Name: 洪 理 川
ID : B113040056

4.6 ~ 4.11 Preview Report

4.6 An Overview of Pipelining

A. Definition

Pipelining is an implementation technique in which multiple instructions are overlapped in execution, much like an assembly line

B. Approach

in laundry

Non-pipelined

1. Put one dirty load in the washer
2. When water is finished, place it to the dryer
3. When dryer is finished, place it on the table and fold
4. When folding is finished, ask roomate to put the clothes away

pipelined Approach

↳ the steps is similar as non-pipelined. But when we get to second step, we put another dirty load in step 1, and go on for steps 3 and 4.

Pipelining → working in parallel, its improves the throughput of the laundry system.

C. Pipeline in Instruction execution

5 steps:

1. Fetch instruction from memory
2. Read registers and decode the instruction.
3. Execute the operation or calculate an address
4. Access an operand in data memory
5. Write the result into a register

Formula of pipelining

$$\text{Time between instructions}_{\text{pipelined}} = \frac{\text{Time between instructions}_{\text{nonpipelined}}}{\text{Number of pipe stages}}$$

D. Pipeline Hazards

↳ condition in pipelining when next instruction can't execute in the next cycle.

3 types → Structural Hazard, Data Hazards, Control Hazard,

1. Structural Hazards

↳ hardware can't execute combination of instructions in the same clock cycle.

Ex: washer and dryer in the same machine.

2. Data Hazards

↳ Pipeline stalled because 1 step must wait for another to complete.

Ex : (x19), x0, x2 sub must wait for x19 add, causing hazard
x2, (x19), x3

3. Control Hazards

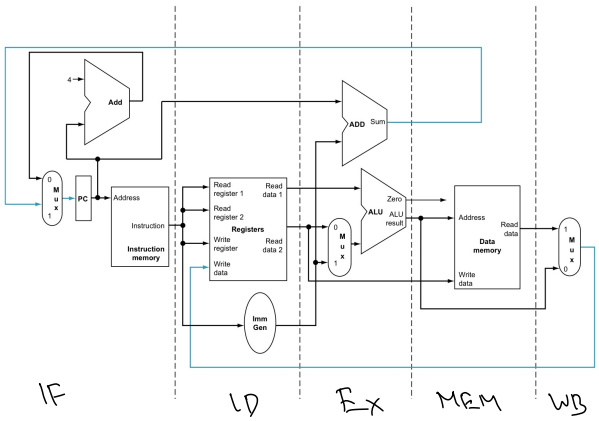
↳ need to make decision on the results of one instructions while others are executing

Computer use prediction to handle this hazards. A popular approach is to keep of history for each branch we have taken.

A.7 Pipelined Datapath and Control

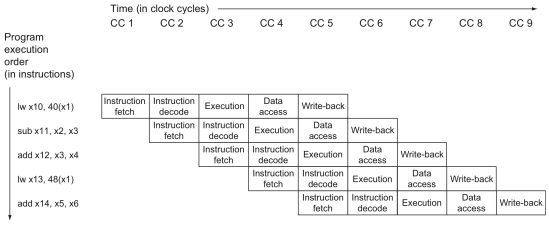
Five-stage pipeline

1. IF : Instruction fetch
2. ID : Instruction decode and register file read
3. EX : Execution or address calculation
4. MEM : Data memory access
5. WB : Write back

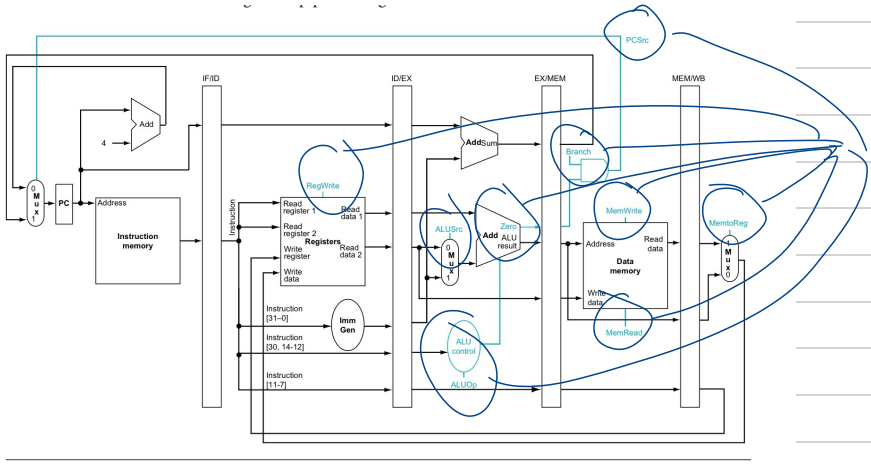


In this Datapath, All stages uses lw instructions and store instructions.

We can graphically draw the multiple-clock-cycle →



Pipeline Control



Control Unit in Pipeline

- Execution/address calculation : The signal set by ALUOp and ALUSrc
- Memory Access : The control lines set in this stage are Branch, MemRead, and MemWrite.
- Write-back : The 2 control lines are MemtoReg

4.8 Data Hazards : Forwarding versus Stalling

Forwarding

↳ Simply forward the data as soon as it is available to any units

Notation of fields in pipeline registers

ID/EX . RegisterRs1

- numbers of 1 registers whose value is found in the pipeline register ID/EX :

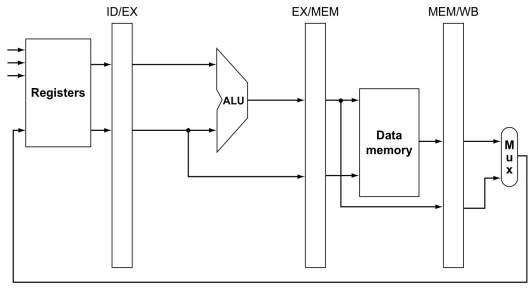
ID/EX → name of pipeline register

RegisterRs1 → name of the field in the register.

Problem : Dependences between the pipeline registers and the inputs to the ALU

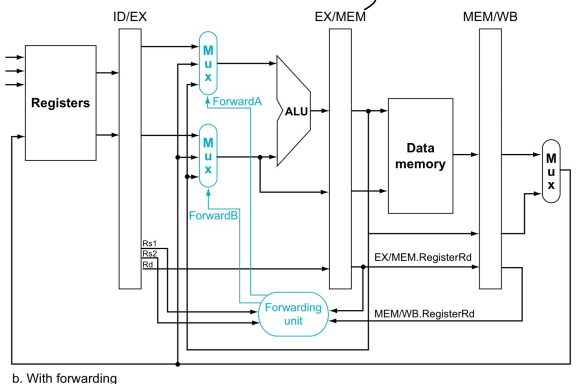
Solution : Take the inputs to the ALU from any pipeline register rather than just ID/EX, then we can forward the correct data

No forwarding



a. No forwarding

With Forwarding



b. With forwarding

Stalls

Problem that Forwarding can't solve: When instruction tries to read a register following a load instruction that writes the same register.

So, we need a FORWARDING + HAZARD DETECTION UNIT.

code for hazard detection unit

if (ID/EX.MemRead and ((ID/EX.RegisterRd = IF/ID.RegisterRs1) or (ID/EX.RegisterRd = IF/ID.RegisterRs2)))

Stall the pipeline.

4.9 Control Hazards

2 schemes & 1 Optimization

→ Assume Branch Not Taken

We want to predict the conditional branch to improve the performance.

Things need to change to make this prediction:

1. Change IF, ID and EX stages when the branch reaches the MEM stage
2. Change control to 0 in the ID stage

→ Reducing the delay of Branches

↳ reduce the cost of the taken branch

We can do it by moving the conditional branch execution to the ID stage. It will reduces the branch to only one instruction. We also need to flush instructions in the IF stage, by introducing a new control line, called IF.Flush.

→ Dynamic Branch Prediction

↳ look up the address of the instruction to see if the conditional branch was taken the last time this instruction was executed, if yes, begin fetch new instructions from the same place as the last time.

Implementation: Branch history

↳ use some kind of buffer

- Prediction right → may put there by another conditional branch that has the same low-order address bit.
- Prediction wrong → wrong predictions deleted, the prediction bit is inverted and stored back, proper sequence is fetched and executed.

4.10 Exceptions

↳ any unexpected change in control from either internal / external.

Type of event	From where?	RISC-V terminology
System reset	External	Exception
I/O device request	External	Interrupt
Invoke the operating system from user program	Internal	Exception
Using an undefined instruction	Internal	Exception
Hardware malfunctions	Either	Either

How Exceptions are Handled

Steps:

1. Save the address of the unfortunate instruction in the supervisor exception cause register (SEPC)
2. The Operating System continue the action, providing some service to user.

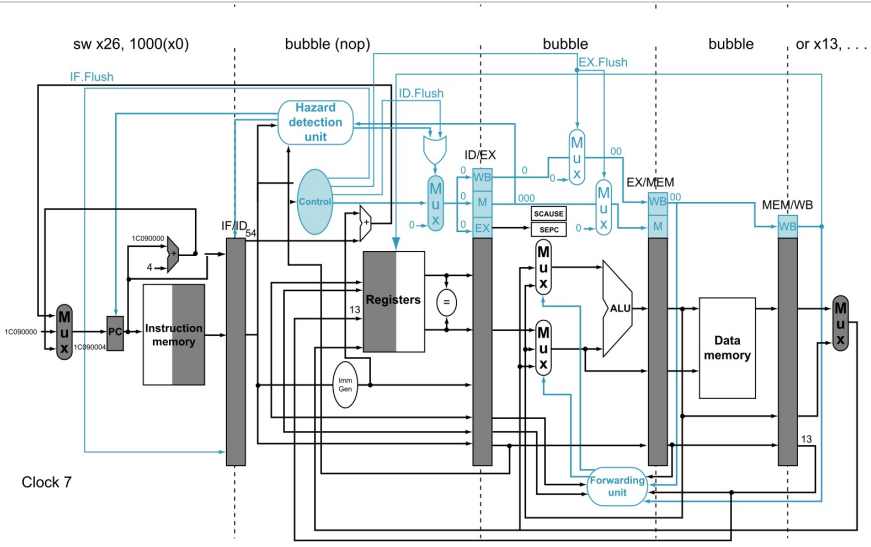
3. Based on the answer, the OS terminate the program or continue execution

Exceptions in a Pipeline Implementation

The easist way: flush the instruction and restart it from the beginning.

Deal with branch misprediction: use the multiplexor already in the ID stage that zeros control signals for stalls.

Final step -> save the address of the offending instruction in the supervisor exception program counter.



Result of Exception with control lines -> Hazard detection unit, IF.Flush, ID.Flush, etc.

4.11 Parallelism via Instructions

ILP - Instruction-level-parallelism

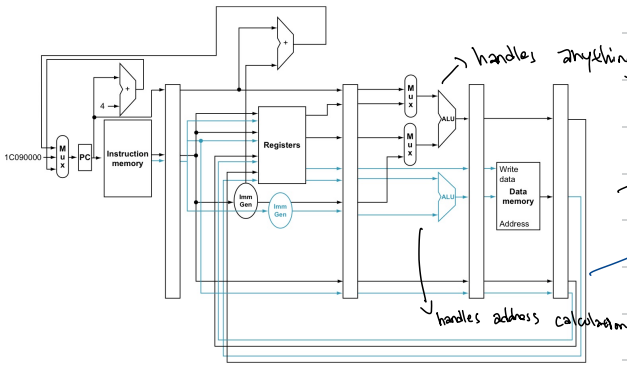
↳ Pipelining exploits the nature potensial of parallelism

2 Methods to improve

1. Increase the depth of the pipeline to overlap more instructions ——→ Make more steps
2. Replicate the internal components of the computer ——→ From 1 washer & Dryer

↓

→ washer & Dryer



→ 2-issue datapath

→ The blue color indicate the other issue datapath