Faculty of Media Engineering and Technology
Dept. of Computer Science and Engineering
Dr. Cherif Salama
Eng. Nada Tarhony

CSEN 601: Computer System Architecture
Summer 2014

Practice Assignment 7 Solution

Exercise 7-1:
Based on the MIPS pipeline implementation you studied, what are the control signals that have to be stored in the ID/EX pipeline register? Group them based on the stage they are needed in.

Solution:

- Control signals needed in the EX phase: ALUSrc (1-bit), RegDest (1-bit), ALUOp (2-bits)
- Control signals needed in the MEM phase: MemRead (1-bit), MemWrite (1-bit), Branch (1-bit)
- Control signals needed in the WB phase: RegWrite (1-bit), MemToReg (1-bit)

Exercise 7-2:
Based on the MIPS pipeline implementation you studied, what are the sizes of the pipeline registers? Justify your answer. Ignore any bits required to detect or handle hazards.

Solution:

- The IF/ID pipeline register has: 64-bits
  - 32-bits instruction
  - 32-bits incremented PC

- The ID/EX pipeline register has: 147-bits
  - 32-bits incremented PC
  - 32-bits read register 1 value
  - 32-bits read register 2 value
  - 32-bits sign extended offset
  - 5-bits Rt field
  - 5-bits Rd field
  - 2-bits WB control signals
  - 3-bits MEM control signals
  - 4-bits EX control signals

- The EX/MEM pipeline register has: 107-bits
  - 32-bits branch address
  - 1-bit zero flag
  - 32-bits ALU result/address
  - 32-bits register value to write to memory
  - 5-bits Rd field (writeReg)
  - 2-bits WB control signals
  - 3-bits MEM control signals
- The MEM/WB pipeline register has: 71-bits

- 32-bits ALU result
- 32-bits memory word read
- 5-bits Rd field (writeReg)
- 2-bits WB control signals

Exercise 7-3:
For the following sequences of instructions:

1.
```
        lw   $1, 40($6)
        beq $2, $0, Label      Assume $2 == $0
        sw   $6, 50($2)
Label:  add $2, $3, $4
        sw   $3, 50($4)
```

2.
```
        lw   $5, -16($5)
        sw   $4, -16($4)
        lw   $3, -20($4)
        beq $2, $0, Label      Assume $2 != $0
        add $5, $1, $4
```

Assuming the following latencies for the individual pipeline stages:

	IF	ID	EX	MEM	WB
1.	100ps	120ps	90ps	130ps	60ps
2.	180ps	100ps	170ps	220ps	60ps

a. Assume that all branches are perfectly predicted (eliminating control hazards)
If we have only one memory (for both instructions and data), there is a structural hazard every time we need to fetch an instruction in the same cycle in which another instruction accesses data. To guarantee forward progress, this hazard must always be resolved in favor of the instruction that accesses data.
What is the total execution time of this instruction sequence in the five-stage pipeline that only has one memory?
Data hazards can be eliminated by adding ***nop***s to the code. Can structural hazard be eliminated in the same way? Why?

b. Assume that all branches are perfectly predicted (eliminating control hazards)
If we change load/store instructions to use a register (without an offset) as the address, these instructions no longer need to use the ALU. As a result, MEM and EX stages can be overlapped and the pipeline has only four stages.
Change this code to accommodate this changed ISA. Assuming this change doesn't affect clock cycle time, what speed-up is achieved this instruction sequence?

c. Repeat the speed-up calculation of part b, but take into account the possible change in clock cycle time and the provided pipeline stage latencies. When EX and MEM are done in a single stage, most of their work can be done in parallel. As a result, EX/MEM stage has a latency that is larger of the original two plus 20ps needed for the work that couldn't be done in parallel.

d. Assuming stall-on-branch, what speed-up is achieved on this code if branch outcomes are determined in the ID stage, relative to the execution where branch outcomes are determined in the EX stage?

e. Assume the latency ID stage increases by 50% and the latency of the EX stage decreases by 10ps when branch outcome resolution is moved to ID.
Repeat the speed-up calculation of part d, but take into account the possible change in clock cycle time and the provided pipeline stage latencies.

f. Assume stall-on-branch, what is the new clock cycle time and execution time of this instruction sequence if ***beq*** address computation is moved to the MEM stage? What is the speed-up in this case? Assume that the latency of the EX stage is reduced by 20ps and the latency of the MEM stage remains unchanged.

Solution:

a. Perfect branch prediction leads to no stalls.
In the pipelined execution, *** represents a stall when an instruction can't be fetched because a load or store instruction is using the memory in that cycle.
We can't add ***nop***s to eliminate structural hazards as ***nop***s need to be fetched just like any other instructions, so this hazard must be addressed with a hardware hazard detection unit in the processor.

	Instructions	Pipeline stage	Cycles

1.	lw $1, 40($6)
	beq $2, $0, Label
	add $2, $3, $4
	sw $3, 50($4)
		IF ID EX MEM WB
		IF ID EX MEM WB
		IF ID EX MEM WB
		*** IF ID EX MEM WB
			9

2.	lw $5, -16($5)
	sw $4, -16($4)
	lw $3, -20($4)
	beq $2, $0, Label
	add $5, $1, $4
		IF ID EX MEM WB
		IF ID EX MEM WB
		IF ID EX MEM WB
		*** *** *** IF ID EX MEM WB
		IF ID EX MEM WB
			12

b. This change only saves one cycle in an entire execution without data hazards.
If there were data hazards from loads to other instructions, the change would help eliminate some stall cycles.

	Instructions executed	Cycles with 5 stages	Cycles with 4 stages	Speed-up
1.	4	4+4 = 8	3+4 = 7	8/7 = 1.14
2.	5	4+5 = 9	3+5 = 8	9/8 = 1.13

c. The clock cycle time is equal to the latency of the longest-latency stage.
Combining EX and MEM stages affect clock time only if the combined EX/MEM stage becomes the longest-latency.

	Cycles time with 5 stages	Cycles time with 4 stages	Speed-up
1.	130ps (MEM)	150ps (MEM +20ps)	(8*130)/(7*150) = 0.99
2.	220ps (MEM)	240ps (MEM +20ps)	(9*220)/(8*240) = 1.03

d. Stall-on-branch delays the fetch of the next instruction until the branch is executed.
When branches execute in the ID stage, each branch cause one stall only.

	Instruction executed	Branches executed	Cycles with branch in EX	Cycles with branch in ID	Speed-up
1.	4	1	4+4+1*2 = 10	4+4+1*1=9	10/9 = 1.11
2.	5	1	4+5+1*2 = 11	4+5+1*1=10	11/10 = 1.1

e.

	New ID latency	NEW EX latency	New cycle time	Old cycle time	Speed-up
1.	180ps	80ps	180ps (ID)	130ps (MEM)	(10*130)/(9*180) = 0.8
2.	150ps	160ps	220ps (MEM)	220ps (MEM)	(11*220)/(10*220) = 1.1

f. The cycle time remains unchanged; a 20ps reduction in EX latency has no effect on clock cycle time because EX is not the longest-latency stage.
The change affects the execution time because it adds one additional stall cycle to each branch, because the clock cycle time doesn't improve but the number of cycles increases.

	Cycles with branch in EX	Execution time (branch in EX)	Cycles with branch in MEM	Execution time (branch in MEM)	Speed-up
1.	4+4+1*2 = 10	10*130 = 1300ps	4+4+1*3 = 11	11*130 = 1430ps	0.91
2.	4+5+1*2 = 11	11*220 = 2420ps	4+5+1*3 = 12	12*220 = 2640ps	0.92