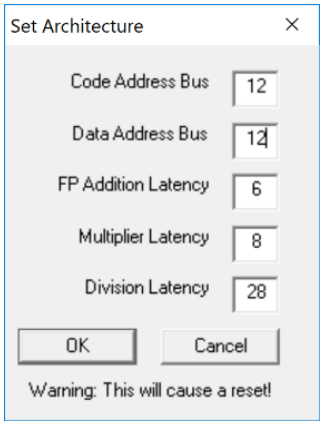
Architetture dei Sistemi di Elaborazione	Delivery date: October 22nd 2021
Laboratory 2	Expected delivery of lab_02.zip must include: - **`program_2.s`** and **`program_3.s`** - This file, filled with information and possibly compiled in a pdf format.

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): 6 stages
- Pipelined multiplier unit (latency): 8 stages
- divider unit (latency): not pipelined unit, 24 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) Write an assembly program (**`program_2.s`**) for the *winMIPS64* architecture described before able to implement the following piece of code described at high-level:

```
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];
}
```

Assume that the vectors `v1[]`, `v2[]`, `v3[]`, and `v4[]` are allocated previously in memory and contain 40 double precision **floating point** values; assume also that `v2[]` does not contain 0 values. Additionally, the vectors `v5[]`, `v6[]`, `v7[]` are empty vectors also allocated in memory.

a. Using the simulator and the *Base Configuration*, disable the Forwarding option and compute how many clock cycles the program takes to execute.

	Number of clock cycles
`program_2.S`	2648

Enable one at a time the **optimization features** that were initially disabled and collect statistics to fill the following table (fill all required data in the table before exporting this file to pdf format to be delivered).

Table 1: **Program performance for different processor configurations**

	Number of clock cycles			
Program	Forwarding	Branch Target Buffer	Delay Slot	Forwarding + Branch Target Buffer
`program_2`	2248	2613	2648	2213

2) Write an assembly program (**`program_3.s`**) for the winMIPS64 architecture able to compute the output (y) of a **neural computation** (see the Fig. below):

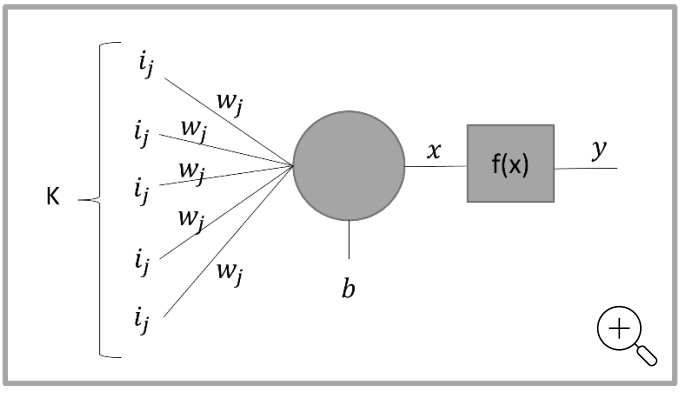
$$x = \sum_{j=0}^{K-1} i_j * w_j + b$$
$$y = f(x)$$

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

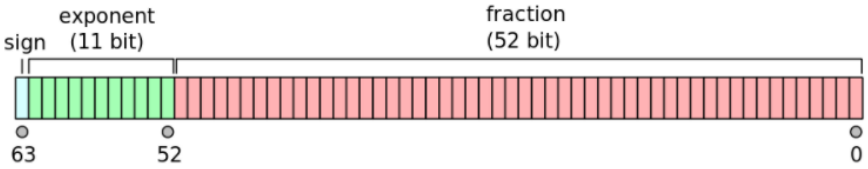
$$f(x) = \begin{cases} 0, & if\ the\ exponent\ part\ of\ x\ is\ equal\ to\ 0x7ff \\ x, & otherwise \end{cases}$$

Assume the vectors i and w respectively store the inputs entering the neuron and the weights of the connections. They contain $K=30$ double precision **floating point** elements. Assume that b is a double precision **floating point** constant and is equal to $0xab$, and y is a double precision **floating point** value stored in memory.

Compute y.



Below is reported the encoding of IEEE 754 double-precision binary floating-point format:



3) Using the WinMIPS64 simulator, validate experimentally the Amdahl's law, defined as follows:

$$speedup_{overall} = \frac{execution\ \ time_{old}}{execution\ \ time_{new}} = \frac{1}{(1 - fraction_{enhanced}) + \frac{fraction_{enhanced}}{speedup_{enhanced}}}$$

a. Using the program developed before: **`program_2.s`**
b. Modify the processor architectural parameters related with multicycle instructions (`Menu→Configure→Architecture`) in the following way:

1) Configuration 1
   - Starting from the *Base Configuration*, change only the FP addition latency to 3
2) Configuration 2
   - Starting from the *Base Configuration*, change only the Multiplier latency to 4
3) Configuration 1
   - Starting from the *Base Configuration*, change only the division latency to 12

Compute by hand (using the Amdahl's Law) and using the simulator the speed-up for any one of the previous processor configurations. Compare the obtained results and complete the following table.

Table 1: **`program_2.s` speed-up computed by hand and by simulation**

Proc. Config. / **Speed-up comp.**	Base config. [c.c.]	Config. 1	Config. 2	Config. 3
`By hand`	2169	1,05	1,17	1,28
`By simulation`	2248	2248 / 2128 = 1,05	2248 / 1928 = 1,16	2248 / 1768 = 1,27