|  |  |
| --- | --- |
| **Architetture dei Sistemi di Elaborazione 02GOLOV** | **Computer Architectures**  **02LSEYG** |
| **Laboratory**  **0x03** | Expected delivery of lab\_03.zip must include:   * **program\_1.s, program\_1\_a.s**, and **program\_1\_b.s** * This file, filled with information and possibly compiled in a **PDF** format. |
| Delivery date:  October 30th 2025 | |

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 GEM5 simulator with the ***Initial Configuration*** provided below:

**INTEGER\_ALU\_LATENCY = 1**

**INTEGER\_MUL\_LATENCY = 1**

**INTEGER\_DIV\_LATENCY = 1**

**FLOAT\_ALU\_LATENCY = 4**

**FLOAT\_MUL\_LATENCY = 6**

**FLOAT\_DIV\_LATENCY = 12**

1. Enhance the assembly program you created in the previous lab called **program\_1.s**:

int m = 1;

float a, b;

for (i = 31; i >= 0; i--) {

    if (i is a multiple of 3) {

        a = v1[i] / ((float) m << i); /\*logic shift \*/

        m = (int) a;

    } else {

        a = v1[i] \* ((float) 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];

}

* + 1. Manually detect the different data, structural, and control hazards that cause a pipeline stall. Report at least 3 hazards in the code (or the pipeline) and fill the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **C Line** | **Corresponding RISC-V Instruction(s)** | **Type of Hazard** | **Pipeline Stage** | **Cause of Stall** |
| e.g. x = (a + b) \* 2; | ADD x5, x1, x2  SLLI x6, x5, 1 | Data - RAW | EX | x5 not ready for shift until ADD complete |
| e.g. if (a > b) x = c; else x = d; | BGT x1, x2, L1  J L2 | Control | IF/ID | … |
|  |  |  |  |  |

* + 1. Optimize the program by re-scheduling 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.
    2. Unroll the program (**program\_1\_a.s**) two times; If necessary, re-schedule instructions and increase the number of registers used. Manually calculate the number of clock cycles to execute the new program (**program\_1\_b.s**) and compare the results obtained with those obtained by the simulator.

Complete the following table with the obtained results:

|  |  |  |  |
| --- | --- | --- | --- |
| **Program** | **program\_1.s** | **program\_1\_a.s** | **program\_1\_b.s** |
| **Clock cycles by hand** |  |  |  |
| **Clock cycles by simulation** |  |  |  |

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | **program\_1.s** | **program\_1\_a.s** | **program\_1\_b.s** |
| **CPI** |  |  |  |

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

Eventual explanation: