Computer Science 605.611

Problem Set 6 Answers

1. The table below lists the minimum time required to complete all activity performed in each stage within our MIPS 5-stage pipeline:

Pipeline Stage	Time required
Fetch	12.5 ns
Decode	8.5 ns
Execute	11.5 ns
Memory	12.5 ns
Write-back	8ns

a) (3) Based on this information, what is the maximum clock rate that can be used for the pipeline?

The clock period has to be as long as the most time consuming stage.
So the maximum clock rate = 1/12.5ns = 80 MHz.

b) (3) In which pipeline stage does the instruction add $8, $4, $2 read $4 ?

All instructions read their input registers in stage 2 (the decode stage) during the second half of the clock cycle.

c) (3) In which pipeline stage does the instruction add $8, $4, $2 write $8 ?

Result registers are written in the write-back stage (stage 5) during the first half of the clock cycle.

d) (3) In which pipeline stage does the instruction lw $8, 4($2) compute the memory address of the word that it reads from memory?

In the execute stage (stage 3) the ALU is used to add the sign-extended displacement to the base register ($2 in this case) to produce the address of the word that is read from memory by the lw instruction in stage 4 (the memory stage).

e) (3) In which pipeline stage does the instruction beq $9,$6,exit compute the branch target address?

The branch target address is computed in stage 3, the execute stage using a separate adder to compute the sum of the PC contents plus the sign-extended immediate field shifted left two bits.

2. (6) What is the difference between a data dependency and a data hazard with respect to the pipeline?

A data dependency exists when an instruction uses an input operand that is written by another instruction ahead of it in the pipeline.
A data dependency is a data hazard only if the dependent instruction attempts to read the required result before the instruction that produces the result writes it.

3. Assume that the 5-stage MIPS pipeline runs at a clock rate of 0.05 GHz.

a) (3) With no data hazards, how many nano-seconds does the instruction lw $8, 4($2) take to go through the pipeline?

If there are no hazards then no stalls are required. The clock rate of 0.05 GHz corresponds to a cycle time of 1/0.05GHz = 20 ns.
Since each of the 5 stages must take 20ns, at least 5*20ns = 100ns are required for the lw instruction to go through the pipeline.

b) (3) With no data hazards, how many nano-seconds does the instruction add $8, $4, $2 take to go through the pipeline?

If there are no hazards then no stalls are required. However, every instruction must go through all 5 pipeline stages, so this instruction takes 100 ns as well.

4. a) (3) What is the maximum number of R-type instructions that our original 5-stage MIPS pipeline completes per second if it runs at 4 GHz?

A 4 GHz clock rate corresponds to a cycle time of 1/4GHz = 0.25 nano-seconds. The pipeline can complete at most 1 instruction per cycle. One instruction every 0.25 nano-seconds corresponds to 4 billions instructions per second.

b) (3) What is the maximum number of R-type instructions that our non-pipelined multi-cycle datapath completes per second if it runs at 4 GHz?

On the multi-cycle datapath, each R-type instruction requires 4 cycles. One instruction every 4 cycles corresponds to one instruction every 4*0.25 nano-seconds = 1 nano-second, or 1 billion instructions per second.

5. a) (4) Once a pipeline bubble is created to cope with a data hazard, what is the minimum number of stages through which the bubble must travel?

Pipeline bubbles are stages in which nothing happens which appear to be empty. Setting all 9 control bits to zero results in no action being taken within the stage containing the bubble. Bubbles are created in the decode stage and travel through the final 3 stages of the pipeline.

b) (4) What is the minimum number of pipeline stages through which a nop must travel?

A nop is a 32-bit machine instruction (it corresponds to the instruction sll $0,$0,0). Nops are fetched from memory like any other machine instruction and must travel through all 5 pipeline stages.

c) (4) Is a nop created by the assembler or is it created by the control unit?

The assembler generates a nop when it translates the instruction with the mnemonic “nop” into a machine instruction that corresponds to sll $0,$0,0 .

d) (4) Is a pipeline bubble created by the assembler or is it created by the control unit?

The control unit creates bubbles in stage 2 by outputting 9 zero control bits. Four of the bits (ALUSrc, RegDst, ALUOp1, ALUOp0) affect the execute stage, three of the control bits (Branch, MemRead, MemWrite) affect the memory stage and two control bits (MemtoReg & Regwrite) affect the write-back stage.

6. (8) Recall that with our 5-stage pipeline, register reads occur in the second half of the clock cycle while register writes occur in the first half of the clock cycle. Consider the following instruction sequence:

```
xor     $2, $0, $3
slt     $5 ,$2, $4
add     $11, $5, $11
sllv    $6, $11, $12
lw      $8, 0x800($2)
sub     $2, $6, $8
```

Assume that the pipeline system employs a hazard detection unit but no other techniques, such as data forwarding or code rearrangement, to cope with data hazards. If the xor instruction is fetched in clock cycle 1, during which clock cycle does the sub instruction complete its write-back stage?

Without a forwarding unit, dependent instructions must be stalled in the decode stage until the instruction producing the required input reaches the write-back stage. Hence the instructions flow through the pipeline as follows:

Cycle	Fetch	Decode	Exec	Mem	Write-back
1	xor $2, $0, $3				
2	slt $5 ,$2, $4	xor $2, $0, $3			
3	add $11, $5, $11	slt $5 ,$2, $4	xor $2, $0, $3		
4	add $11, $5, $11	slt $5 ,$2, $4	bubble	xor $2, $0, $3	
5	add $11, $5, $11	slt $5 ,$2, $4	bubble	bubble	xor $2, $0, $3
6	sllv $6, $11, $12	add $11, $5, $11	slt $5 ,$2, $4	bubble	bubble
7	sllv $6, $11, $12	add $11, $5, $11	bubble	slt $5 ,$2, $4	bubble
8	sllv $6, $11, $12	add $11, $5, $11	bubble	bubble	slt $5 ,$2, $4
9	lw $8, 0x800($2)	sllv $6, $11, $12	add $11, $5, $11	bubble	bubble
10	lw $8, 0x800($2)	sllv $6, $11, $12	bubble	add $11, $5, $11	bubble
11	lw $8, 0x800($2)	sllv $6, $11, $12	bubble	bubble	add $11, $5, $11
12	sub $2, $6, $8	lw $8, 0x800($2)	sllv $6, $11, $12	bubble	bubble
13		sub $2, $6, $8	lw $8, 0x800($2)	sllv $6, $11, $12	bubble
14		sub $2, $6, $8	bubble	lw $8, 0x800($2)	sllv $6, $11, $12
15		sub $2, $6, $8	bubble	bubble	lw $8, 0x800($2)
16			sub $2, $6, $8	bubble	bubble
17				sub $2, $6, $8	bubble
18					sub $2, $6, $8

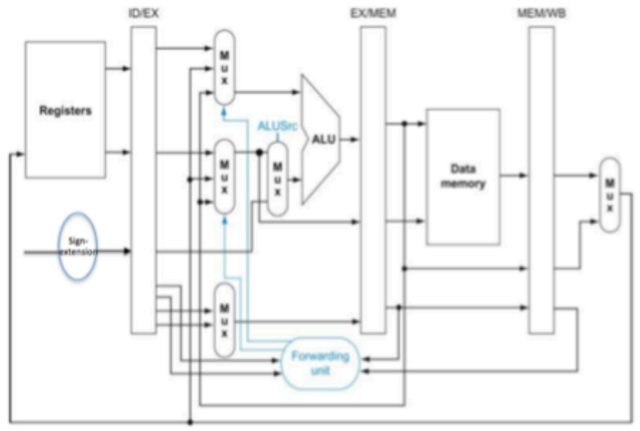
The sub instruction completes in cycle 18.

7. (5) A sequence of 21 MIPS instructions contains 10 R-type instructions, 5 lw instructions and 6 sw instructions. If there are no hazards of any type, what speedup would the 5-stage pipeline provide for this instruction sequence compared to the multi-cycle datapath with both datapaths running at the same clock rate? Express your answer to 2 decimal places.
On the multi-cycle datapath R-type instructions take 4 cycles, lw takes 5 and sw takes 4. Hence the total number of cycles consumed on the multi-cycle datapath is 10*4 + 5*5 + 6*4 = 89.
If there are no hazards, then no pipeline stalls or bubbles are needed. Hence the total number of cycles consumed on the pipeline is 5 + 20 = 25.
So the speedup provided by the pipeline is 89/25 = 3.56.

8. (5) A sequence of N instructions from our MIPS core instruction subset executing on the non-pipelined multi-cycle datapath achieves an average CPI of 3 cycles per instruction. Running this same sequence on our 5-stage pipeline with no hazards and at the same clock rate provides a speedup of 2.5. If the number of instructions in the sequence is doubled while keeping the average CPI the same, what speedup does the pipeline provide for this sequence of 2N instructions running at the same clock rate if there are no hazards? Round your answer to 2 decimal places.
For the sequence of N instructions the speedup = 2.5
= (cycle_time*N*3) / [cycle_time*(5+N-1)]
So N*3 = 2.5*(4+N)
N*3 – 2.5*N = 2.5*4
0.5*N = 10
N = 20

For the sequence of 2N instructions the speedup = (2*N*3) / (4 +2*N)
= 120 / 44 = 2.73

Base your answers to problems 9 through 11 on the version of the 5-stage pipelined datapath whose decode stage contains a data hazard unit and whose execute stage contains a forwarding unit is as shown below:



9. (8) Consider the following instruction sequence:

```
lui     $2, 0x3E4
lw      $8, 0x8($2)
add     $10, $2, $8
slt     $6, $10, $8
sw      $6,8($2)
sub     $2, $6, $8
```

Assume that the instructions are executed in the order shown. Complete the table below to show the instruction in each pipeline stage for each clock cycle until the entire instruction sequence completes. Do not transpose or otherwise rearrange the table. You may include as many additional rows as you need.

With a forwarding unit, any required inputs can be forwarded to the execute stage of dependent instructions eliminating one or more bubbles. However, a dependent instruction in a load delay slot (the slot immediately following a lw instruction) must be stalled in the decode stage until the lw completes its memory stage. Hence the instructions would flow through the pipeline as follows:

Cycle	Fetch	Decode	Exec	Mem	Write-back
1	lui $2,0x3E4				
2	lw $8,8($2)	lui $2,0x3E4			
3	add $10,$2,$8	lw $8,8($2)	lui $2,0x3E4		
4	slt $6,$10,$8	add $10,$2,$8	lw $8,8($2)	lui $2,0x3E4	
5	slt $6,$10,$8	add $10,$2,$8	bubble	lw $8,8($2)	lui $2,0x3E4
6	sw $6,8($2)	slt $6,$10,$8	add $10,$2,$8	bubble	lw $8,8($2)
7	sub $2,$6,$8	sw $6,8($2)	slt $6,$10,$8	add $10,$2,$8	bubble
8		sub $2,$6,$8	sw $6,8($2)	slt $6,$10,$8	add $10,$2,$8
9			sub $2,$6,$8	sw $6,8($2)	slt $6,$10,$8
10				sub $2,$6,$8	sw $6,8($2)
11					sub $2,$6,$8

10. The following short sequence of instructions runs on the pipelined datapath:

```
ori    $8,$0,0x1008
sll    $8$8,16
ori    $6,$0,4
sw     $6,4($8)
sw     $8,8($8)
lw    $6,0($8)
```

a) (3) When ori $6,$0,4 is in the execute stage, show what the following bit patterns should be for the datapath shown above:
ForwardA = __00__, ForwardB = __00__, ALUSrc = __1___
The ori $6,$0,4 instruction requires no forwarded operands, so ForwardA = 00, ForwardB=00 but ALUSrc=1 to select the constant 4 as its lower ALU input.

b) (3) When sw $6,4($8) is in the execute stage, show what the following bit patterns should be for the datapath shown above:
ForwardA = __01__, ForwardB = __10__, ALUSrc = __1___
The sw $6,4($8) requires the forwarded value for $8 (in the MEM/WB pipeline register produced by the sll instruction) as the upper ALU A input and requires the constant 4 as the lower ALU B input. Hence ALUSrc=1, ForwardA = 01 and ForwardB=10 to select the result in the EX/MEM pipeline register generated by the ori $6,$0,4 instruction to pass on to the next stage in place of the stale value that was read for $6 when the sw was in the decode stage.

c) (3) When sw $8,8($8) is in the execute stage, show what the following bit patterns should be for the datapath shown above:
ForwardA = __00__, ForwardB = __00__, ALUSrc = __1___
The sw $8,8($8) instruction uses $8 as the upper ALU input and the constant 8 as the lower ALU input. Register $8 will have already been written by the sll instruction when sw $8,8($8) reads register $8, so no forwarding is required. So ForwardA = 00, ForwardB = 00, and ALUSrc=1 to select the constant 8 as the lower ALU input.

d) (3) When lw $6,0($8) is in the execute stage, show what the following bit patterns should be for the datapath shown above:
ForwardA = __00__, ForwardB = __00_, ALUSrc = __1__
The lw $6,0($8) uses the constant 0 as the lower ALU B-input, and $8 produced by the sll instruction as the upper ALU A-input. Note that a store word instruction (sw) writes to memory, it does not write $8. So no forwarding is required in this case either. ForwardA = 00, ForwardB = 00 and ALUSrc=1 to select the constant 0 as the lower ALU B-input. This lw instruction reads $6 in stage 2 even though it does not use $6 as an input operand. Register $6 would have already been written by the second ori instruction in any case.

11. The following instruction sequence is executed in order on the pipelined datapath:

Ins1:	or $5,$0,$0
Ins2:	lui $7,0x3A
Ins3:	addi $8,$7,0x4004
Ins4:	sw $5,24($8)
Ins5:	lw $8,44($5)
Ins6:	add $6,$8,$5
Ins7:	sw $5,64($5)

a) (6) Use the labels (Ins1, Ins2, etc.) to identify all instructions in the sequence for which there is a data hazard.
Ins3 (addi) requires $7 from Ins2 (lui) which is a data hazard.
Ins4 (sw) requires $8 from Ins3 (addi) which is a data hazard.
Ins6 (add) requires $8 from Ins5 (lw) which is a data hazard.

Ins4 (sw) depends on $5 from Ins1 (or). However, this is not a data hazard since Ins1 writes $5 before Ins4 reads $5.

b) (4) Use the labels (Ins1, Ins2, etc.) to identify all dependent instructions in the sequence for which data forwarding eliminates its data hazard.
Data forwarding eliminates the data hazard for Ins3 and Ins4.

c) (3) For any dependent instructions in the sequence for which data forwarding does not eliminate all stalls, identify the dependent instruction and state how many cycles the dependent instruction must be stalled.
Ins6 (add) must be stalled for one cycle to allow Ins5 (lw) time to complete its memory read stage.