

The J Computer

Advay Mengle

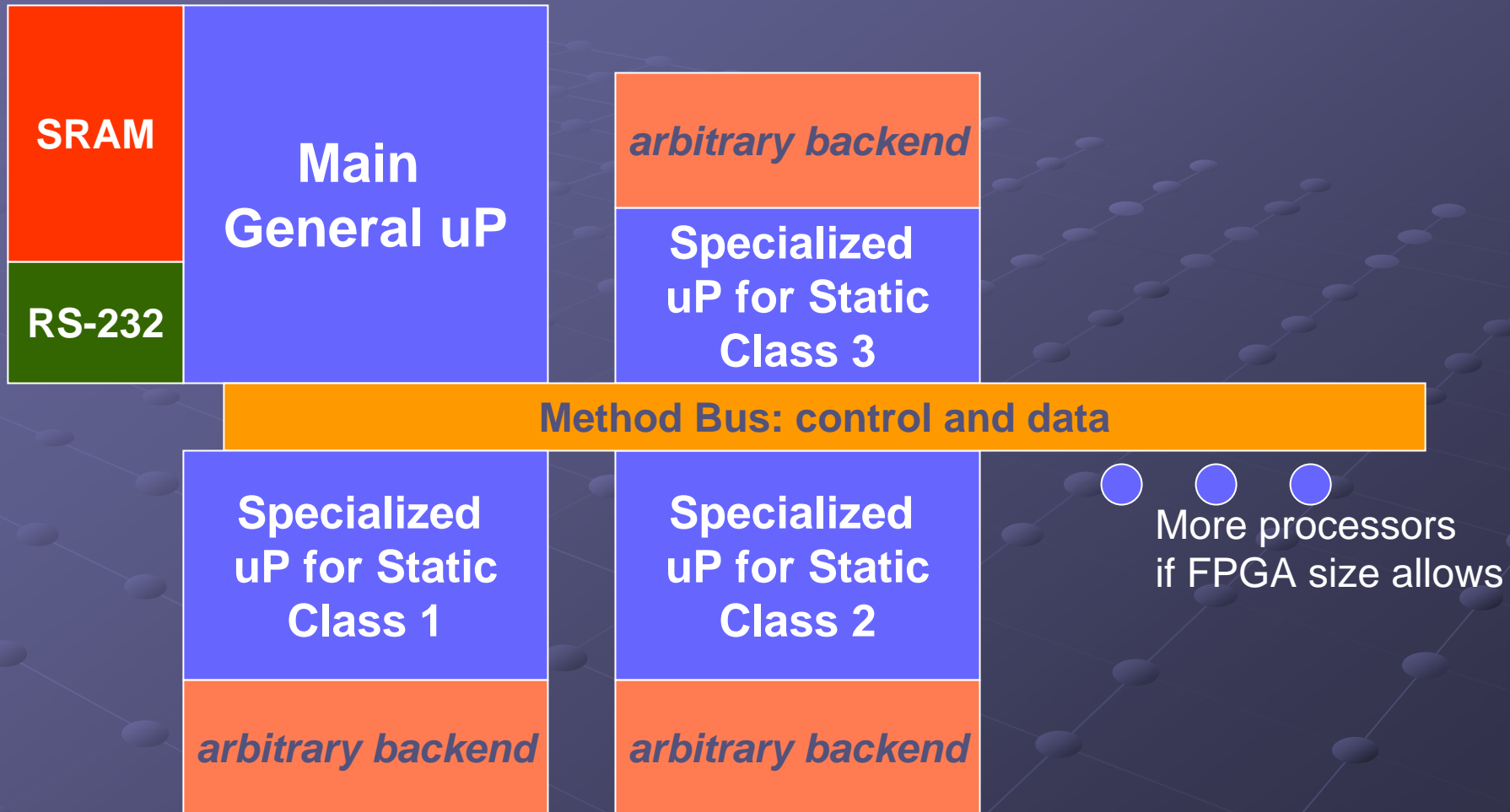
MIT 6.111, Spring 2007

April 25, 2007

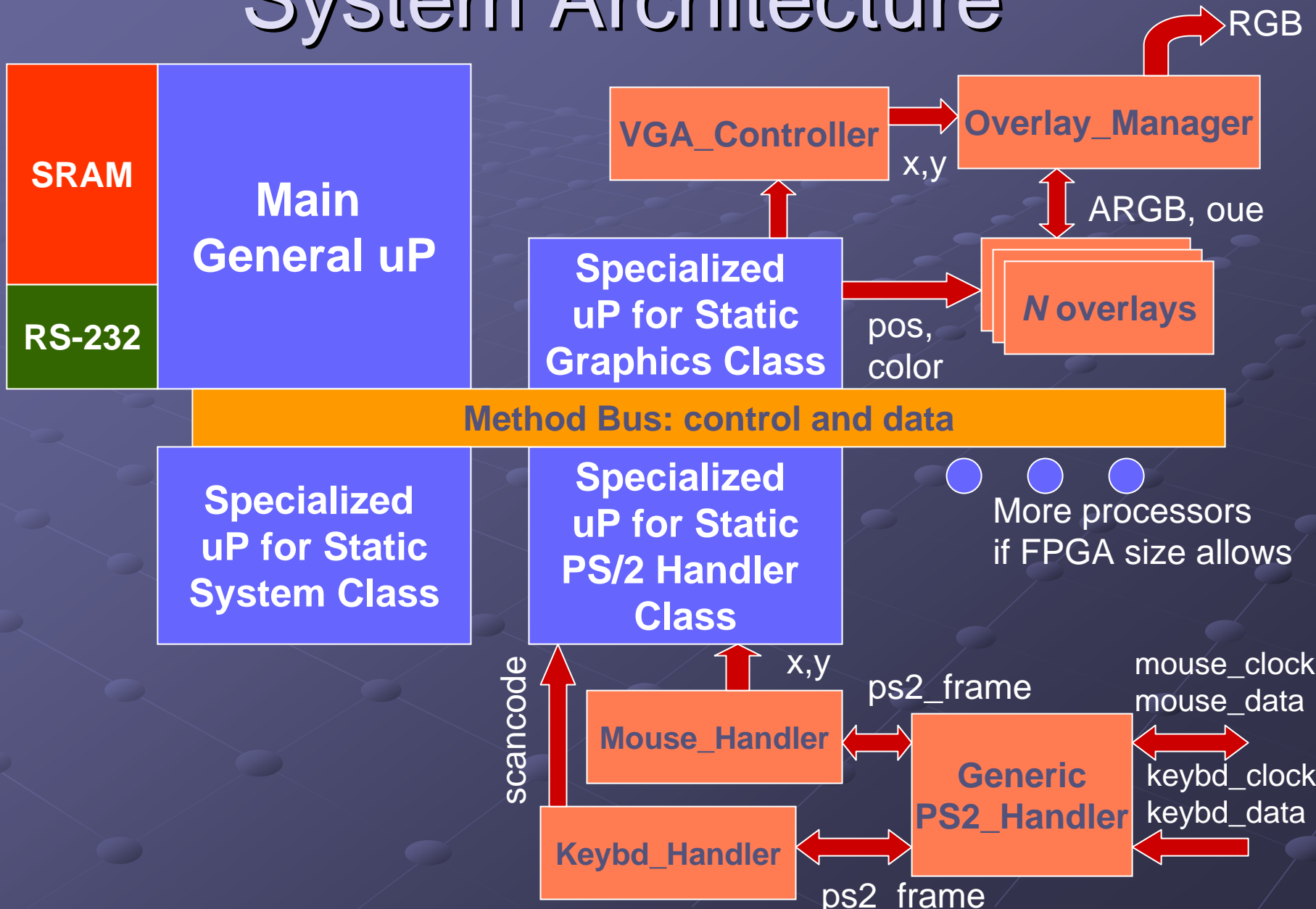
The J Computer

- Executes Java ME-like bytecode in hardware
- Constraints: (relatively) slow clock rate, not much memory
 - Use Connected, Limited Device Configuration v1.0
- Downloads user-specified class files from PC over RS-232 connection
- Interfaces with hardware devices to allow user interaction

System Architecture



System Architecture



Executing HLLs in Software vs. in Hardware

Software VM

Java Source

Compiled

javac

Java Bytecode

Verified
Interpreted
JIT-compiled

java (VM)

Native MC

Executed

μP

Native Processor

Java Source

Compiled

javac

Java Bytecode

Executed

μP

Each bytecode can be one or more native instructions

General Microprocessor: Simplified

to/from
SRAM

Bytecode Processing

Variable length instructions split into 8-bit "cells"
loading, parsing, flow control

to/from
desktop

bytecode_cell_addr

Bytecode
_Reader
reads bytecode at PC

Jump
_Control
controls PC

Invoker
calls static or
instance methods

control

Method Bus

Boot
controls boot-up

data

bytecode_cell ready

jmp_pc en index en reset (to all)

Bytecode
_to_SOP
translation

sop ready

SOP (Simple OPeration)

Processing

40-bit instructions w/ 8-bit op-code, and
either 4 8-bit params or one 32-bit literal
computation

raddr1,2

rdata1,2

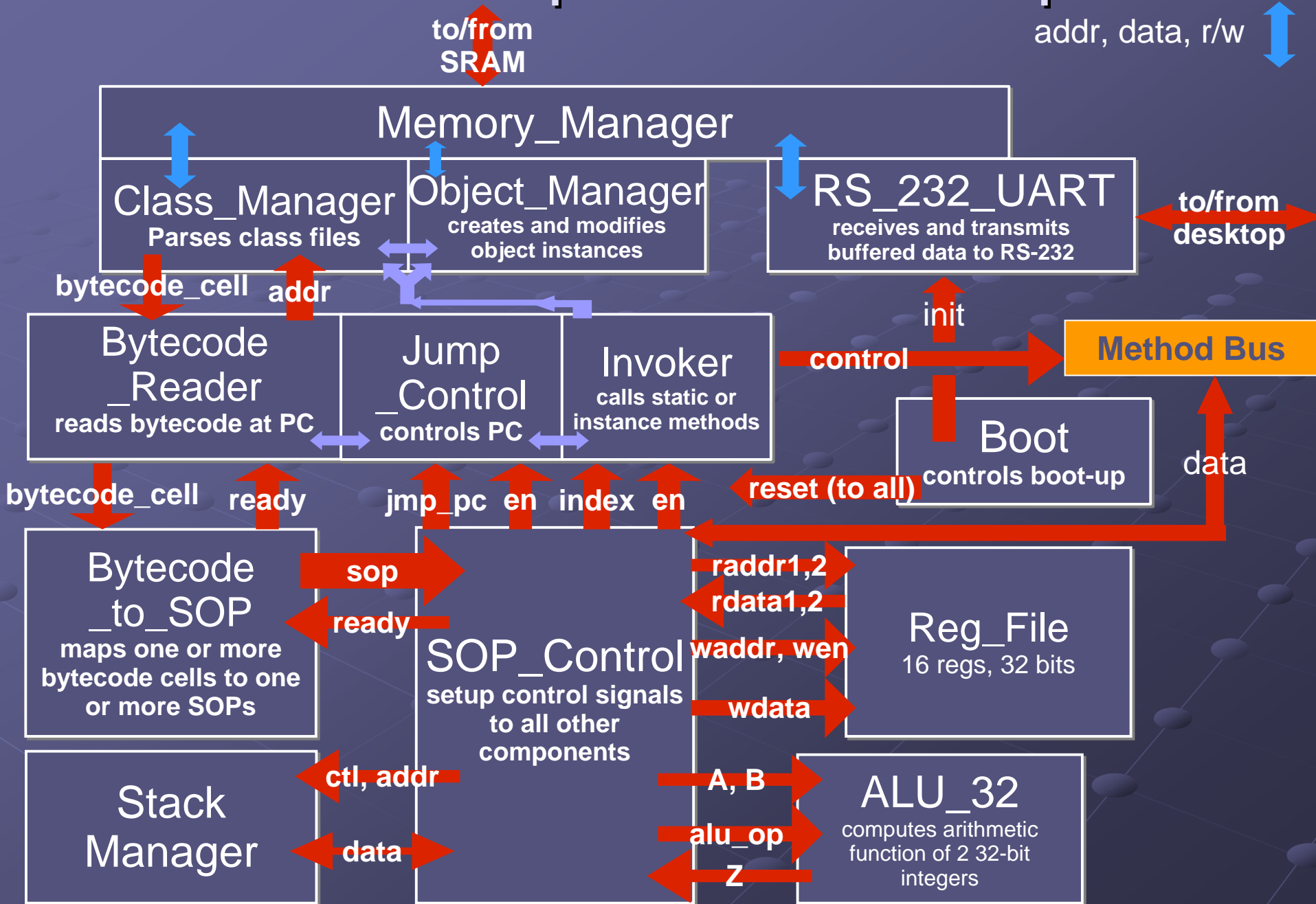
16 regs, 32 bits

Stack
Manager

ctl, addr data

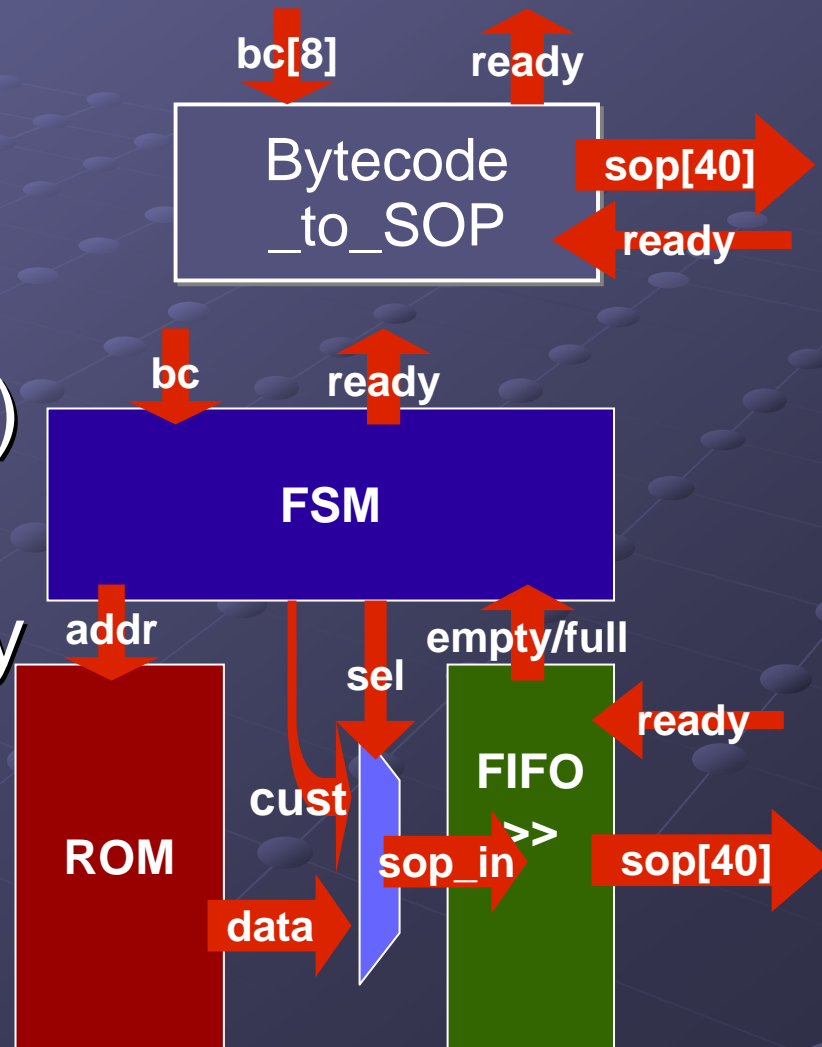
arithmetic
function of 2 32-bit
integers

General Microprocessor: Simplified



Bytecode-to-SOP translation

- Most bytecodes can be directly translated into one or more constant SOPs (looked up in ROM)
- Others may generate custom SOPs dynamically



Goals & Challenges

- Final demo: Write MIT Pong in Java
- The most complex bytecodes: **instanceof**, **invoke_virtual**
 - Need to traverse type trees
 - Raise traps via SOP and write in Java, or
 - Implement via in-BRAM data structures
- Gate count
 - Can we fit 4 processors on a single FPGA?

Questions?

