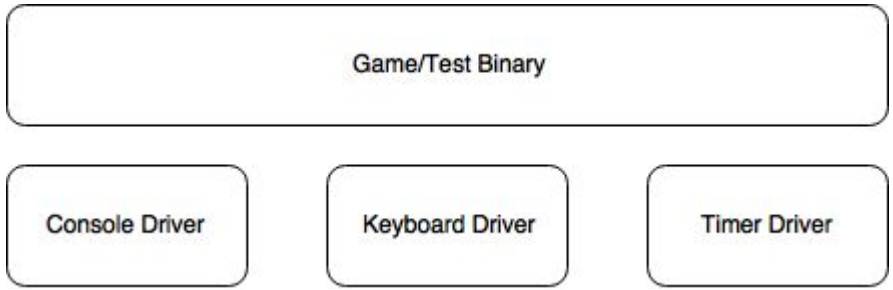
Design a Processor - Core



- 6 Stage Pipeline
- Branch Prediction and Data Forwarding
- Two Stage Instruction Fetch
  - Memory Operations have a large clock-to-out propagation delay
  - Decode stage is expensive in 447 cores
- Single Stage Memory Load/Store
  - Large clock-to-out can be lumped into the register writeback stage

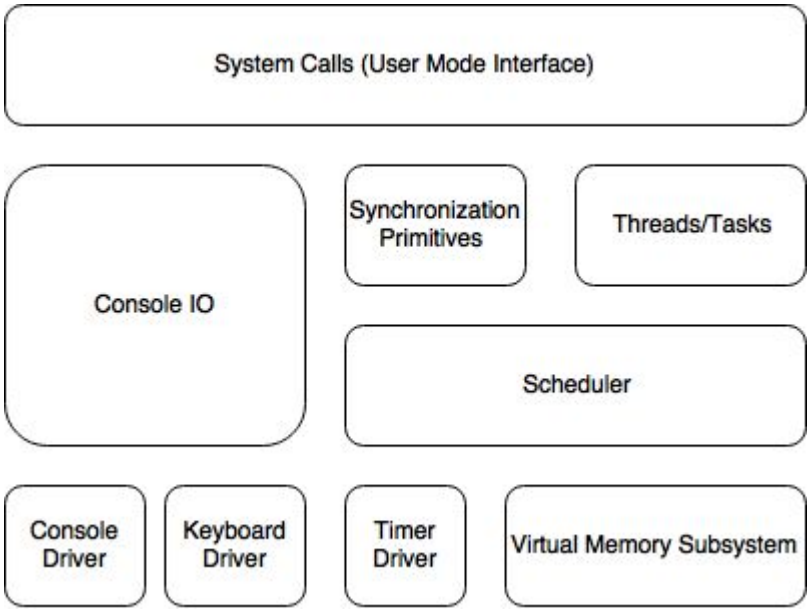
Develop Microkernels

- No virtual memory
- Same drivers as full kernel
- Targeted benchmarks to ensure correctness of processor
- Random benchmarks to increase confidence in processor



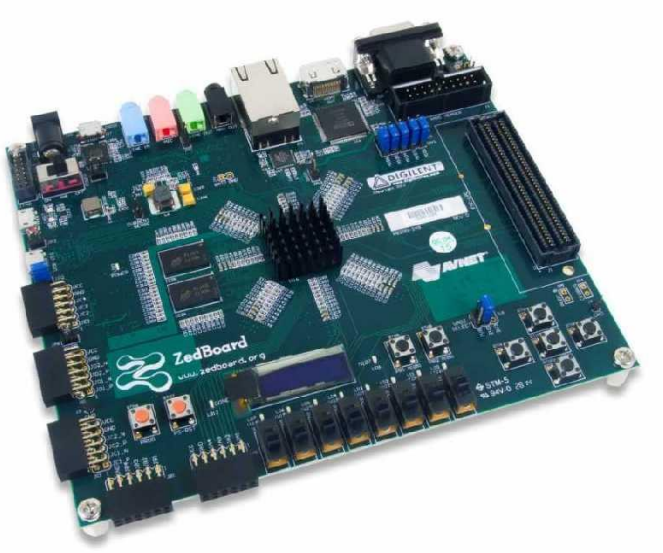
Develop The Kernel

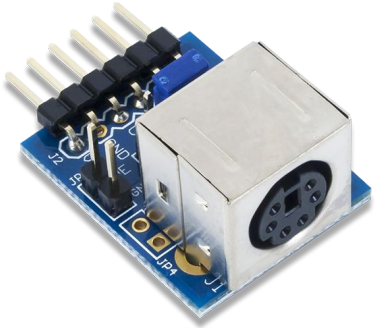
- Pebbles Kernel
  - Small specification (25 System Calls)
  - Task Life Cycle
  - Thread Life Cycle
  - Memory Management
  - Console IO
- Custom vs. Port (from x86)
  - Generic Libraries (Port)
  - Drivers (Custom)
  - Synchronization Primitives (Custom)
  - System Calls (Port/Custom)

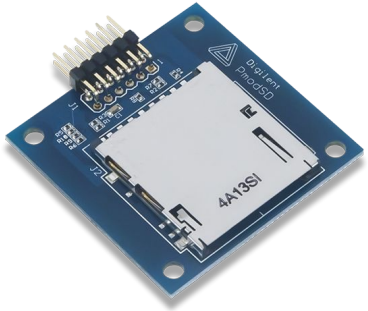


Platform

- Hardware
  - Zedboard ($0)
  - PS2 Keyboard ($0)
  - VGA Display ($0)
  - PMOD PS2 ($8.99)
  - PMOD SD ($9.99)
- Software
  - Vivado ($0)
  - Architectural Simulator (Written by Us) ($0)







Demo

- Boot microkernel games
  - Minesweeper
  - Tetris
- Boot a shell on the kernel
  - Computational Benchmarks
    - Ackermann
    - Fibonacci
    - “cho” - Continuous Hours of Operation
  - Visually Pleasing Binaries
    - Racer
    - Mandelbrot
    - Bistromath (Chess)
    - Nibbles (Snake)

Correctness - Primary Metric

- Processor Correctness
  - Goal: 100M LOA (Lines of Assembly) Coverage
  - Goal: 100 Targeted Tests to validate edge cases in instructions (70 instructions)
- Kernel Correctness
  - Goal: 150 Correctness Binaries
  - Expected: 15B Assembly Instructions Executed

Performance - Secondary Metric

- Goals
  - Boot the kernel in under 30 seconds
  - Run Tetris at 60 frames per second*
    - In a full kernel, with other tasks able to run at the same time
- Estimates

Binary	Instructions	IPC	Clock Period (ns)	Runtime (s)
Tetris (60 frames)	792,000	0.70	240	0.271
Tetris (60 frames)	792,000	0.70	10	0.011
Kernel Boot	900,000,000	0.70	240	308 (5 minutes)
Kernel Boot	900,000,000	0.70	10	12.857

Timeline

- Key
  - Blues: Software
  - Red/Orange: Hardware
  - Green: Demo/Presentation
- Current Status
  - Schedule is slipping
  - Memory control complexity is much higher than anticipated
  - DRAM AXI is inconsistent with our interpretation of documentation

Week	Milestone	Reid	Teguh
2/12		Arch. Sim.	VGA
2/19		Memory	DRAM
2/26	Design Presentations	Core	DRAM
3/5		Core	Core
3/12	Spring Break	PS2	Boot Loader
3/19	Working Processor	Kernel Drivers	Boot Loader
3/26		Minesweeper	PIC
4/2	Mid-Point Presentation	TLB	Tetris
4/9		Kernel	Kernel
4/16	Carnival		
4/23		Kernel	Kernel
4/30	Final Presentation	Demo	Demo

Risks

- Unknown Complexity
  - Schedule the components we are least confident in first to discover surprise complexity early in the schedule
- Complexity Overload
  - Utilize high quality software development techniques like pair programing, peer review, and frequent communication
- Off-the-Shelf Component Incompatibility
  - Integrate off-the-shelf components early in the schedule to discover the issues before they block later progress

Questions?