

Name: ANKIT MONDAL

Roll No: 2021CS10229

(COL 216) Computer Architecture

March 26, 2023

Minor 2

Duration: 60 minutes

(30 marks)

**Beware:** Be concise in your writing. You can use rough sheets for calculations. But you cannot submit any additional sheet for grading on Gradescope. So make sure you are certain when you write something (after rough work, or use a dark pencil). If you cheat, you will surely get an F in this course.

1. A 5-stage MIPS pipeline has a 1 cycle **load-to-use** delay i.e. you have to wait for 1 cycle between a *lw* instruction and any ALU instruction that uses it, as you cannot forward from MEM where data is ready at the end of cycle to EX where data is needed at the beginning of cycle. The pipeline is otherwise ideal i.e. the pipeline is always full, there is no forwarding delay, and each instruction takes 1 clock cycle. A program has 20% loads, but the compiler can find independent instructions to put after the load only for half of them. What is the slowdown due to delay/NOP slots? [3 marks]

We assume that when we have a dependent inst. after load it is an ALU inst.  
Let there be  $x$  instructions. Thus  $0.2x$  of them are branch inst.  $0.1x$  of them have a dependent inst. after them.  
Each such case will stall for 1 cycle. Thus we will need  $0.1x$  extra cycles. Since each inst. completes in 1 cycle, ~~see~~  $x$  inst. will take  $x$  cycles. Thus time taken =  $x + 0.1x$  cycles =  $1.1x$  cycles.  
Expected no. of cycles =  $x$  cycles.  
Slowdown =  $\frac{1.1x}{x} = 1.1$   
Slowdown % =  $\frac{1.1-1}{1} \times 100\% = 10\%$

Name: ANKIT MONDAL

Roll No: 2021CS10229

2. Out of the 5 MIPS pipeline stages IF, ID, EX, MEM, W — which are never destinations for forwarding/bypassing? Why? [3 marks]

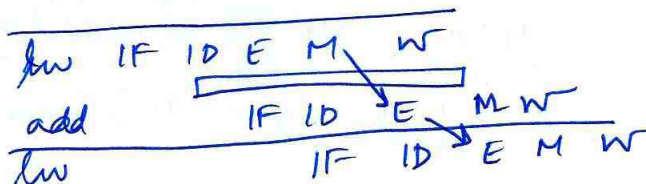
IF ~~and~~, ID ~~and~~ and W stages are never destinations for forwarding. This is because, IF and ID stages are required to interpret the current instruction and fetch corresponding data from reg. memory. There is no latch before IF, and the latch before ID stores the data corresponding to current inst. Since the nature of inst. cannot be modified by data hazards from previous instructions, we don't need forwarding here. Similarly W takes information evaluated from current instruction, forwarding here from previous inst. is equivalent to not running current instruction which won't happen in data hazards.

3. Consider a 1000 instruction program of alternating *lw* and *add* instructions: *lw*, *add*, *lw*, *add* etc. The *add* instruction depends (and only depends) on the *lw* instruction immediately before it. The *lw* instruction depends (and only depends) on the *add* instruction immediately before it. Calculate CPI on MIPS 5-stage pipeline datapath with and without forwarding. [2+2 = 4 marks]

*lw* ↘ loaded data  
*add* ↗ will be needed in *add*

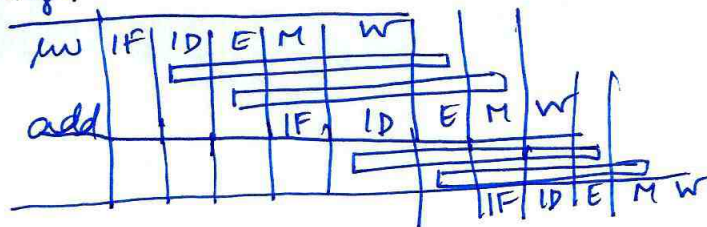
*add* ↘ data comp. by *add*  
*lw* ↗ load ← needed for *load*.

With forwarding.



For each *lw* one extra cycle needed. So approx 1500 cycles needed for 1000 inst.  $\Rightarrow$  CPI = 1.5.

Without forwarding.



2 nop after each inst. so, approx 3000 cycles, (1000 for inst. 2000 for nop) needed for 1000 inst.  $\Rightarrow$  CPI =  $\frac{3000}{1000} = 3$ .

Name: ANKIT MONDAL

Roll No: 2021CS10223

4. In a single cycle processor, all instructions use a common clock. The table shows the time needed by three instruction types, and their respective percentage in a set of instructions. What is the fastest clock cycle this single cycle processor can use? What percentage of time is wasted in executing 10 instructions? [3 marks]

Instruction	Time	% of instructions
Type 1	800 ps	30%
Type 2	200 ps	20%
Type 3	250 ps	50%

Table 1: Instruction details

Since all instructions use a common clock and finish in a single cycle, the fastest clock cycle is limited by the ~~total~~ time taken by the longest inst. Thus the fastest clock cycle can be 800 ps.

In 10 inst., 3 will be Type 1, 2 of Type 2, 5 of Type 3. Each inst. of Type 2 wastes  $800 - 200 = 600$  ps, Type 3 wastes  $800 - 250 = 550$  ps. Thus time wasted by 2 cycles of Type 2 =  $2 \times 600$  ps and 5 Type 3 =  $5 \times 550 = 2750$  ps. Total time wasted = 3950 ps. Total time taken =  $800 \times 10 = 8000$  ps. Time wasted % =  $\frac{3950}{8000} \times 100\% = 49.375\%$ .

5. A processor takes 100ns and 100pJ for every instruction. It can be infinitely pipelined with each pipeline register taking 2ns and 2pJ. What is the throughput, latency per instruction and energy per instruction for a 100 stage processor, compared to the original processor? [3 marks]

If we take a 100 stage processor assume that each stage has been divided equally, then each stage will take 1 ns. Assuming we add register to every stage, time taken per pipeline stage =  $1 \text{ ns} + 2 \text{ ns} = 3 \text{ ns}$ .

Latency = no. of pipeline stages  $\times$  time per pipeline stage =  $100 \times 3 = 300 \text{ ns}$

Throughput =  $\frac{1}{\text{time of 1 inst.}} = \frac{1}{\text{time for 1 pip. stage}} = \frac{1}{3 \times 10^{-9} \text{ s}^{-1}} = 3.33 \times 10^8 \text{ s}^{-1}$

If 1 reg. is added to every stage, we have 100 new regs and each takes up 2pJ energy. Total extra energy =  $100 \times 2 \text{ pJ} = 200 \text{ pJ}$ .

Net energy consumed =  $100 \text{ pJ} + 200 \text{ pJ} = 300 \text{ pJ}$ . Ans

(If ~~the~~ last stage already has reg, power =  $100 + 99 \times 2 = 298 \text{ pJ}$ )

Name: ANKIT MONDAL

Roll No: 2021CS10223

6. The 5 stages of a processor have the following latencies.

Fetch	Decode	Execute	Memory	Writeback
300ps	400ps	350ps	500ps	100ps

Table 2: Processor latencies

Assume that when pipelining, each pipeline stage costs 20ps extra for the registers between pipeline stages. If you could split one of the pipeline stages into 2 equal halves, which one would you choose? Do you see any improvement in cycle time, latency for one instruction and throughput, with this split? [4 marks]

If we could split one stage into 2 it will be the memory stage. This is because currently memory has largest latency of 500 ps, which limits clock cycle to ~~520~~ 520 ps. If we split this into 2 eq. halves, Each stage will take  $250 + 20 = 270$  ps. Then limiting stage will be decode with 420 ps time.

Fetch	Decode	Execute	Mem1	Mem2	WB
320 ps	420 ps	370 ps	270 ps	270 ps	<del>100</del> 120 ps

(assuming WB register is already present)  
(we are adding 20 ps to every stage in pipeline as mentioned in ques)

$$\text{Latency} = \text{cycle time} \times \text{no. of stages}$$

$$= 420 \text{ ps} \times 6$$

$$= 2520 \text{ ps.}$$

$$\text{Initial latency} = 520 \times 5$$

$$= 2600 \text{ ps.}$$

$$\text{Throughput}_{\text{now}} = \frac{1}{\text{cycle time}} = \frac{1}{420 \times 10^{-12}} = \frac{10^{12}}{420}$$

$$= 2.38 \times 10^9$$

$$\text{Throughput}_{\text{old}} = \frac{1}{\text{cycle time}} = \frac{1}{520 \times 10^{-12}} = 1.92 \times 10^9.$$

Thus throughput increases and latency decreases with the split. Thus both throughput and latency improve.

Name: ANKIT MONDAL

Roll No: 2021CS10229

7. A 1-bit saturation counter toggles between 0,1 states and a 2-bit counter toggles between 00,01,10,11 states. We build two branch predictors, one with 1-bit and the other with 2-bit counters. We have two traces of branch taken (T) and not-taken (N): Trace1: NNNTNNN, Trace2: NTNTNTN. What will be the accuracies for each trace for each predictor? Assume each counter starts at strongly not taken state. **5 marks.**

1 bit.

Trace 1:

Predicted state	Actual state	next state	
N	N	N	✓
N	N	N	✓
N	N	N	✓
N	T	T	✗
T	N	N	✗
N	N	N	✓
N	N	N	✓

$$\text{Accuracy} = \frac{5}{7} = 71.43\%$$

Trace 2

Pred.	Act.	Next	
N	N	N	✓
N	T	T	✗
T	N	N	✗
N	T	T	✗
T	N	N	✗
N	T	T	✗
T	N	N	✗

$$\text{Accuracy} = \frac{1}{7} = 14.29\%$$

2 bit's

Trace 1:

state	Pred.	Act.	Next	
00	N	N	00	✓
00	N	N	00	✓
00	N	N	00	✓
00	N	T	01	✗
01	N	N	00	✓
00	N	N	00	✓
00	N	N	00	✓

$$\text{Accuracy} = \frac{6}{7} = 85.71\%$$

Trace 2

state	Pred	Act.	Next	
00	N	N	00	✓
00	N	T	01	✗
01	N	N	00	✓
00	N	T	01	✗
01	N	N	00	✓
00	N	T	01	✗
01	N	N	00	✓

$$\text{Accuracy} = \frac{4}{7} = 57.14\%$$

Name: Ankit Mondal

Roll No: 2021CS10229

8. What operations can the following ALU perform? Please label the diagram to describe which operation each input, gate and output in the ALU is related to? [5 marks]

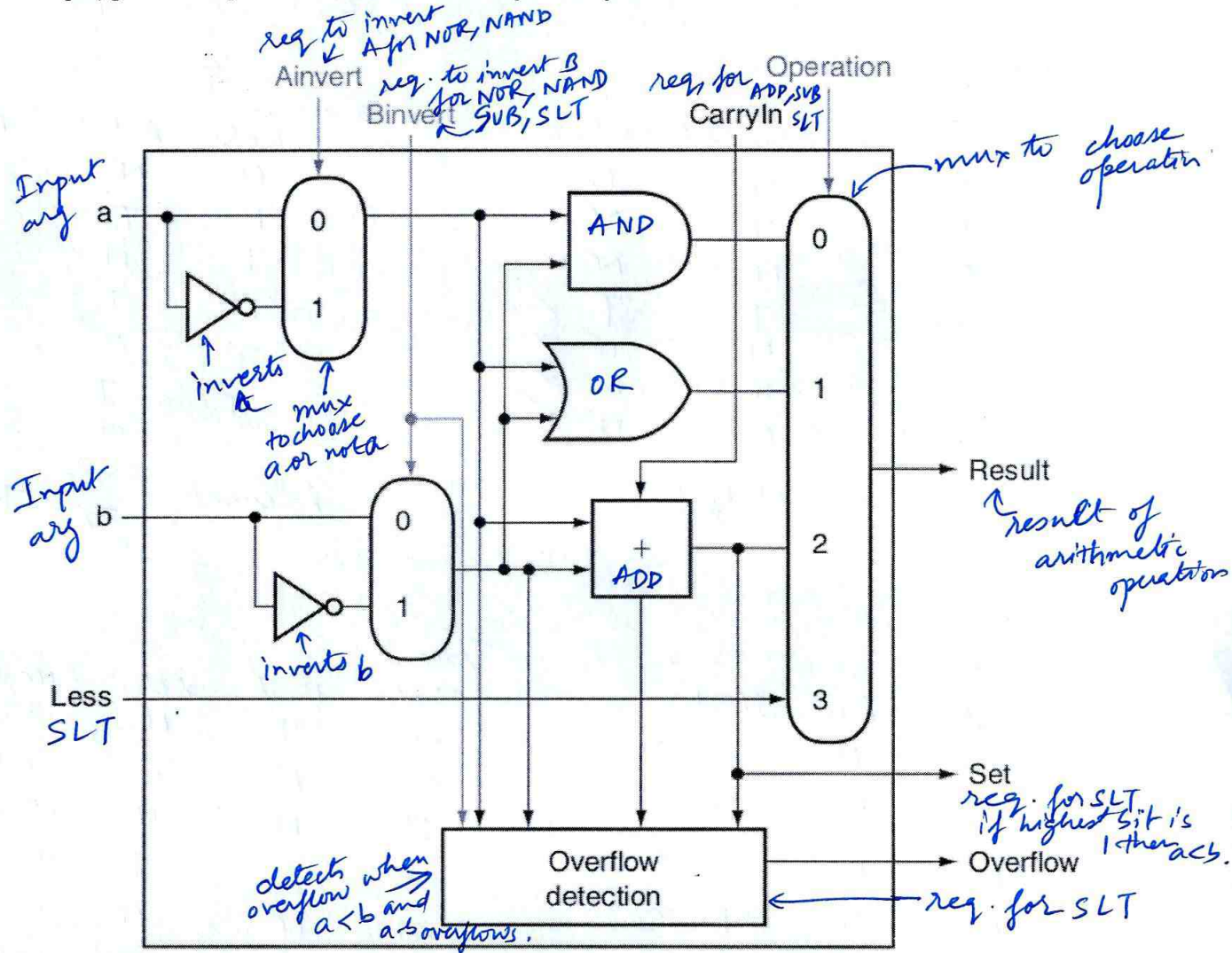


Figure 1: ALU design for arithmetic and logical operations.

Operations : \*

AND

OR

ADD.

NAND

NOR.

SUB

SLT