Homework #2

Assigned: 7/04/2021
Due: 14/04/2021

Note: Only hardcopy submissions are acceptable

Cavit Çakır 23657

1. (**50 pts**) Consider the following piece of code that will be executed in a five-stage pipelined datapath of MIPS processor:

```
lw  $t2, 4($t5)
add $t7, $t3, $t3
sw  $t7, -24($t2)
```

Assume that a value written to a register can be read in the following clock cycle.

a. Indicate dependencies. (**10 pts**)

First Instruction: No dependecy, t2 avaiable after mem
Second Instruction: No dependecy, t7 avaiable after exe
Third Instruction: t7 and t2 depends on first and second instructions.

b. Assume that there is no forwarding in this pipelined processor. Indicate hazards and add **nop** instructions to eliminate them. (**10 pts**)

```
if  -> id -> exe  -> mem ->  wb
     if  -> id   ->  exe   ->  mem ->  wb
          nop -> nop   ->   nop   -> nop -> nop
                 nop   ->   nop   -> nop ->  nop ->  nop
                            nop   -> nop ->  nop ->  nop -> nop
                                     if   ->  id   ->  exe ->  mem ->  wb
```

In instruction 3, 2 data hazards occurs due to t7 and t2 dependencies.
t7 and t2 must be ready at the id stage of instruction 3.

c. Assume there is forwarding only from the ALU. Indicate hazards and add `nop` instructions to eliminate them. (**10 pts**)

```
if  -> id -> exe  -> mem ->   wb
       if  -> id    ->  exe   ->  mem ->  wb
              nop -> nop   ->   nop   ->  nop -> nop
                       nop   ->   nop   -> nop ->  nop ->  nop
                                           if    ->  id    ->  exe ->  mem ->  wb
```

A data hazard lefts.
Only Instruction 3 waits for the instruction 1 to finish.
Due to ALU forwarding, t1 is available after wb.
Instruction 2 is available after mem stage.

d. Assume there is full forwarding. Indicate hazards and add `nop` instructions to eliminate them. (**10 pts**)

```
if  -> id -> exe  -> mem ->   wb
       if  -> id    ->  exe   ->  mem ->  wb
              if    ->  id    ->     exe ->    mem ->  wb
```

Use the following clock cycle times for the remainder of this exercise. Notice that the forwarding technique complicates the data path, which can result in slower clock frequency.

Without forwarding	With ALU forwarding	With full forwarding
300 ps	360 ps	400 ps

e. What is the total execution time of this instruction sequence without forwarding, with ALU-forwarding and with full forwarding? (**10 pts**)

- Without forwarding: 10 cycle * 300 = 3000ps
- With ALU-forwarding: 9 cycle * 360 = 3240ps
- With full forwarding: 7 cycle * 400 = 2800ps

2. (**25 pts**) Consider the following code segment executing on a five-stage MIPS processor:

```
      ...
Loop: lw    $t0,  0($s1)
      sub   $t0,  $t0,  $s2
      sw    $t0,  0($s1)
      addi  $s1,  $s1,  -4
      bne   $s1,  $zero, Loop
      ...
```

a. During the seventh clock cycle of the first iteration of the loop, which registers are being read and which registers are being written? Assume that data forwarding is used. Do not change the order of the instructions. (**5 pts**)

bne reads s1's and zero's value
sub writes t0

(**Hint:** You can fill the following table for the schedule of the code)

instruction	CC 1	CC 2	CC 3	CC 4	CC5	CC 6	CC 7	CC 8	CC 9
lw	IF	ID	EX	MEM	WB				
sub			if	id	ex	mem	wb		
sw				if	id	ex	mem	wb	
addi					if	id	exe	mem	wb
bne						if	id	exe	mem

b. Reorder the instructions so that all data and control hazards are eliminated. Assume that **delayed-branch** technique is used. (**Hint:** the delayed branch technique places an independent instruction into the slot following a branch instruction, where this instruction is always executed independent of the branch outcome. Note that the branch outcome is known in the ID stage of the pipeline.) (**10 pts**)

```
Loop:  lw    $t0,   0($s1)
       addi  $s1,   $s1,    -4
       sub   $t0,   $t0,    $s2
       bne   $s1,   $zero, Loop
       sw    $t0,   4($s1)
```

c. Unroll the loop twice and remove all hazards. Assume that `$s1` contains an even number before the loop. (**10 pts**)

```
Loop:   addi    $s1,    $s1,    -8
        lw      $t0,    8($s1)
        lw      $t1,    4($s1)
        sub     $t0,    $t0,    $s2
        sub     $t1,    $t1,    $s2
        sw      $t0,    8($s1)
        sw      $t1,    4($s1)
        bne     $s1,    $zero,  Loop
```

3. (**25 pts**) Consider a five-stage pipelined datapath for MIPS architecture with the following latencies:

IF	ID	EX	MEM	WB	Pipeline registers
335 ps	315 ps	365 ps	335 ps	300 ps	25 ps

Note that pipeline registers also have some latency, namely 25 ps (last column in the table).

a. Assuming there are no stalls, what is the speedup achieved by pipelining a single-cycle datapath? (**10 pts**)

A cycle = 335 + 315 + 365 + 335 + 300 = 1650 ps
Longest stage = 365 + 25 = 390 ps
improvement = 1650 / 390 = 4.23 times faster by pipelining.

b. Assume that instructions of a benchmark executed by the pipelined processor are broken down as follows:

Load	Store	Branch	Jump	R-type
30%	15%	10%	5%	40%

Also assume that 40% of loads are immediately followed by an instruction that uses the result of load; and branch penalty on misprediction is three clock cycles and 20% of branches are mispredicted. Jumps take two clock cycles to execute. Compute CPI of the pipelined processor (**15 pts**).

Loads requires 1 extra cycle due to next operation available at mem stage.

Load = (0.3) * (0.6 * 5 + 0.4 * 6) = 1.62 cycle
Store = 0.15 * 5 = 0.75 cycle
Branch = 0.1 * (0.8 * 5 + 0.2 * 8) = 0.56 cycle
Jump = 0.05 * 2 = 0.1 cycle
R-type = 0.4 * 5 = 2 cycle
Total = 5.03 cycle