Homework 7

4.9:

> In this exercise, we examine how data dependencies affect execution in the basic 5-stage pipeline described in Section 4.5. Problems in this exercise refer to the following sequence of instructions:
>
> > or r1, r2, r3
> > or r2, r1, r4
> > or r1, r1, r2
>
> Also, assume the following cycle times for each of the options related to forwarding
>
> | Without Forwarding | With Full Forwarding | With ALU-ALU Forwarding Only |
> |---|---|---|
> | 250px | 300ps | 290ps |

4.9.1

> Indicate dependences and their type

Answer:

RAW on R1 from I1 to I2 and I3

RAW on R2 from I2 to I3

WAR on R2 from I1 to I2

WAR on R1 from I2 to I3

WAW on R1 from I1 to I3

4.9.2

> Assume there is no forwarding in this pipelined processor. Indicate hazards and add nop instructions to

> eliminate them.

Answer:

4.9.3

> Assume there is full forwarding. Indicate hazards and add NOP instructions to eliminate them.

Answer:

No nops are needed. Both RAW hazards are eliminated by forwarding

4.9.4

> What is the total execution time of this instruction sequence without forwarding and with full forwarding What is the speedup achieved by adding full forwarding to a pipeline that had no forwarding?

Answer:

clock cycle time * number of cycles

Without any stalls, a three instruction sequence executes in 7 cycles (5 to complete the first instruction, then one per instruction). The execution with or without forwarding must add a stall for every nop.

Without Forwarding	With Full Forwarding	Speedup
2700ps	2800ps	0.96
1800ps	2000ps	0.9

4.9.5

> Add nop instructions to this code to eliminate hazards if there is ALU-ALU forwarding only (no forwarding from the MEM to the EX stage

Answer:

4.9.6

> What is the total execution time of this instruction sequence with only ALU-ALU forwarding? What is the speedup over a no-forwarding pupeline?

Answer:

Without Forwarding	With Full Forwarding	Speedup
2700ps	3240ps	0.83
1800ps	1980ps	0.91

4.11:

> Consider the following loop
>
> > loop: lw r1, 0(r1)
> > and r1, r1, r2
> > lw r1, 0(r1)
> > lw r1, 0(r1)
> > beq r1, r0, loop
>
> Assume that perfect branch prediction is used (no stalls due to control hazards), and there are no delayslots, and that the pipeline has full forwarding support. Also assume that many iterations of this loop are executed before the loop exits.

4.11.1

> Show a pipeline execution diagram for the third iteration of this loop, from the cycle in which we fetch the first instruction of that iteration up to (but not including) the cycle in which we can fetch the first instruction of the next iteration. Show all instructions that are in the pupeline during these cycles (not jusst those from the third iteration).

Answer:

IF	ID	EX	MEM	WB
lw r1, 0(r1)	*	*	*	*
and r1, r1, r2		*	*	*
lw r1, 0(r1)			*	*
lw r1, 0(r1)				*
beq r1, r0, loop				

4.11.2

> How often(as a percentage of all cycles(do we have a cycle in which all five pipeline stages are doing useful work?

Answer: none of the cycles have all five pipeline stages doing useful work.

4.13:

> 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:
>
>> add r5, r2, r1
>> lw r3, 4(r5)
>> lw r2, 0(r2)
>> or r3, r5, r3 sw r3, 0(r5)

Answer:

4.13.1

> If there is no forwarding or hazard detection, insert nops to ensure correction execution.

Answer:

R1 I1 to I2 , I3
R2 I2 to I4

R1 I3 to I4

4.13.2

> Repeat 4.13.1 but now use nops only when a hazard cannot be avoided by changing or rearranging these instructions. You can assume register R7 can be used to hold temporary values in your modified code.

Answer:

R1 I1 to I2 , I3
R2 I1 to I3
R1 I1 to I2

4.13.3

> If the processor has forwarding, but we forgot to implement the hazard detection unit, what happens when this code executes?

Answer:

a stall will occur

4.13.4

> 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 Firgure 4.60.

Answer:

or r3, r5, r3) and lw r2, 0(r2)

4.13.5

> If there is no forwarding, what new inputs and output signals do we need for the hazard detection unit in Figure 4.60? Using this instruction sequence as an example, explain why each signal is needed

Answer:

4:15:

> This importance of having a good branch predictor depends on how often conditional branches are executed. Together with branch predictor accuracy, this will determine how much time is spent stalling due to mispredicted branches. In this exercise, assume that the breakdown of dynamic instructions into various instruction categories is as follows:
>
> | R-Type | BEQ | JMP | LW | SW |
> |---|---|---|---|---|
> | 40% | 25% | 5% | 25% | 5% |
>
> Also, assume the following branch predictor accuracies:
>
> | Always-Taken | Alwas-Not-Taken | 2-Bit |
> |---|---|---|
> | 45% | 55% | 85% |

4.15.1

> Stall cycles due to mispredicted branches increase the CPI. What is the extra CPU due to mispredicted branches with the always-taken predictor? Assume that branch outcomes are determined in the EX stage, that there are no data hazards, and that no delay slots are used.

Answer:

Type	CPI
always taken	CPI = 1 + 0.25 * (1 - 0.45) * 2 = 1.275
Always not taken	CPI = 1 + 0.25 * (1 - 0.55) * 2 = 1.225
2-bit predictor	CPI = 1 + 0.25 * (1 - 0.25) * 2 = 1.075