|  |  |
| --- | --- |
| **Architetture dei Sistemi di Elaborazione 02GOLOV** | **Computer Architectures**  **02LSEYG** |
| **Laboratory**  **0x04** | Expected delivery of lab\_04.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 deadlines:**  **GROUP1: 07/11/2025**  **GROUP2: 08/11/2025**  **GROUP3: 12/11/2025**  **(you can check your group on the new schedule on the portal)**  **THE NAME OF THE UPLOADED ZIP MUST BE “lab\_04.zip”**  **If you completed the laboratory in collaboration with other students (maximum of three per group), include at the beginning of your report the following statement: "Solution developed in collaboration with**  **[Collaborator’s Last Name] [Collaborator’s Student ID]" If there are two collaborators, list both names and IDs.**  **Every member must upload the report.** | |

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 = 8**

**FLOAT\_DIV\_LATENCY = 20**

1. Write an assembly program (**program\_1.s**) for the RISCV architecture able to compute the output (y) of a **neural computation** (see the Fig. below):

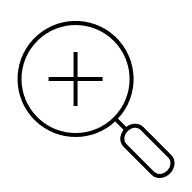
*x =*

*y = f(x)*

where, to prevent the propagation of NaN (Not a Number), the activation function *f* is defined as:

Assume the vectors *i* and *w* respectively store the inputs entering the neuron and the weights of the connections. They contain *K=16* single precision **floating point** elements. Assume that *b* is a single precision **floating point** constant and is equal to *0xab*, and *y* is a single precision **floating point** value stored in memory.

Compute *y*.

Immagine che contiene cerchio, schermata, nero, design

Il contenuto generato dall'IA potrebbe non essere corretto.

Given the *Base Configuration*, run your program and extract the following information.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Number of clock cycles | Total Instructions | CPI (Clock per Instructions) |
| program\_1.s |  |  |  |

* + 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** |  |  |  |

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

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

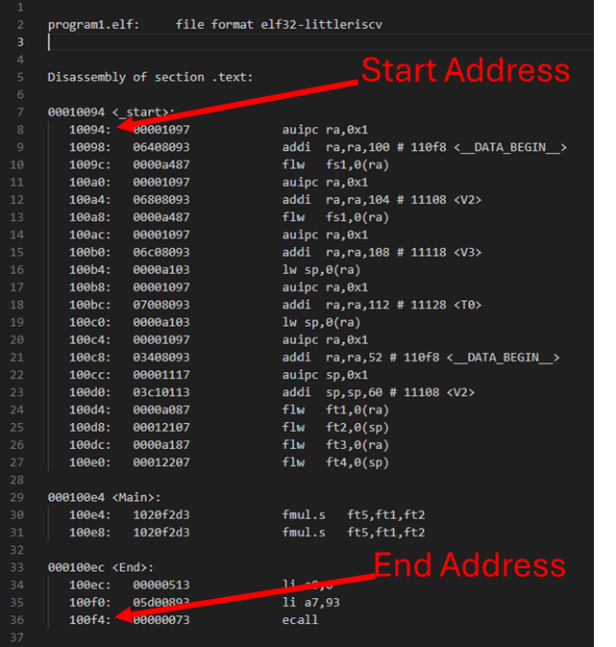
Compare the results obtained in 1) and provide some explanation if the results are different.

Eventual explanation:

* + 1. Try different unrolls for the program (**program\_1\_a.s**); 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** | **Number of Loop Unrolling** | **Code size [bytes]** | **Clock Cycles**  **[CC]** |
| **program\_1\_a.s** |  |  |  |
| 2 |  |  |
| 4 |  |  |
| 8 |  |  |
| 16 |  |  |

In order to calculate the code size of your program you can open in *./programs/you\_program/your\_program*.*dump* file to retrieve the compiled disassembled code of your executable in which you have the effective addresses from which the code will be executed.

For example (on the right), for *your\_program* you have your disassembled file on 32 bit addresses.