RISC-V RTL Design Flow Skill Series

Session 1

IEEE UCF

UCF College of Engineering and Computer Science

Ice Breaker!

Epic

About you!

- Name
- Major
- Year
- Why you came to the workshop
- What you hope to take away from the workshop

Agenda & Helpful knowledge

Session 1:

- Learn about RISC-V ISA
- Learn about the single cycle datapath
- Layout the design plan for pieces of our processor
- Make modules of necessary hardware functions

Need to know:

- Verilog
- Vivado
- Basic knowledge of ISAs

What is RISC & RISC-V?

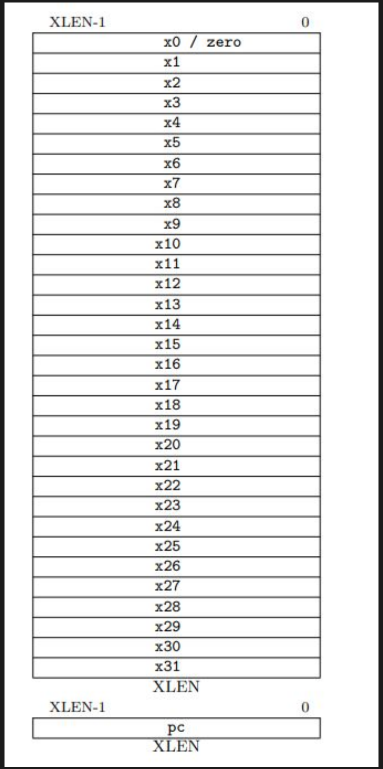
RISC stands for **reduced instruction set computer** which allows for more simple computer architecture design in regards to instructions.

RISC-V is an open source instruction set architecture (ISA), that has specific standards in regards to how the ISA functions.

RISC-V Register & Instruction Format

In RISC-V there are 32 integer registers that you as the hardware design architect have access to to use.

The names of these registers range from x0-x31 where x0 is always set to 0.



XLEN-1 0

x0 / zero
x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
x11
x12
x13
x14
x15
x16
x17
x18
x19
x20
x21
x22
x23
x24
x25
x26
x27
x28
x29
x30
x31

XLEN

XLEN-1 0

pc

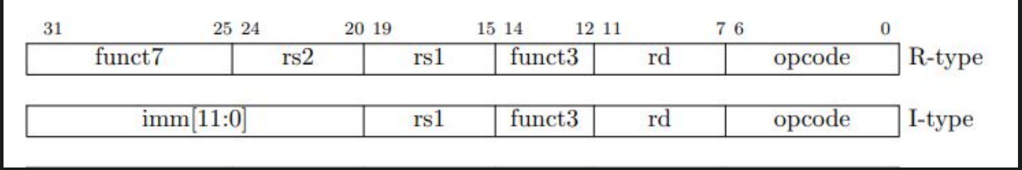
XLEN

RISC-V Register & Instruction Format

RISC-V has different instruction format types but the ones we will be using are the R-type and I-type instructions which are 32-bits

The fields being used are as follows:

Funct7, funct3, rs1, rs2, rd, opcode, and immediate



31 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11:0]		rs1	funct3	rd	opcode	I-type

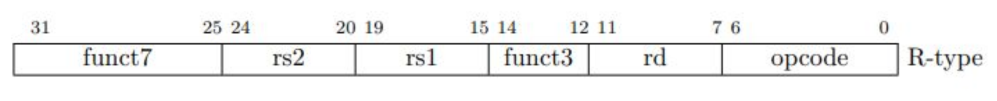
R-Type Instructions

R-Type instruction are instructions that change or manipulate registers by using only registers in the operation.

For example:

Add x1, x2, x3

To right shows how to setup the bit-fields for R-type in RISC-V



31 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type

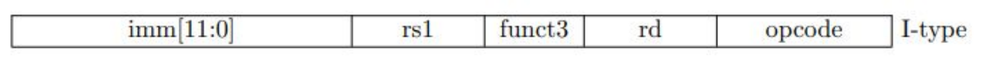
I-Type Instructions

I-Type instructions are similar to R-Type by changing and manipulating registers but I can use 32-bit integers values and registers to perform the operation.

For example:

Add x1, x2, 10

To right shows how to setup the bit-fields for I-type in RISC-V



imm[11:0]	rs1	funct3	rd	opcode	I-type

RISC-V Instruction to Machine Code

Let's take the instruction: **ADDI x1, x1, 10**

We know that this is an I-Type instruction so how do we get the bit-fields from this instruction?

We will need: Imm, rs1, rd, Funct3,and opcode

Let’s do it together!

Answer: **0X00A08093**

RISC-V Instruction to Machine Code

Let's take the instruction: **ADD x3, x2, x1**

We know that this is an I-Type instruction so how do we get the bit-fields from this instruction?

We will need: Funct7, Funct3, rs1, rs2, rd, and opcode

Let’s do it together!

Answer: **0X001101B3**

Single Cycle Datapath

How does the hardware perform these instructions?

Introducing the single cycle datapath! This is a hardware model of one instruction can be performed in a single clock cycle.

Although this isn’t what usually happens industry, it's a great way to understand how instructions are interpreted into hardware.

A pipelined architecture is what is best for for general processing.

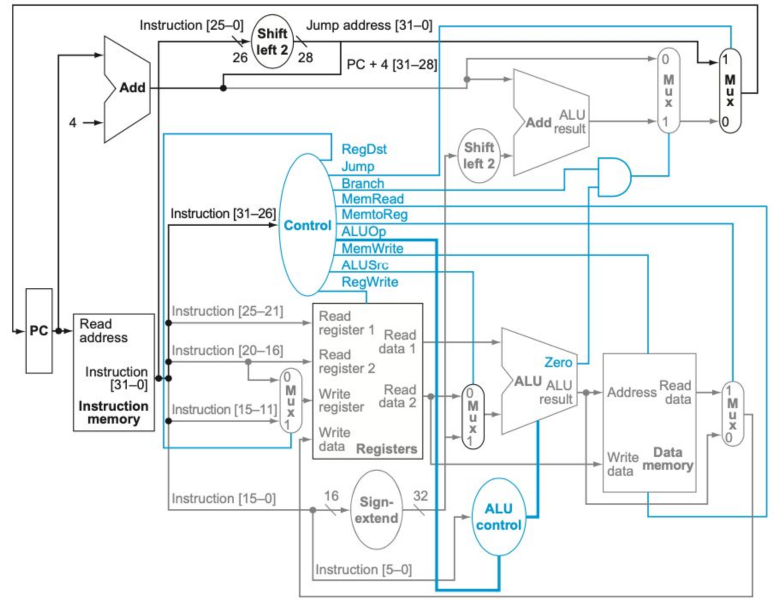
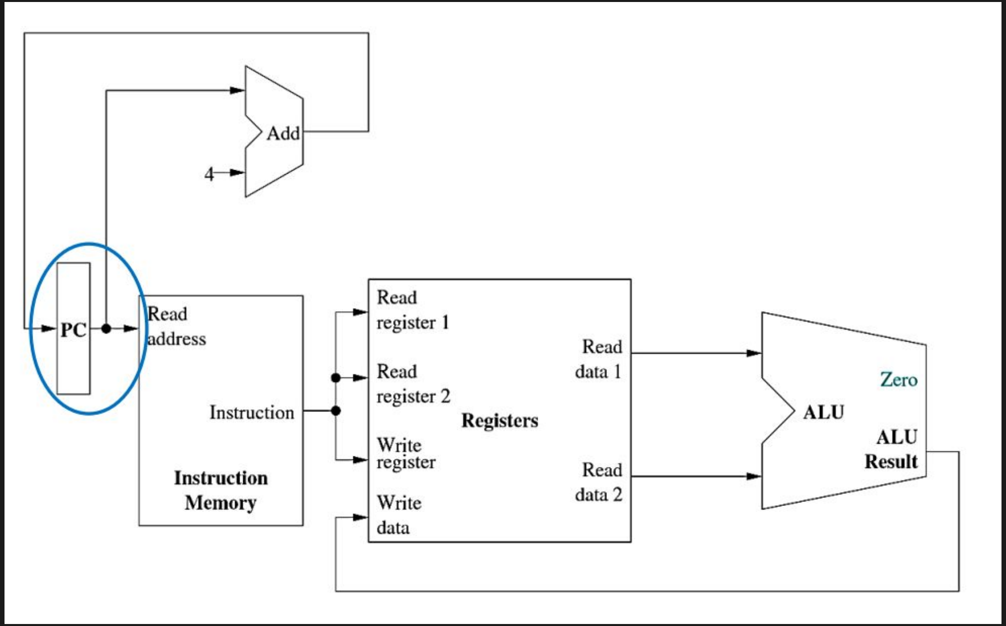


FIGURE 4.24 The simple control and datapath are extended to handle the jump instruction. An additional multiplexor (at the upper right) is used to choose between the jump target and either the branch target or the sequential instruction following this one. This multiplexor is controlled by the jump control signal. The jump target address is obtained by shifting the lower 26 bits of the jump instruction left 2 bits, effectively adding 00 as the low-order bits, and then concatenating the upper 4 bits of PC + 4 as the high-order bits, thus yielding a 32-bit address.

RISC-V ALU Single Cycle Datapath Design

Our goal is to design a RISC-V based computer architecture that will be able to support R-Type and I-Type math based instructions.

This means we will need to build instruction memory, register file module, and an ALU module.



Verilog & Vivado Module Setup

In this session we will only setup the the modules we need to perform and verify our RISC-V ALU architecture works.

We will need 4 design modules and 3 testbench module

Design Modules:

alu_top.v

instruction_mem.sv

register_select.v

risc_top.v

Testbench Modules:

risc_tb.v

reg_select_tb.v

alu_tb.v

Follow Along!

Epically