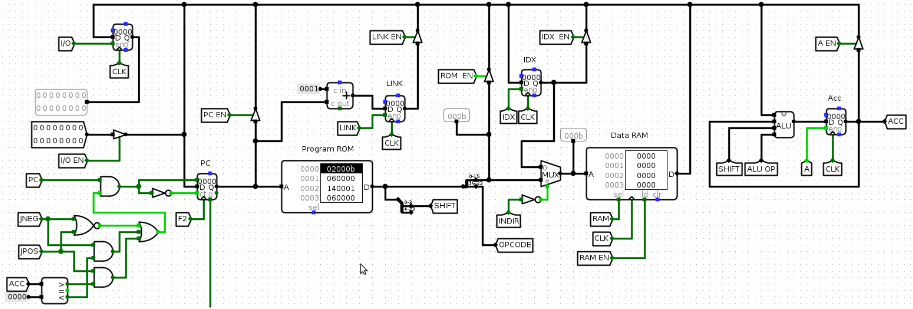
Pipelines

Saulius Gražulis

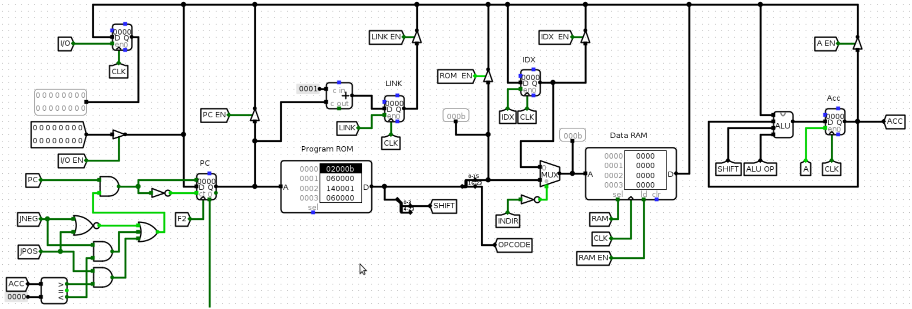
Vilnius, 2023

Vilnius University, Faculty of Mathematics and Informatics
Institute of Informatics

Single cycle CPU



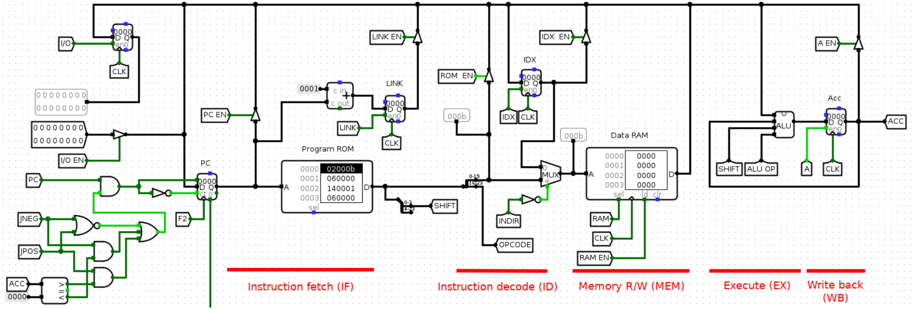
Single cycle CPU



Propagation delay:

$$t_{\text{total}} = t_{\text{PC}} + t_{\text{ROM}} + t_{\text{decoder}} + t_{\text{RAM}} + t_{\text{ALU}} + t_{\text{Acc}}$$

Single cycle CPU



Instruction fetch (IF)

Instruction decode (ID)

Memory R/W (MEM)

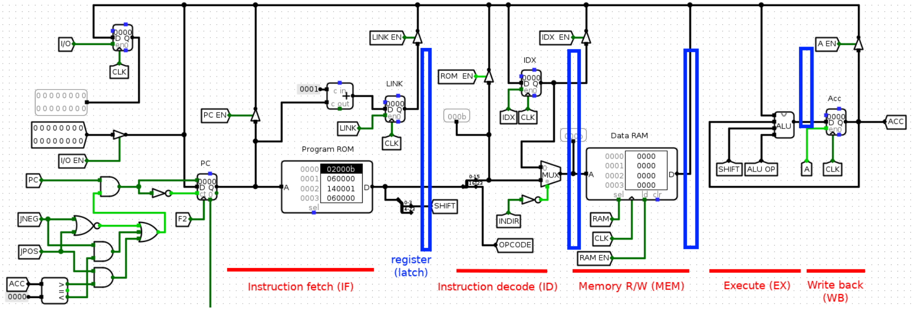
Execute (EX)

Write back (WB)

Propagation delay:

$$t_{\text{total}} = t_{\text{PC}} + t_{\text{ROM}} + t_{\text{decoder}} + t_{\text{RAM}} + t_{\text{ALU}} + t_{\text{Acc}}$$

Single cycle CPU



register (latch)

Instruction fetch (IF)

Instruction decode (ID)

Memory R/W (MEM)

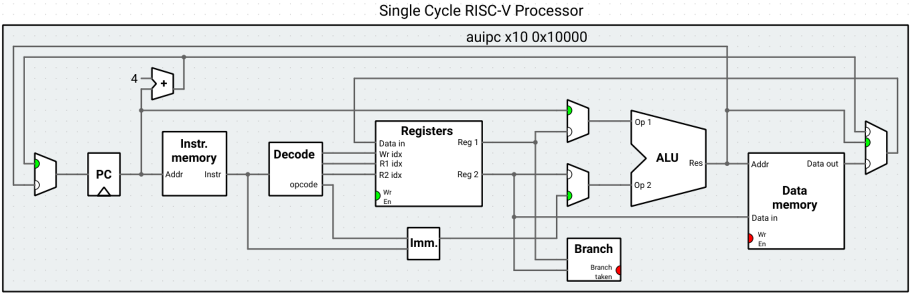
Execute (EX)

Write back (WB)

Propagation delay:

$$t_{\text{total}} = \max(t_{\text{PC}} + t_{\text{ROM}}, t_{\text{decoder}}, t_{\text{RAM}}, t_{\text{ALU}}, t_{\text{Acc}}) + t_{register}$$

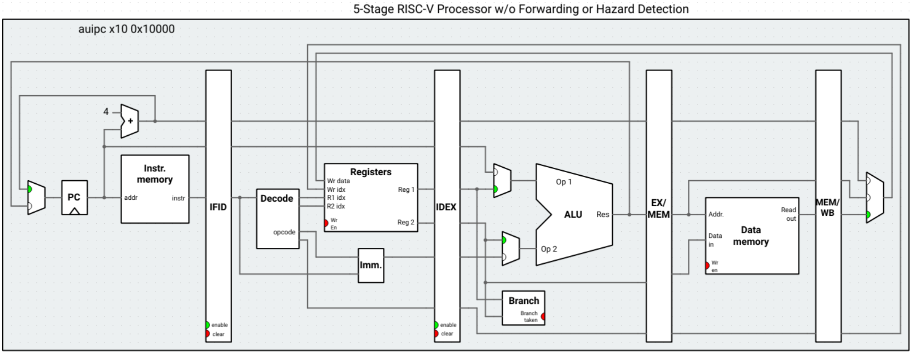
Ripes RISC-V single cycle



M. B. Petersen, The Ripes simulator

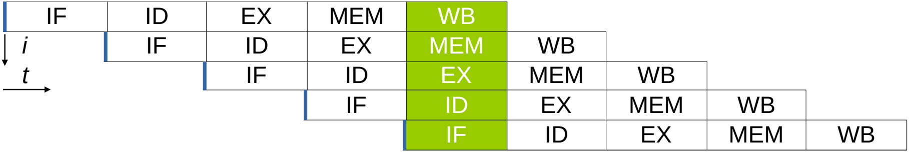
Ripes RISC-V 5 stage

w/o forwarding or hazard detection



M. B. Petersen, The Ripes simulator

Classic 5-stage pipeline



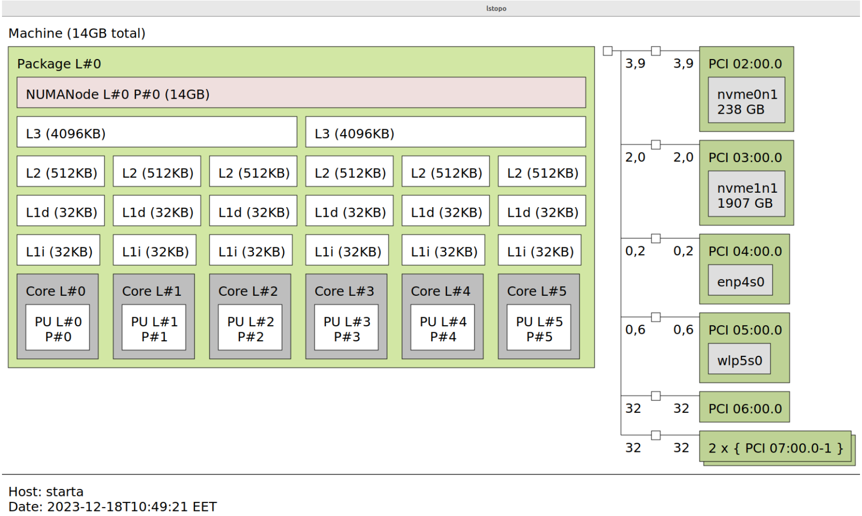
IF	ID	EX	MEM	WB				
	IF	ID	EX	MEM	WB			
		IF	ID	EX	MEM	WB		
			IF	ID	EX	MEM	WB	
				IF	ID	EX	MEM	WB

Sandstorm de, CC BY-SA 4.0 via Wikimedia Commons

- RISC-V: example cores with 5 stages;
- ARM: ARM11 (Raspberry Pi) – 8-stage instruction pipeline (Upton 2016);
- Intel: many current CPUs have 20 stages or more (Upton 2016);

Architecture of a really existing CPU

Command `lstopo`



lstopo

Machine (14GB total)

Package L#0

NUMANode L#0 P#0 (14GB)

L3 (4096KB) L3 (4096KB)

L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB) L2 (512KB)

L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB) L1d (32KB)

L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB) L1i (32KB)

Core L#0 PU L#0 P#0

Core L#1 PU L#1 P#1

Core L#2 PU L#2 P#2

Core L#3 PU L#3 P#3

Core L#4 PU L#4 P#4

Core L#5 PU L#5 P#5

3,9 3,9 PCI 02:00.0 nvme0n1 238 GB

2,0 2,0 PCI 03:00.0 nvme1n1 1907 GB

0,2 0,2 PCI 04:00.0 enp4s0

0,6 0,6 PCI 05:00.0 wlp5s0

32 32 PCI 06:00.0

32 32 2 x { PCI 07:00.0-1 }

Host: starta
Date: 2023-12-18T10:49:21 EET

Hazards

- Control hazards: Caused by conditional branch instruction
- Data hazards: Caused by data dependency between instructions
- Structural hazards: Caused by resource conflicts

Data hazards

```
add x5,x1,x2
add x7,x5,x1
```

- Stages:

add **x5**,x1,x2	Memory (MEM)
add x7,**x5**,x1	Execute (EX)

- x5 is not yet ready for the second instruction
- Solutions:
  - Stall the pipeline/insert “bubbles”
  - Reorder instructions (in software)

```
add x5,x1,x2
lw  x20,(x11)
add x7,x5,x1
```

  - Reorder instructions (in hardware)
  - Fast-forward the data

Control hazards

```
add x5,x1,x2
bgt x1,x2,label
add x7,x5,x1
```

- Stages:

add **x5**,x1,x2	Memory (WB)
bgt x1,x2,label	Execute (MEM)
add x7,**x5**,x1	Decode (EX)

- x5 The next add must not be executed!
- Solutions:
  - Reset the pipeline/insert “bubbles”;
  - Execute add anyway (delay slot);
  - Speculative execution;

Convergence of CISC and RISC

- Large number of registers (≥ 16);
- Orthogonal instruction sets;
- Load/Store operation
- Pipelines
- Instruction and data caches
- Modified Harvard architecture

References I

Upton, Eben (Jan. 2016). *Learning Computer Architecture with Raspberry Pi*. Wiley. ISBN: 978-11-1918-393-8.