Computer Science 605.611

Problem Set 6

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?

Slowest cycle: 12.5 ns
Clock Rate = 1 / cycle time = 0.08 GHz

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

Register read operations are done during the Decode operation, which is the second pipeline stage in our MIPS 5-stage pipeline.

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

Register write operations occur in the Write-back operation, which is the fifth pipeline stage in our MIPS 5-stage pipeline.

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?

Memory address computations are done in the Execute stage, which is the third stage in our MIPS 5-stage pipeline.

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

Computing branch target addresses happens in the Execute stage, which is the third stage in our MIPS 5-stage pipeline.

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

A data dependency occurs when the result of one operation in the MIPS program is needed as the input to another MIPS operation. A data hazard occurs when the result of one operation is not yet written to the register before it is used as input to another MIPS operation, thus causing an incorrect result in the second operation. Not all data dependencies cause data hazards, however all data hazards are the eventual result of a data dependency.

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?

0.05 GHz Clock Rate -> cycle time = 20 ns
The lw instruction takes all 5 stages of the 5-stage MIPS pipeline to complete, therefore the lw instruction takes 100 ns to complete

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

The add instruction takes just 4 stages of the 5-stage MIPS pipeline to complete, since the add instruction does not have to go through the memory access cycle. Therefore the add instruction takes 80 ns to complete

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?

4 GHz Clock Rate -> cycle time = 0.25 ns
With no data hazards and utilizing pipelining, the 5-stage MIPS pipeline can complete one R-type instruction per clock cycle, thus each instruction would take 0.25ns. This translates to 4e^9 R-type 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?

Each R-type instruction takes 4 cycles of the 5-stage MIPS cycle, since R-type instructions do not have to complete the access memory operand (4^{th}) stage. If we do not utilize pipelining, each operation would take 1ns to complete, which would translate to 1e^9 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?

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

A bubble in our 5-stage MIPS cycle must travel three stages. This is because bubbles are created when an instruction to stall instructions in the instruction decode/register file read phase, and is propagated the next three cycles to get through the MIPS 5-stage system.

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

Nop is created by the assembler when no other operations can be put in between adjacent dependent instructions

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

Bubbles are created by the control unit whenever a data hazard is detected

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?

Cycle #	Fetch	Reg Read	ALU Op	Mem Access	Reg Write
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

sub completes write-back in stage 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.

R-type instructions take 4 cycles to complete on the multi-cycle datapath
lw instructions take 5 cycles to complete on the multi-cycle datapath
sw instructions take 4 cycles to complete on the multi-cycle datapath
4 * 10 + 5 * 5 + 4 * 6 = 89 cycles on the multi-cycle datapath

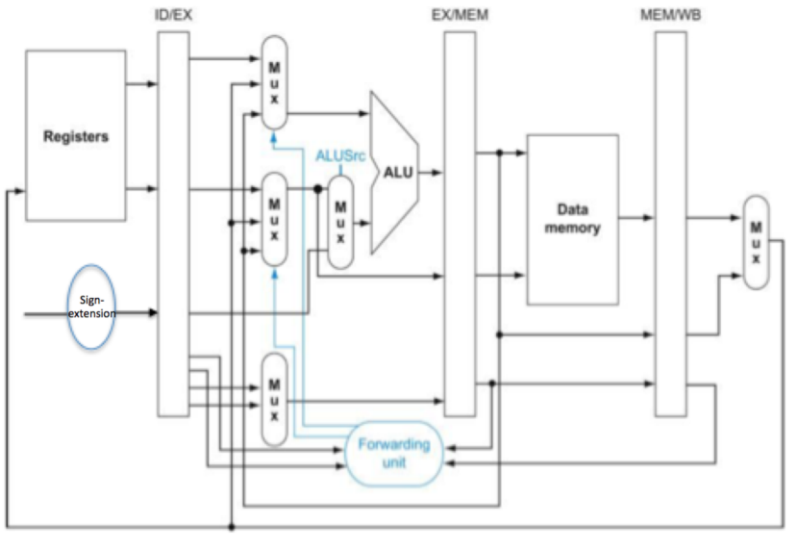
21 instructions running on the 5-stage pipeline would require 25 cycles

Therefore the speedup 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?

Assuming N is sufficiently large, comparing 2N instructions on the multi-cycle datapath to 2N instructions on the 5-stage pipeline should produce the same speedup: 2.5

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.

Cycle	IF	ID	EX	MEM	WB
1	lui $2, 0x3E4				
2	lw $8, 0x8($2)	lui $2, 0x3E4			
3	add $10,$2,$8	lw $8, 0x8($2)	lui $2, 0x3E4		
4	add $10,$2,$8	lw $8, 0x8($2)		lui $2, 0x3E4	
5	add $10,$2,$8	lw $8, 0x8($2)			lui $2, 0x3E4
6	slt $6,$10,$8	add $10,$2,$8	lw $8, 0x8($2)		
7	slt $6,$10,$8	add $10,$2,$8		lui $2, 0x3E4	
8	slt $6,$10,$8	add $10,$2,$8			lui $2, 0x3E4
9	sw $6, 8($2)	slt $6,$10,$8	add $10,$2,$8		
10	sub $2,$6,$8	sw $6, 8($2)	slt $6,$10,$8	add $10,$2,$8	

11		sub $2,$6,$8	sw $6, 8($2)	slt $6,$10,$8	add $10,$2,$8
12		sub $2,$6,$8		sw $6, 8($2)	slt $6,$10,$8
13		sub $2,$6,$8			sw $6, 8($2)
14			sub $2,$6,$8		
15				sub $2,$6,$8	
16					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 = 0

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 = 00, ForwardB = 01, ALUSrc = 1

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

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 = 01, ALUSrc = 1

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) (5) Use the labels (Ins1, Ins2, etc.) to identify all instructions in the sequence for which there is a data hazard.

lns3, lns4, lns5, lns6

b) (5) Use the labels (Ins1, Ins2, etc.) to identify all dependent instructions in the sequence for which data forwarding eliminates its data hazard.

lns4, lns6

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.

lns3 is dependent on lns2. lns3 must be stalled for 2 cycles to allow lns2 to reach the writeback stage.

lns5 is dependent on lns4. lns5 must be stalled for 2 cycles to allow lns4 to reach the writeback stage.