Architetture dei Sistemi di Elaborazione	Delivery date: October 29th 2021
Laboratory 3	Expected delivery of lab_03.zip must include: - `program_2_a.s`, `program_2_b.s` and `program_2_c.s` - this file compiled and if possible in pdf format.

Please, configure the winMIPS64 simulator with the *Base Configuration* provided in the following:

- Code address bus: 12
- Data address bus: 12
- Pipelined FP arithmetic unit (latency): 4 stages
- Pipelined multiplier unit (latency): 8 stages
- divider unit (latency): not pipelined unit, 12 clock cycles
- Forwarding is enabled
- Branch prediction is disabled
- Branch delay slot is disabled
- *Integer ALU: 1 clock cycle*
- *Data memory: 1 clock cycle*
- *Branch delay slot: 1 clock cycle*.

1) Starting from the assembly program you created in the previous lab called **`program_2.s`**,:

```
for (i = 0; i < 40; i++){
        v5[i] = v1[i]+(v2[i] * v3[i]);
        v6[i] = v5[i]*v4[i];
        v7[i] = v6[i]/v2[i];
}
```

   a. Detect manually the different data, structural and control hazards that provoke a pipeline stall
   b. Optimize the program by re-scheduling the program instructions in order to eliminate as much hazards as possible. Compute manually the number of clock cycles the new program (**`program_2_a.s`**) requires to execute, and compare the obtained results with the ones obtained by the simulator.

   c. Starting from **`program_2_a.s`**, enable the *branch delay slot* and re-schedule some instructions in order to improve the previous program execution time. Compute manually the number of clock cycles the new program (**`program_2_b.s`**) requires to execute, and compare the obtained results with the ones obtained by the simulator.

   d. Unroll 4 times the program (**`program_2_b.s`**), if necessary re-schedule some instructions and renaming the used registers. Compute manually the number of clock cycles the new program (**`program_2_c.s`**) requires to

execute, and compare the obtained results with the ones obtained by the simulator.

Complete the following table with the obtained results:

Program / Clock cycle computation	`program_2.s`	`program_2_a.s`	`program_2_b.s`	`program_2_c.s`
`By hand`	1686	981	981	796
`By simulation`	1686	981	981	796

Compare the results obtained in the point 1, and provide some explanation in the case the results are different.

Eventual explanation:

a. Il modo in cui Winmips calcola gli stalli strutturali è leggermente diverso da quello che abbiamo visto a lezione. In particolare, quando lo stadio di fetch o decode stalla perché gli stadi successivi sono occupati, Winmips non lo conta come stallo mentre noi a lezione si. Vengono conteggiati quindi solo 80 stalli strutturali (2 per ciclo) durante i quali la causa è la fase di MEM occupata, mentre contando anche gli altri sarebbero molti di più (ne ho contati circa 2000).
   Control hazards ne rilevo come Winmips uno ogni taken-branch, quindi 39.
   Gli stalli dovuti a dipendenze di dato sono il vero collo di bottiglia di questo programma, il calcolo a mano mi è venuto diverso da quello del simulatore in quanto abbiamo usato due approcci leggermente diversi. Winmips infatti anche se ha una dipendenza di dato entra nella fase di esecuzione dell'istruzione ed esegue il primo clock, liberando così lo stadio di decode per permettere all'istruzione successiva di entrarci; in questo modo alcuni degli stalli che avevo conteggiato come strutturali diventano di dato. Ho contato 1400 data stall contro i 1680 di winmips.
   In generale il numero e tipo di stalli calcolati dal simulatore rispetto a quelli "a mano" cambia, ma il conteggio totale dei colpi di clock necessari è identico.
b. Semplicemente cambiando l'ordine delle operazioni e usando due registri in più ho quasi dimezzato le istruzioni necessarie. Ho cercato di fare la maggior parte delle operazioni possibile durante la lunga divisione che blocca il programma.
c. Il fatto di effettuare un'operazione dopo il branch non mi ha portato vantaggi in quanto era un'operazione "gratis" ovvero che non occupa colpi di clock. Ho solamente ridotto il numero di stalli di controllo.
d. Ho potuto notare che il guadagno non è affatto un 4x con l'unroll, infatti la presenza della divisione unpipelined impedisce di eseguire le operazioni in "parallelo". Sarebbe probabilmente ancora leggermente migliorabile ordinando opportunamente le operazioni.