Dylan Thornbury

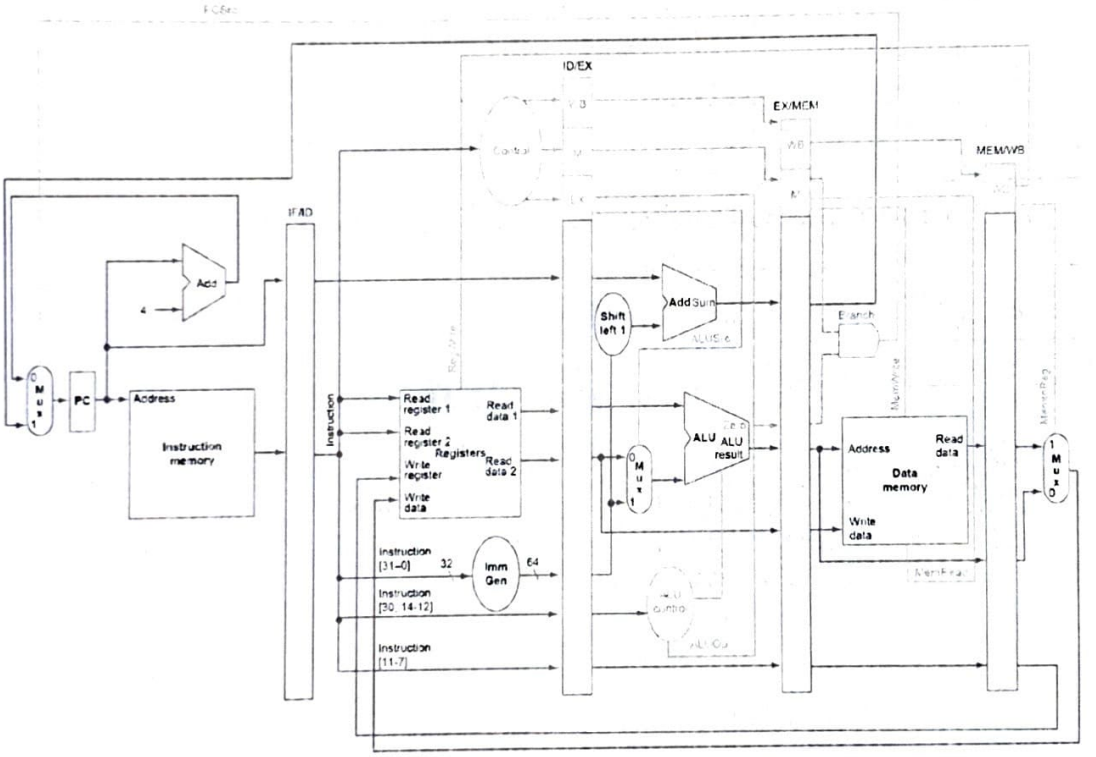
SANTA CLARA UNIVERSITY	ECEN 122 Winter 2024	Hoeseok Yang

Homework #6: RISC-V datapath

Answer the following problems and upload your answers to Camino in pdf format.

Posted: February 22, 2024. Due by: 9pm on February 28, 2024

1. [1.2 pts] Below is the simplified pipelined RISC-V control and datapath:



Suppose that the logic blocks used to implement the above datapath have the following latencies:

Instr./Data Memory	Register File	MUX	ALU	Add	Imm Gen	Control	ALU Control	Shift
250 ps	150 ps[1]	25 ps	230 ps	150 ps	50 ps	50 ps	50 ps	0 ps[2]

It takes 30 ps (from the rising clock edge) to read data from registers, such as PC, IF/ID, ID/EX, and so on. We refer to this delay as "register read delay". Likewise, "register setup delay" refers to the amount of time a register's data input must be stable before the next rising edge of the clock. In this case, we have

- Register read delay: 30 ps
- Register setup delay: 20 ps

[1] We assume that the latency of "Register File" includes register read delay.
[2] Shift left 2 can be easily implemented by rewiring signals, thus no propagation delay applies here.

Name:_____ Student ID:_____

A. (0.3 pts) We try to determine the minimum time required for each of the five pipeline stages. For example, in the IF stage, it takes 30 ps (register read delay) for reading the current PC value, then it takes 250 ps (Instr. Memory latency) for reading an instruction from the instruction memory. Then, we have to consider the register setup delay (20 ps) as the result (instruction) should be written to IF/ID. So, the minimum time required for IF is 30 ps + 250 ps + 20 ps.[3] Complete the following table by filling in the minimum time required for each pipeline stage.

1. Instruction Fetch (IF)	2. Instruction Decode (ID)	3. Execution (EX)	4. Memory (MEM)	5. Write Back (WB)
300 ps	200 ps	330 ps	300 ps	225 ps

B. (0.3 pts) Based on the previous answer, calculate the theoretical maximum clock frequency that this datapath may have.

slowest is 330 ps : 3,030,303,030 ≈ 3ghz

only one stage done per clock cycle

$\frac{1}{330\text{ ps}}$

C. (0.3 pts) Which of the following is the most efficient way to improve the throughput of this datapath? Choose the most efficient option and explain why you think so.

A. To use a faster instruction memory

B. To use a faster ALU (in EX)

C. To use a faster data memory

D. To reduce the latency of the Control logic

It's the ALU as the Ex step is the longest step. It forces all other steps to be that slow,

D. (0.3 pts) We didn't consider the data hazard issue in the above datapth. Suppose that we are adding a forwarding unit to this datapath to handle data hazards. To which pipeline stage do we need to add this unit?

we need to add it at execute, output of EX_n connected to input of EX_{n+1}

[3] You may wonder why the delay of the MUX (in front of the PC) is not considered. Note that this MUX is used for choosing the "next" PC value (to be stored in the next cycle). For fetching the "current" instruction, you don't need to wait for the completion of this MUX.

2. [1.2 pts] Assume that we added the forwarding unit to the pipeline shown in Problem #1 to handle data hazards. Consider the fragment of RISC-V assembly code below:

```
...
lw      x1, 0(x0)      # instr. 1
lw      x2, 4(x0)      # instr. 2   } raw hazard
add     x3, x1, x2     # instr. 3
addi    x4, x0, 12     # instr. 4   } raw Hazard
lw      x5, 0(x4)      # instr. 5   } RAW Hazard
add     x6, x4, x5     # instr. 6   } RAW Hazard
sw      x6, 4(x4)      # instr. 7
...
```

stall needed →

stall needed →

A. (0.4 pts) Identify all data hazards that exist in the given assembly code.

Only data hazards (see above)

B. (0.4 pts) Even with forwarding, we sometimes need to add stalls between instructions. How many stalls (nops) do we need to insert to the given assembly code (even with forwarding) in order to avoid causality issues?

2; after every lw that has a hazard

C. (0.4 pts) Minimize the number of stalls by reordering the instructions of the given RISC-V assembly code.

one stall

```
lw x1, 0(x0)
lw x2, 4(x0)
addi x4, x0, 12
add x3, x1, x2
lw x5, 0(x4)
sw x6, 4(x4)
add x6, x4, x5
```

```
addi x4, x0, 12
lw x1, 0(x0)
lw x2, 4(x0)
lw x5, 0(x4)
add x3, x1, x2
add x6, x4, x5
sw x6, 4(x4)
```

famsuer

Name: _____ Student ID: _____

3. [0.6 pts] Read the Wikipedia article (https://en.wikipedia.org/wiki/Branch_predictor) on static branch prediction and briefly explain the difference between the static branch predictors used in early versions of SPARC and MIPS and advanced static branch predictors used in the RISC-V architecture.

Basic Static Branch Predictors

SPARC MIPS

- single direction prediction
- always predict a cond. jump won't be taken
- always fetch next instr
- 2 cycles long

Advanced Static Branch Predictors

RISC-V

- assumes backward branching
- typically faster b/c of how prevalent loops are
- skips forward branching

Name:_____ Student ID:_____