113 計算機組織 Homework 4-2 Due date: YYYY/MM/DD

- You must write down / type the calculations, or you won't get the scores.
- 請用手寫於 A4 紙上,並掃描上傳至 I 學園
- 請勿抄襲，抄襲者與被抄襲者單一分數取平均

一、(20%) How pipelining affects the clock cycle time of the processor. Assume that individual stages of the datapath have the following latencies:

IF	ID	EX	MEM	WB
400ps	300ps	500ps	350ps	450ps

1.1. (6%) What is the clock cycle time in a pipelined and non-pipelined processor?

1.2. (7%) What is the total latency of an LW instruction in a pipelined and non-pipelined processor?

1.3. (7%) If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

二、(20%) To understand the cost/complexity/performance trade-offs of forwarding in a pipelined processor. Refer to pipelined datapaths from Figure 4.45, assume that, all the instructions executed in a processor, the following fraction of these instructions have a particular type of RAW data dependence. The type of RAW data dependence is identified by the stage that produces the result (EX or MEM) and the instruction that consumes the result (1st instruction that follows the one that produces the result, 2nd instruction that follows, or both). Assume that the register write is done in the first half of the clock cycle and that register reads are done in the second half of the cycle, so "EX to 3rd" and "MEM to 3rd" dependencies are not counted because they cannot result in data hazards. Also, assume that the CPI of the processor is 1 if there are no data hazards.

- "EX to 1st only 10%" indicates that the data produced in the EX stage is required only by the next instruction, and 10% represents that this scenario occurs in 10% of all executed instructions
- "EX to 2nd only 20%" indicates that the data produced in the EX stage is required only by the second instruction, and 20% represents that this scenario occurs in 20% of all executed instructions

EX to 1^{st} Only	MEM to 1^{st} Only	EX to 2^{nd} Only	MEM to 2^{nd} Only	EX to 1^{st} and MEM to 2^{nd}	Other RAW Dependences
5%	10%	15%	10%	10%	10%

2.1. (6%) If we use no forwarding, what fraction of cycles are we stalling due to data hazards?

2.2. (7%) If we use full forwarding (forward all results that can be forwarded), what fraction of cycles are we stalling due to data hazards?

2.3. (7%) Let us assume that we cannot afford to have three-input Muxes that are needed for full forwarding. We must decide if it is better to forward only from the EX/MEM pipeline register (next-cycle forwarding) or only from the MEM/WB pipeline register (two-cycle forwarding). Which of the two options results in fewer data stall cycles?

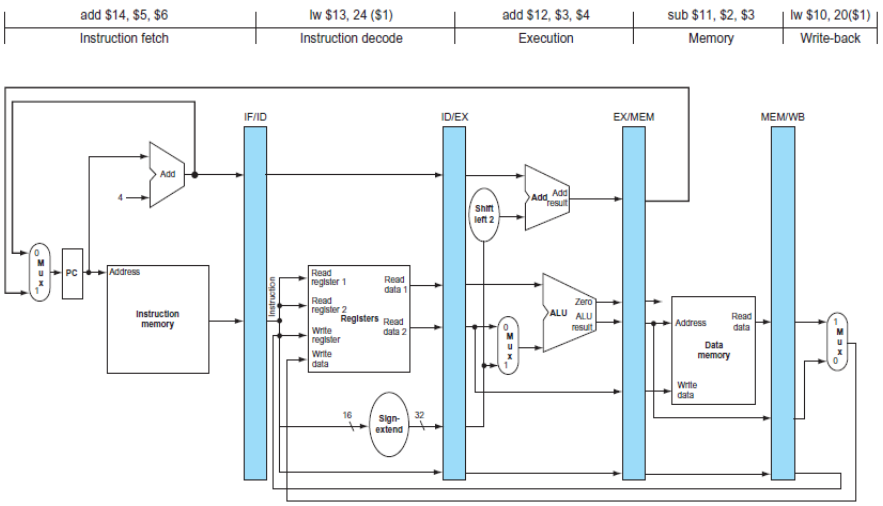

FIGURE 4.45 The single-clock-cycle diagram corresponding to clock cycle 5 of the pipeline in Figures 4.43 and 4.44.
As you can see, a single-clock-cycle figure is a vertical slice through a multiple-clock-cycle diagram.

三、(20%) This exercise is intended to help you understand the relationship between forwarding, hazard detection, and ISA design. Problems in this exercise refer to the following sequence of instructions, and assume that it is executed on a 5-stage pipelined datapath:

lw r4,4(r4)
sub r4,r2,r4
sw r3,0(r4)
or r3,r4,r3
lw r5,0(r4)

3.1.(5%) If there is no forwarding or hazard detection, how many **nops** needed to be inserted to ensure correct execution, and how many cycles needed to execute the codes?

3.2.(5%) If the processor has full forwarding and hazard detection, how many **nops** still needed to be inserted to ensure correct execution, and how many cycles needed to execute the codes?

3.3. (5%) If the processor **has forwarding**, but we **forgot to implement the load-use hazard detection unit**, what happens when this code executes?

3.4.(5%) If there is forwarding, for the first five cycles during the execution of this code, specify which signals are asserted in each cycle by **hazard detection and forwarding units** in the following Figure 4.55 and Figure 4.60 (PCWrite, IF/IDWrite, Forward_A, and Forward_B)

Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

FIGURE 4.55 The control values for the forwarding multiplexors in Figure 4.54. The signed immediate that is another input to the ALU is described in the *Elaboration* at the end of this section.

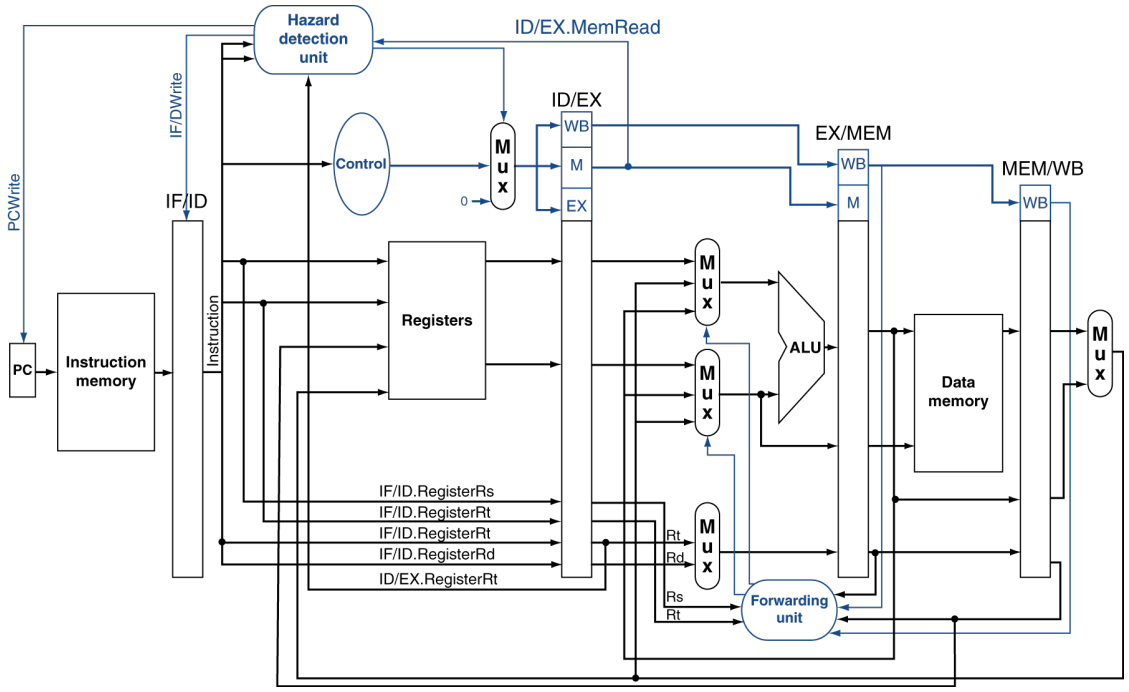

Figure 4.60

四、(20%) This exercise is intended to help you understand the relationship between delay slots, control hazards, and branch execution in a pipelined processor. In this exercise, we assume that the following MIPS code is executed on a pipelined processor with a 5-stage pipeline, full forwarding, and a predict-taken branch predictor:

```
            lw r2,0(r1)
label1:     beq r2,r0,label2 # not taken once, then taken
            lw r3,0(r2)
            beq r3,r0,label1 # taken
            add r1,r3,r1
label2:     sw r1,0(r2)
```

4.1 (10%) Assuming there are no delay slots and that branches **decide in the MEM stage**. How many cycles will it take to complete the above MIPS codes?

4.2. (10%) How many cycles will it take to complete? Assume that delay slots are used. In the given code, the instruction following the branch is now the delay slot instruction for that branch.
(Hint: Branches **decide** in the ID stage)

五、(20%) This exercise examines the accuracy of various branch predictors for the following repeating pattern (e.g., in a loop) of branch outcomes: T, T, NT, NT, T, T, NT

5.1.(6%) What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?

5.2. (7%)What is the accuracy of the two-bit predictor for the first 4 branches in this pattern, assuming that the predictor starts off in the bottom left state from Figure 4.63 (predict not taken)?

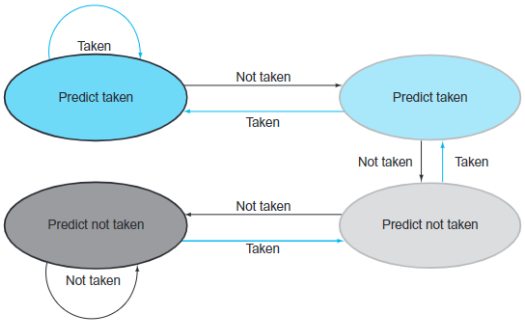

FIGURE 4.63 The states in a 2-bit prediction scheme. By using 2 bits rather than 1, a branch that strongly favors taken or not taken—as many branches do—will be mispredicted only once. The 2 bits are used to encode the four states in the system. The 2-bit scheme is a general instance of a counter-based predictor, which is incremented when the prediction is accurate and decremented otherwise, and uses the mid-point of its range as the division between taken and not taken.

5.3.(7%) What is the accuracy of the two-bit predictor if this pattern is repeated forever, assuming that the predictor starts off in the top left state from Figure 4.63 (predict taken)?