Using Ubuntu Linux

1. In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

IF	ID	EX	MEM	WB
400ps	300ps	200ps	350ps	250ps

Also, assume that instruction executed by the processor are distributed as follows:

ALU	BEQ	LW	SW
40%	30%	20%	10%

(a) What is the clock cycle time in a pipelined and non-pipelined processor?

- In a pipelined process the clock cycle time is 400ps.
- In a non-pipelined processor the clock cycle time is 1500 (400ps + 300ps + 200ps + 350ps + 250ps). Each stage must be completed one after the other.

(b) What is the total latency of an LW instruction in a pipelined and non-pipelined processor? LW uses all stages.

- pipelined: 400ps*5 = 2000ps
- non-pipelined: 1500ps

(c) If we can split one stage of the pipelined data path into two new stages, each with half the latency of the original stage, which stage would you split and what are the new clock time and latency of an LW in the pipelined processor?

- The largest stage should be split up. If IF is split then the cycle time would be 350 for pipelined.
- For nonpipelined it would stay the same since it must go through all stages. 1500ps
- The latency of an LW in a pipelined processor would be 5*350 = 1750

(d) Assuming there are no stalls or hazards, what is the utilization of the data memory?

- 30% would be the utilization of data memory since only LW and Store word use data memory.

(e) Assuming there are no stalls or hazards, what is the utilization of the write-register port of the `Registers` unit

i. 60% for ALU and LW utilization

(f) Instead of a single-cycle organization, we can use a multi-cycle organization where each instruction takes multiple cycles but one instruction finishes before another is fetched. In this organization, an instruction only goes through stages it actually needs (e.g.., SW only takes 4 cycles because it does not need the WB stage). What are the average cycle times of each type of processors (single-cycle, multiple-cycle, and pipelined)?

$$n = \frac{\text{performance(X)}}{\text{performance(Y)}}$$

Instruction	# Stages
Branch Jump	3
Arithmetic-logical	4
Stores	4
Load	5

- Single-cycle: (400ps + 300ps + 200ps + 350ps + 250ps) = 1500ps
- Multi-cycle: $400 \cdot (4 \cdot .4 + 3 \cdot .3 + 4 \cdot .1 + 5 \cdot .2)) = 1560$ps
- Pipelined: 400ps

2. In this exercise, we examine how data dependence affect execution int the basic 5-stage pipeline described in Chapter 4.5 (An overview of pipelining). Problems in this exercise refer to the following sequence of instructions:

```
1  add r3, r1, r2
2  add r2, r1, r3
3  add r2, r2, r3
```

Also, assume registers can read and written in the same cycle and the following cycle times for each of the options related to forwarding.

(a) Indicate ALL dependencies and their type (RAW/WAR/WAW).

i. RAW (Read after Write)
ii. WAR (Write after Read)
iii. WAW (Write after Write)

- RAW: on r3 from I1 to I2
- RAW: on r3 from I1 to I3
- RAW: on r2 from I2 to I3
- WAR: on r2 from I1 to I2
- WAW: on r2 from I2 to I3

Explanation:

(b) Assume there is **no forwarding** in this pipelined processor. Draw a pipeline execution diagram. Insert **nop** instructions to eliminate any hazards.
WAR and WAW do no cause any hazards.
Since add is R-Format it does not use MEM stage

```
1       add r3, r1, r2
2       nop
3       nop
4       add r2, r1, r3
5       nop
6       nop
7       add r2, r2, r3
```

```
1                          R3(ready)      r2(ready)
2  ------------------------------------------------------------------
3  |   | cc1| cc2| cc3| cc4| cc5| cc6| cc7| cc8| cc9| cc10| cc11|
4  ------------------------------------------------------------------
5  |   | IF | ID | EX | MEM| WB|
6  |       | NOP|   -   -   -   -   -   -   -
7  |           | NOP|   -   -   -   -   -   -   -
8  |               | IF | ID | EX | MEM| WB|
9  |                   | NOP|    -   -   -   -   -   -
10 |                       | NOP|   -   -   -   -   -
11 |                           | IF | ID | EX | MEM| WB |
```

(c) Assume there is full forwarding. Draw a pipeline execution diagram. Insert nop instructions to eliminate any hazards.
WAR and WAW do no cause any hazards.
ALU instruction can forward a value to EX stage of the next instruction withouth a hazard.

```
1  add r3, r1, r2
2  add r2, r1, r3 – No RAW hazard on r3 from I1 (forwarded)
3  add r2, r2, r3 – No RAW hazard on r2 from I2 (forwarded)
4  – No Raw hazard on r3 (I3) from I1
```

```
1                          R3(ready)      r2(ready)
2
3  ------------------------------------------------------------------
4  |   | cc1| cc2| cc3| cc4| cc5| cc6| cc7| cc8| cc9| cc10| cc11|
5  ------------------------------------------------------------------
6  |   | IF | ID | EX | MEM| WB|
7  |       | IF | ID | EX | MEM| WB|
8  |           | IF | ID | EX | MEM| WB |
```

(d) 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?

- No forwarding: $(7+4)*200ps = 2200ps$
- Full forwarding: $(7)*350ps = 2450$
- Speed up: $2200/2450 = .89$

(e) Assume there is ALU-ALU forwarding only (no forwarding from the MEM to the EX stage). Draw a pipeline execution diagram. Insert nop instructions to eliminate any hazards.

- ALU instruction can forward to the next instruction,but not to the second-next instruction (That would be forwarding from MEM to EX). A load cannot forward at all because it determines the data value in MEM stage.

```
            add r3, r1, r2
            add r2, r1, r3
            add r2, r2, r3
```

```
_____________________________________________________________________
|   | cc1| cc2| cc3| cc4| cc5| cc6| cc7| cc8| cc9| cc10| cc11|
_____________________________________________________________________
|   | IF | ID | EX | MEM| WB|
|       | IF | ID | EX | MEM| WB|
|           | NOP|   —   —   —   —   —   —   —
|               | NOP|   —   —   —   —   —   —   —
|                   | IF | ID | EX | MEM| WB |
```

(f) What is the total execution time of this instruction sequence with only ALU-ALU forwarding? What is the speedup over a no-forwarding pipeline?

```
_____________________________________________________________________
|   | cc1| cc2| cc3| cc4| cc5| cc6| cc7| cc8| cc9| cc10| cc11|
_____________________________________________________________________
|   | IF | ID | EX | MEM| WB|
|       | IF | ID | EX | MEM| WB|
|           | IF | ID | EX | MEM| WB |
```

3. In this exercise, we examine how resource hazards, control hazards, and Instruction Set Architecture (ISA) design can affect pipelined execution. Problems in this exercise refer to the following fragment of MIPS code:

```
    sw r16, 12(r6)
    lw r16, 8(r6)
    beq r5, r4, Label
```

```
4     add r5, r1, r4
5     slt r5, r15 r4
```

Unless stated, all problems are independent scenarios
Also, assume registers can be read and written in the same cycle and the following cycle times for each of the options related to forwarding:

IF	ID	EX	MEM	WB
200ps	100ps	150ps	250ps	100ps

(a) For this problem, assume that all branches are perfectly predicted (this eliminates all control hazards) and that no delay slots are used. Draw a pipeline execution diagram. What is the total execution time of the instruction sequence?

- No stalls

```
1  ____________________________________________________________________
2  |   | cc1| cc2| cc3| cc4| cc5| cc6| cc7| cc8| cc9| cc10| cc11| Cycles
3  ____________________________________________________________________
4  |   | IF | ID | EX | MEM| WB|                                  | 8
5  |       | IF | ID | EX | MEM| WB|
6  |           | IF | ID | EX | MEM| WB |
7  |               | IF | ID |  EX| MEM| WB|
8  |                   | IF |  ID| EX|  WB
```

(b) For this problem, assume that all branches are **perfectly predicted** (this eliminates all control hazards) and that **no delay slots are used**. 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, every instruction will only utilize either MEM or EX stage, and the pipeline has only 4 stages. Draw a pipeline execution diagram to reflect this change. Assuming this change does not affect clock cycle time, what is the total execution time of the instruction sequence? What is the speedup of this change compared to part a?

i. 250*8 = 2000

ii. 2000/2250 = .89

```
1  ____________________________________________________________________
2  |   | cc1| cc2| cc3| cc4| cc5| cc6| cc7| cc8| cc9| cc10| cc11| Cycles
3  ____________________________________________________________________
4  |   | IF | ID | MEM| WB|                                  | 8
5  |       | IF | ID | MEM| WB|
6  |           | IF | ID | EX | MEM| WB |
7  |               | IF | ID | EX | MEM| WB|
8  |                   | IF |  ID| EX | WB|
```

(c) For this problem, assume that all branches are **perfectly predicted** (this eliminates all control hazards) and that no delay slots are used. If we only have 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. Draw a pipeline execution diagram. What is the total execution time of this instruction sequence in the 5-stage pipeline that only has one memory?

i. 11*250 = 2750

```
1  ____________________________________________________________________
2  |   | cc1| cc2| cc3| cc4| cc5| cc6| cc7| cc8| cc9| cc10| cc11| Cycles
3  ____________________________________________________________________
4  |   | IF | ID | EX|  MEM| WB|                                  | 11
5  |       | IF | ID | EX | MEM| WB|
6  |           | IF | ID | EX | MEM| WB |
7  |                               | IF | ID | EX | MEM| WB|
8  |                                   | IF |  ID| EX | WB|
```

(d) We have seen that data hazards can be eliminated by adding nops to the code. Can you do the same with the structural hazard described in part c? Why?

i. NOPs are fetched just like any other instructions,' so this hazard must be addressed with a hardware hazard detection unit in the processor.

(e) Assuming stall-on-branch and no delay slots and branch outcomes are determined in the ID stage instead of EXE stage, draw a pipeline execution diagram. Insert **nops** if necessary. What is the total execution time of the instruction sequence? 10 cycles. 10* 250 = 2,500 - beq nop

- Need to enter a nop.

```
1  ____________________________________________________________________
2  |   | cc1| cc2| cc3| cc4| cc5| cc6| cc7| cc8| cc9| cc10| cc11| Cycles
3  ____________________________________________________________________
4  |   | IF | ID | EX|  MEM| WB|                                  | 10
5  |       | IF | ID | EX | MEM| WB|
6  |           | IF | ID | EX | MEM| WB |
7  |       NOP    —   —   —   —   —   —   —   —
8  |      NOP    —   —   —   —   —   —   —   —
9  |                     | IF | ID | EX | MEM| WB|
10 |
```