Computer Science 605.611

Problem Set 6

1. A certain program contains 2,480,000 instructions. Each instruction is executed exactly once on a processor running at a 4 GHz clock rate. There are no branch or jump instructions in the program. Twenty-two percent of the instructions are lw instructions, twelve percent of the instructions are sw instructions, and all of the remaining instructions are R-type instructions. Our MIPS pipeline system is an example of a scalar pipeline since no more than one instruction at a time can occupy any stage.

a) (5) If the first instruction to enter the scalar pipeline is an R-type instruction, what is the maximum number of nano-seconds required for that R-type instruction to complete all five pipeline stages if the clock rate is 4 GHz?

**It would take 1.25 ns for the instruction to complete. (each stage taking .25 ns)**

b) (5) Once the first R-type instruction completes the write-back stage, what is the maximum number of instructions per second that this pipeline can complete thereafter?

**Assuming each instruction takes around 1.25ns, it could complete 800 million instructions per second.**

2. (5) Each of the data paths listed below is described in this course. For each of these data paths, explain why including a data hazard unit is beneficial or is not beneficial.

Pipelined data path

The data hazard unit is incredibly beneficial in the pipelined data path. This is because the instructions are sent to multiple stages concurrently. Due to this without a data hazard unit there can become incorrect results due to certain instructions not finishing before another one or one being run to early, the hazard unit helps mitigate those issues.

Single-cycle data path

The data hazard unit would not be beneficial here. This is because only one instruction is being run at a time. With that being said there are rare cases that a data hazard unit may be used, but overall it is not very beneficial in this datapath.

Multi-cycle data path

It is still beneficial to have a data hazard unit in a multi cycle data path. Even though it is not being run concurrently like the pipelined data path there are still data hazards that can pop up here and there.

3. a) (3) If the table below shows the minimum number of nano-seconds needed to complete the activity performed in each of our 5 MIPS pipeline stages, what is the maximum clock rate that can be used for the pipeline?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fetch | Decode | Execute | Memory | Writeback |
| 10.25 ns | 6.75 ns | 8.5 ns | 12.5 ns | 4.25 ns |

The critical stage would be memory running at 12.25ns. This means that 1/12.25ns = **81.63MHz.**

b) (3) What is the minimum number of nano-seconds required for any type of instruction to go through all 5 stages on this pipeline?

The minimum number of ns would be 10.25+6.75+8.5+12.5+4.25 = 42.25ns

4. Recall that the 9 control bits used by the pipelined system are: ALUOp1, ALUOp0, ALUSrc, RegDst, RegWrite, MemtoReg, MemRead, MemWrite and Branch.

a) (3) Which of these control bits are used within the execute stage?

ALUOp1, ALUOp0, and ALUSrc are all apart of the execute stage (ALU controls execution)

b) (3) Which of these control bits are used within the memory stage?

MemRead and MemWrite are apart of the memory stage (They read or write to memory)

c) (3) Which of these control bits are used within the write-back stage?

RegWrite and MemtoReg (Regwrite if instruction should written to register, and MemtoReg if the data to be written is from memory)

5. Consider the following instruction sequence that runs on our MIPS 5-stage pipeline:

ori $5, $0, 1

or $6, $0, $0

sll $2, $0, 31

srl $2, $0, 3

add $8, $4, $4

lw $9 44($2)

beq $6, $7, exit

Assume that the instruction: ori $5, $0, 1 is fetched in clock cycle 1.

a) (3) During which clock cycle will the add $8, $4, $4 instruction read register $4?

Clock cycle 6 will have the add instruction read register $4.

b) (3) In which pipeline stage does the instruction: lw $9, 44($2) compute the address of its memory operand?

Cycle 8 it will be in the EX stage and compute the address of the memory operand

c) (3) What hardware unit or device within the pipelined data path is used to compute the branch target address for the beq $6 ,$7,exit instruction?

The ALU would handle this.

d) (3) If the beq $6, $7, exit instruction is fetched from memory address 0x40000008 and the low 16 bits of the instruction contain the pattern 0x8345, what address will the PC register contain when this branch instruction is in the decode stage?

e) (5) Assume that the beq $6, $7, exit instruction is fetched from memory address 0x40000008 and the low 16 bits of the instruction contain the pattern 0x8345. To what 32-bit memory address must the label exit correspond?

To do this you must subtract the branch offset from current address. 0x40000008 – 0x8345 = 0x3FFF7CC3

6. a) (3) What is the minimum number of clock cycles required to execute a nop machine instruction on our MIPS pipelined data path?

The minimum number of clock cycles required would be 1 clock cycle.

b) (3) What is the name and binary value of each control bit that is used when a nop machine instruction is in the execute stage?

ALUOp1: 0

ALUOp2: 0

ALUSrc: 0

RegDst: 0

RegWrite: 0

MemtoReg: 0

MemRead: 0

MemWrite: 0

Branch: 0

All 0 so nothing happens.

c) (3) Can a pipeline stall be caused by executing a nop machine instruction? If so, what is the maximum number of pipeline stalls that it can cause?

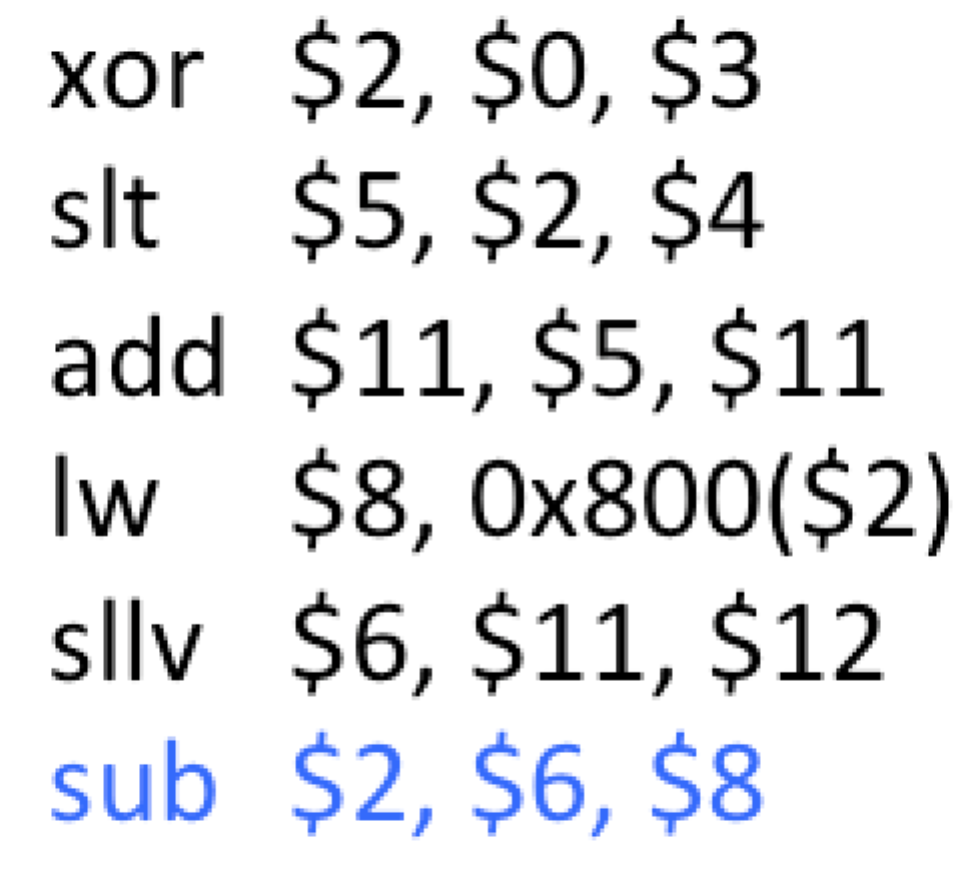
A nop machine instruction can be used to cause a stall. The maximum number of stalls it can cause would be 5 as that is the amount of stages in our pipeline.

7. (5) A sequence of MIPS instructions contains 17 R-type instructions,

5 lw instructions and 4 sw instructions for a total of 26 instructions. If there are no hazards of any type and both data paths run at the same clock rate, what speedup ratio does the 5-stage pipeline provide for this instruction sequence compared to the non-pipelined multi-cycle data path? Round your answer to two decimal places.

Based on the slides, out lw instructions take 800ps, sw takes 700ps, and R-type take 600ps. Giving us a time of 4000ps for lw, 2800ps for sw, and 10,200ps or a total of 17000ps or 17ns. So the first will take 1000ps and the rest 800ps so 1000ps + 25\*(800ps) = 21000ps or 21ns. SO the speedup would be **1.47**

8. Recall that with our 5-stage MIPS 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:



a) (5) Assume that the pipeline system employs a hazard detection unit but it does not use data forwarding unit. If the xor instruction is fetched in clock cycle 1, during which clock cycle does the sub $2, $6, $8 instruction complete its write-back stage? The instructions must be executed in the order shown.

It wouldn’t be until cycle 13 that sub completes its write back.

b) (5) Assume that the same instruction sequence is executed again, this time with both a data forwarding unit as well as a hazard detection unit, but with no other techniques to handle data hazards. If the xor instruction is fetched in clock cycle 1, during which clock cycle does the sub $2, $6, $8 instruction now complete its write-back stage? The instructions must be executed in the order shown.

It would be in cycle 10 that sub completes its writeback stage.

9. Consider the three instructions:

sw $3, 4($8)

lw $3, 4($8)

slt $2, $3, $0

a) (3) If these instructions are executed on our pipelined data path with a data hazard unit but with no data forwarding unit, how many clock cycles will the lw instruction be stalled in the decode stage?

The lw instruction will be stalled in the decode stage for one cycle, this is because there is data hazard between sw and lw instructions.

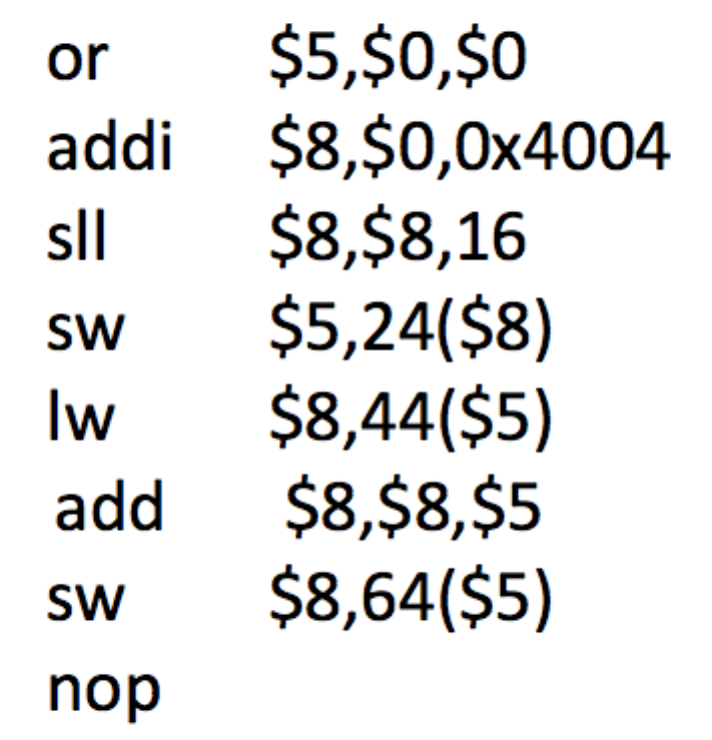
b) (3) If these instructions are executed on our pipelined data path with a data hazard unit but with no data forwarding unit, how many clock cycles will the slt instruction be stalled in the decode stage?

The slt will also be stalled for one cycle in the decode stage waiting for the result of the lw.

c) (3) If these instructions are executed on our pipelined data path containing both a data hazard unit and a data forwarding unit, how many clock cycles will the slt instruction be stalled in the decode stage?

There will be no stalls for the slt instruction since we have both a hazard unit and a data forwarding unit.

10. Assume that the instruction sequence shown below is executed on our 5-stage MIPS pipeline in the order shown. The pipeline contains a data hazard detection unit. It does not contain a data forwarding unit.



a) (5) How many clock cycles are required to complete this instruction sequence on the pipeline?

It would take 20 cycles to complete this instruction sequence.

b) (5) What speedup ratio (rounded to two decimal places) is provided for this same instruction sequence by executing it on the version of the pipelined data path shown below?

Diagram, engineering drawing

Description automatically generated

Since there is a data forwarding unit in this example compared to our MIPS 5 stage data path described above, we can figure a few things. Looking at this we only need 13 clock cycles as we no longer need as many stalls. This gives us 20/13 or **1.54**

11. (5) When a dependent instruction is stalled in stage 2 of our MIPS pipeline, the instruction ahead of it in stage 3 is allowed to advance. Hence, in the next clock cycle, stage 3 will be empty. This empty stage is called a pipeline bubble. Is the ALU operation performed when a bubble is in the execute stage the same ALU operation as that performed when a nop machine instruction is in the execute stage? Explain your answer.

It is at least a similar to the nop instruction. This is because a bubble and nop both just add an instruction to create space so the data hazard is mitigated. They are not exactly the same thing but they serve a near identical purpose.

12. (5) The following instruction sequence executes on our MIPS pipeline with both a data hazard unit and a data forwarding unit:

ori $2, $0 ,5

addiu $4, $2 ,21

sll $0, $4, 3

add $6, $4, $0

Recall that forwarding can allow dependent instructions to avoid stalling since any stale register value the instruction uses can be replaced by the forwarded value when the dependent instruction is in the execute stage. Identify all instructions in this sequence that require one or more forwarded values and indicate the stale register whose contents should be replaced.

Addiu requires a forwarded value from register $2 and sll requires a forwarded value from register $4.