Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

Implementing RISC-V in Synchronous Message Exchange

Daniel Ramyar

Niels Bohr Institute
University of Copenhagen

eScience lunch talk
October 2, 2019

Overview

1 Introduction
  Motivation
  Synchronous Message Exchange

2 Implementing RISC-V in SME
  Single Cycle RISC-V
  Fetching instruction and increment
  Instruction decode and execution
  Memory access
  Branching
  Simple datapath
  Instructions
  Control

3 Further work

Motivation

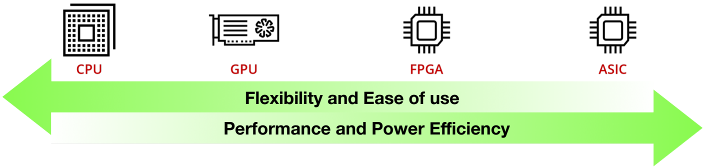
Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

- Most measurement instruments are based on Intel x86 CPU
- Limits the bandwidth at which data collection is possible
- Limits the possibility for custom solutions

Motivation

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

- RISC-V especially well suited embedding in scientific instruments
- Fully customizable = faster and more power efficient
- Could be implemented in a FPGA
- But FPGA programming is complicated



Synchronous Message Exchange

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

- Write FPGA applications within .Net framework[1]
- Enjoy productivity features of modern language
- Automatic VHDL conversion
- Based on Communicating Sequential Processes (CSP)[2]

[1] Building Hardware from C# models, Kenneth Skovhede and Brian Vinter
[2] Communicating sequential processes, Charles Antony Richard Hoare

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

Synchronous Message Exchange

- Processes are isolated no memory sharing
- Communicates with each other through channels
- Globally synchronous
- Perfect for implementing RISC-V!

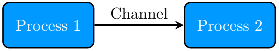


Figure: One to one

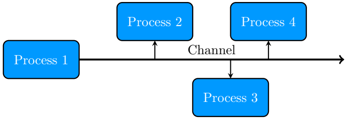
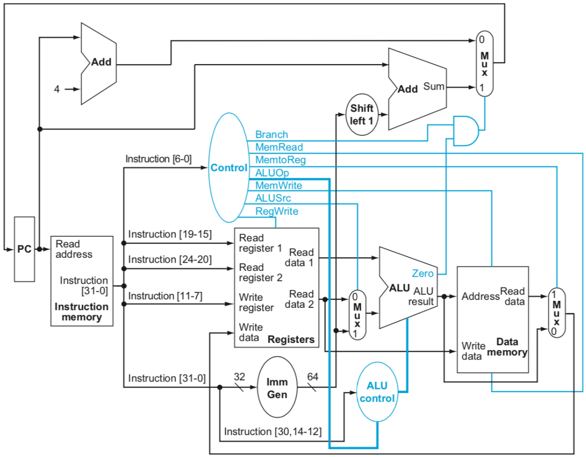


Figure: One to many

Single Cycle RISC-V

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

Full datapath for single cycle RISC-V

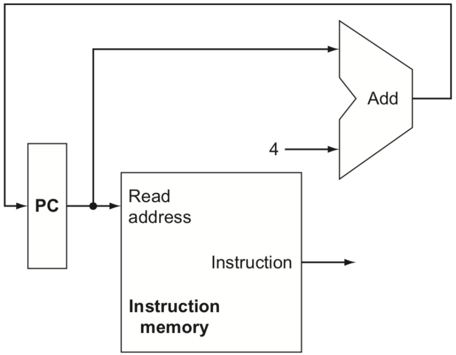


Taken from *Computer organization and design RISC V*

Fetching instruction and increment

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

We need memory for instructions and a way to remember current instruction

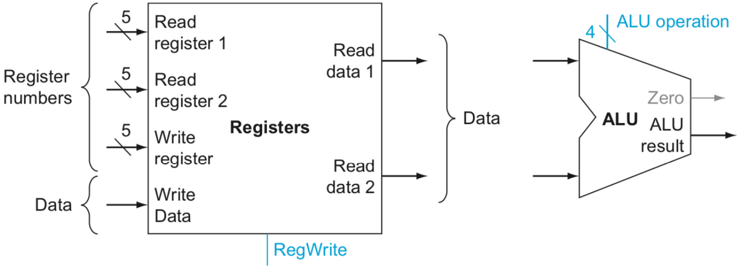


Taken from *Computer organization and design RISC V*

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

Instruction decode and execution

Storage for data we currently work with and unit for execution for instructions

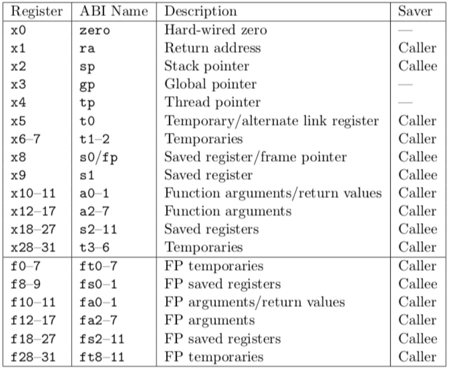


Taken from *Computer organization and design RISC V*

Instruction decode and execution

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

Conventions according to RISC-V spec sheet



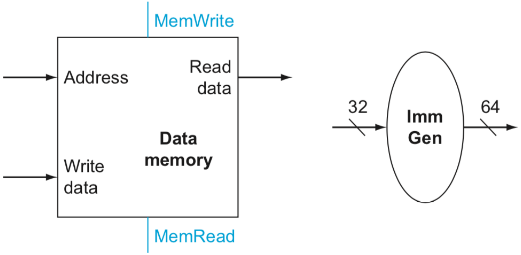
Register	ABI Name	Description	Saver
`x0`	`zero`	Hard-wired zero	—
`x1`	`ra`	Return address	Caller
`x2`	`sp`	Stack pointer	Callee
`x3`	`gp`	Global pointer	—
`x4`	`tp`	Thread pointer	—
`x5`	`t0`	Temporary/alternate link register	Caller
`x6–7`	`t1–2`	Temporaries	Caller
`x8`	`s0/fp`	Saved register/frame pointer	Callee
`x9`	`s1`	Saved register	Callee
`x10–11`	`a0–1`	Function arguments/return values	Caller
`x12–17`	`a2–7`	Function arguments	Caller
`x18–27`	`s2–11`	Saved registers	Callee
`x28–31`	`t3–6`	Temporaries	Caller
`f0–7`	`ft0–7`	FP temporaries	Caller
`f8–9`	`fs0–1`	FP saved registers	Callee
`f10–11`	`fa0–1`	FP arguments/return values	Caller
`f12–17`	`fa2–7`	FP arguments	Caller
`f18–27`	`fs2–11`	FP saved registers	Callee
`f28–31`	`ft8–11`	FP temporaries	Caller

Taken from *The RISC-V Instruction Set Manual - Volume I: User-Level ISA*

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

Memory access

We need memory unit for storing data and immediate generation unit for calculating memory address

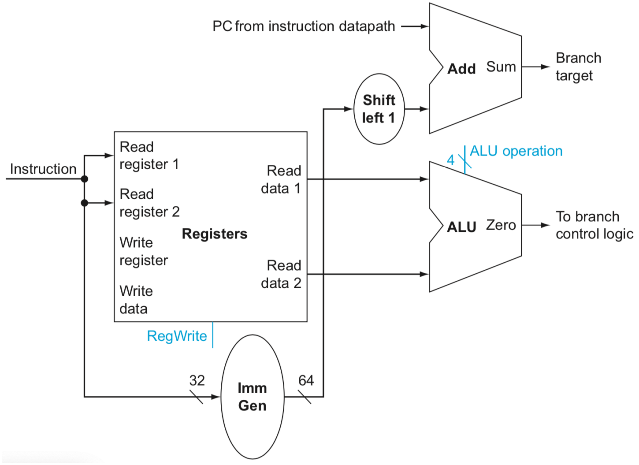


Taken from *Computer organization and design RISC V*

Branching

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

Additional unit needed for branching instructions

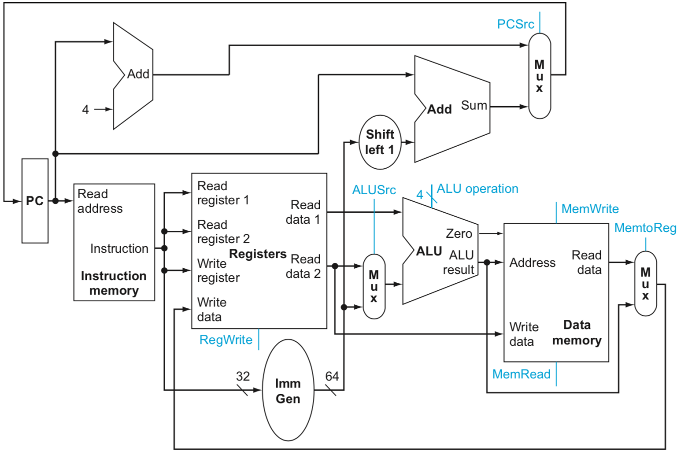


Taken from *Computer organization and design RISC V*

Simple datapath

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

Wiring it up we have our simple datapath



Taken from *Computer organization and design RISC V*

Instructions

Instruction Formats

31 25	24 20	19 15	14	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type
imm[11:5] | imm[5]	imm[4:0]	rs1	funct3	rd	opcode	I-type*
imm[11:5]	rs2	rs1	imm[4:0]	rd	opcode	S-type
imm[12|10:5]	rs2	rs1	imm[4:1|11]	rd	opcode	B-type
imm[31:12]				rd	opcode	U-type
imm[20|10:1|11|19:12]				rd	opcode	J-type

* This is a special case of the RV64I I-type format used by slli, srli and srai instructions where the lower 6 bits (imm[5] and imm[4:0]) are used to determine the shift amount (shamt). If slliw, srliw and sraiw are used it should generate an error if imm[5] $\neq 0$

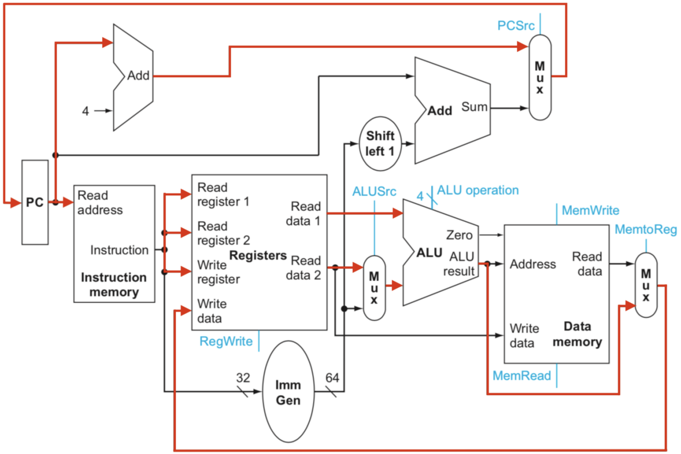
RV64I Base Instructions

Name	Fmt	Opcode	Funct3	Funct7/ imm[11:5]	Assembly	Description (in C)
Add	R	0110011	000	0000000	add rd, rs1, rs2	rd = rs1 + rs2
Subtract	R	0110011	000	0100000	sub rd, rs1, rs2	rd = rs1 − rs2
AND	R	0110011	111	0000000	and rd, rs1, rs2	rd = rs1 & rs2
OR	R	0110011	110	0000000	or rd, rs1, rs2	rd = rs1 | rs2
XOR	R	0110011	100	0000000	xor rd, rs1, rs2	rd = rs1 ˆ rs2

Simple datapath

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

The R-Type instruction datapath would look like

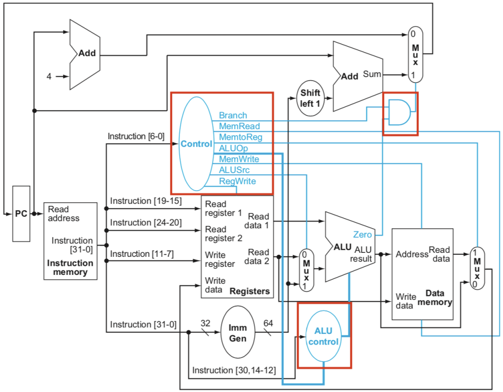


Taken from *Computer organization and design RISC V*

Control

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work

We need control unit to determine path for instructions. Main Control and ALU control are separated to simplify complexity of system.



Taken from *Computer organization and design RISC V*

Further work

- Implement rest of base instructions
- Pipeline the datapath
- Test on a FPGA

Introduction
Motivation
Synchronous Message Exchange
Implementing RISC-V in SME
Single Cycle RISC-V
Fetching instruction and increment
Instruction decode and execution
Memory access
Branching
Simple datapath
Instructions
Control
Further work