Computer Architectures 02LSEOV **Laboratory 3**	Delivery date: **By 2:00 AM on October, 23 2024** Expected delivery of lab_03.zip must include: - program_1_a.s, program_1_b.s, and program_1_c.s - This file, filled with information and possibly compiled in a pdf format.

This lab will explore some of the concepts seen during the lessons, such as hazards, rescheduling, and loop unrolling. The first thing to do is to configure the WinMIPS64 simulator with the *Initial Configuration* provided below:

- *Integer ALU: 1 clock cycle*
- *Data memory: 1 clock cycle*
- Code address bus: 12
- Data address bus: 12
- FP arithmetic unit: pipelined, 4 clock cycles
- FP multiplier unit: pipelined, 6 clock cycles
- FP divider unit: not pipelined, 30 clock cycles
- Forwarding is enabled
- Branch prediction is disabled
- Branch delay slot is disabled

1) Enhance the assembly program you created in the previous lab called **`program_1.s`**:

```
int m=1 /* 64  bit */
double a, b
for (i = 31; i >= 0; i--){
        if (i is a multiple of 3) {
                a = v1[i] / ((double) m<< i) /*logic shift */
                m = (int) a
        } else {
                a = v1[i] * ((double) m* i))
                m = (int) a
        }
        v4[i] = a*v1[i] – v2[i];
        v5[i] = v4[i]/v3[i] – b;
        v6[i] = (v4[i]-v1[i])*v5[i];
}
```

a. Manually detect the different data, structural, and control hazards that cause a pipeline stall.

b. Optimize the program by re-scheduling the program instructions to eliminate as many hazards as possible. Manually calculate the number of clock cycles for the new program (`program_1_a.s`) to execute and compare the results with those obtained by the simulator.

c. Starting from `program_1_a.s`, enable the *branch delay slot* and re-schedule some instructions to improve the previous program execution time. Manually

calculate the number of clock cycles needed by the new program (`program_1_b.s`) to execute and compare the results obtained with those obtained by the simulator.

d. Unroll the program (**`program_1_b.s`**) 3 times; if necessary, re-schedule some instructions and increase the number of registers used. Manually calculate the number of clock cycles to execute the new program (**`program_1_c.s`**) and compare the results obtained with those obtained by the simulator.

Complete the following table with the obtained results:

Program / Clock cycle computation	`program_1.s`	`program_1_a.s`	`program_1_b.s`	`program_1_c.s`
By hand	1920	2694	2640	3971
By simulation	1926	3941	123	3254

2) Collect the Cycles Per Instruction (CPI) from the simulator for different programs

	`program_1.s`	`program_1_a.s`	`program_1_b.s`	`program_1_c.s`
CPI	3988	4326	3618	4596

Compare the results obtained in 1) and provide some explanation if the results are different.

Eventual explanation:

The difference between the theoretical results and those obtained on WinMIPS for the different programs is mainly due to the more detailed simulation offered by WinMIPS compared to manual calculations. Manual calculations often idealize execution, ignoring effects such as pipeline stalls, memory latency, and branch penalties—especially in the absence of branch prediction—while WinMIPS takes these details into account. Other factors include the efficiency of forwarding to reduce dependencies and contention for resources like arithmetic units. Consequently, WinMIPS provides a more realistic representation of the actual execution of the code. In program 1_b the high discrepancy is due to the branch delay slot enabled in WinMIPS