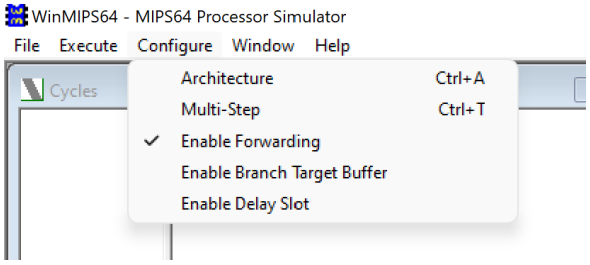
Architetture dei Sistemi di Elaborazione 02GOLOV **Laboratory 1**	Delivery date: **By 2:00 AM on October, 9 2024** Expected delivery of lab_01.zip including: - program_0.s - lab_01.pdf (fill and export this file to pdf)

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

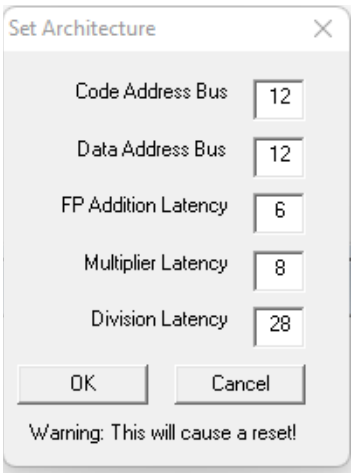
- *Integer ALU: 1 clock cycle*
- *Data memory: 1 clock cycle*
- *Branch delay slot: 1 clock cycle*
- Code address bus: 12
- Data address bus: 12
- Pipelined FP arithmetic unit (latency): 6 stages
- Pipelined FP multiplier unit (latency): 8 stages
- FP divider unit (latency): not pipelined unit, 28 clock cycles
- Forwarding optimization is disabled
- Branch prediction is disabled
- Branch delay slot optimization is disabled.

Use the Configure menu:

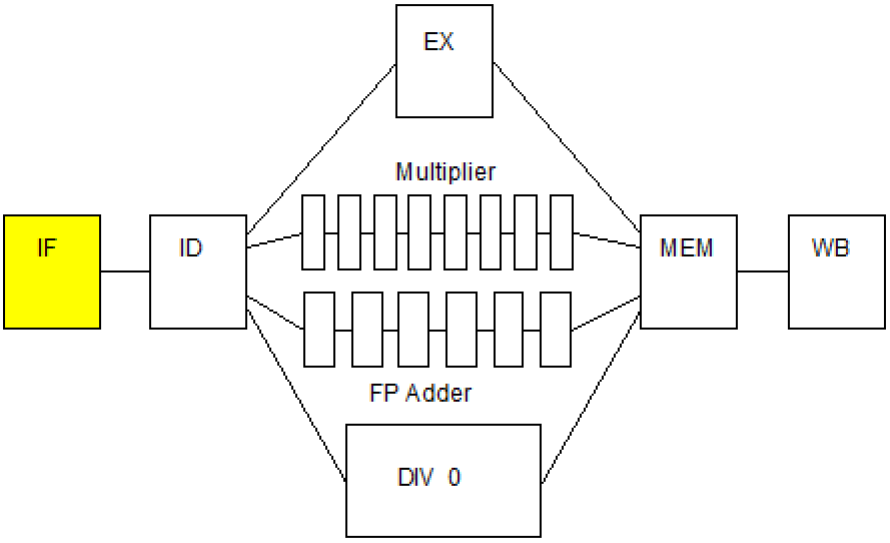
- Running the *WinMIPS* simulator, launching the graphical user interface (*folder_to_simulator*)...\winMIPS64\winmips64.exe
- Disable ALL the optimization (a mark appears when they are enabled)
- Browse the Architecture menu (Ctrl-A)



- Modify the defaults Architectural parameters (where needed)



- Verify in the Pipeline window that the configuration is effective (usually in the left bottom window)



1) Exercise your assembly skills.
Write and run an assembly program called **program_0.s (to be delivered)** for the *MIPS64* architecture.

The program must:

1. Given two arrays of 10 8-bit integer numbers (`v1`,`v2`), check if any element of `v1` is included in `v2`, at least once. Save the matching value in a third vector (`v3`).

An example:

```
v1:     .byte     2, 6, -3, 11, 9, 18, -13, 16, 5, 1
v2:     .byte     4, 2, -13, 3, 9, 9, 7, 16, 4, 7
```

The third vector will be composed as follows.

```
v3:     .byte     2, 9, -13, 16
```

2. Set three flags (`flag1, flag2, flag3`) to indicate three conditions:
   a. The third vector *(`v3`)* is empty. Use one 8-bit unsigned variable (`flag1`) to flag the condition. The variable will be equal to 1 if `v3` is empty, `0` otherwise.

   b. The third vector *(`v3`)* is not empty, and each element is greater than the previous one (`v3[i+1]>v3[i]`). In this case, use one 8-bit unsigned variable (`flag2`) to flag the condition. The variable will be equal to 1 if condition is satisfied, 0 otherwise.

   c. The third vector *(`v3`)* is not empty, and each element is smaller than the previous one (`v3[i+1]<v3[i]`). In this case, use one 8-bit unsigned variable (`flag3`) to flag the condition. The variable will be equal to 1 if condition is satisfied, 0 otherwise.

2) Use the *WinMIPS* simulator.
Identify and use the main components of the simulator:
   a. Running the *WinMIPS* simulator

      - Launch the graphic interface

      ...\winMIPS64\winmips64.exe

   b. Load your program in the simulator:

      - Load the program from the **`File→Open`** menu (*CTRL-O*). In the case the of errors, you may use the following command in the command line to compile the program and check the errors:

      `...\winMIPS64\asm program_0.s`

   c. Run your program step by step (*F7*), identifying the whole processor behavior in the six simulator windows:
   Pipeline, **Code**, **Data**, **Register**, **Cycles** and **Statistics**

   d. Collect the clock cycles 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 the specific processor configurations**

Program	Clock cycles	Number of Instructions	Clocks per instruction (CPI)	Instructions per Clock (IPC)
`program_0`	`1281`	`708`	`1,809`	`0,552693`

3) Perform execution and time measurements.
   Measure the processor performance by running a benchmark of programs. Change the weights of the programs as indicated in the following to evaluate how these variations may produce different performance results.

   Search in the winMIPS64 folder the following benchmark programs:
   a. `testio.s`
   b. `mult.s`
   c. `series.s`
   d. `program_0.s` (your program)

Starting from the basic configuration with no optimizations, compute by simulation the number of cycles required to execute these programs; in this initial scenario, it is assumed that the weight of the programs is the same (25%) for everyone. Assume a processor frequency of 1.75 kHz (*a very old technology node*).

Then, change processor configuration and vary the programs' weights as follows. Compute again the performance for every case and fill the table below **(fill all required data in the table before exporting this file to pdf format to be delivered)**.:

1) Configuration 1
   a. Enable Forwarding
   b. Disable branch target buffer
   c. Disable Delay Slot
   Assume that the weight of all programs is the same (25%).

2) Configuration 2
   a. Enable Forwarding
   b. Enable branch target buffer
   c. Disable Delay Slot
   Assume that the weight of all programs is the same (25%).

3) Configuration 3
   Configuration 1, but assume that the weight of the program *your program* is 43.33%.

4) Configuration 4
   Configuration 1, but assume that the weight of the program `series.s` is 60%.

Table 2: **Processor performance for different weighted programs**

Program	No opt	Conf. 1	Conf. 2	Conf. 3	Conf. 4
testio.s	(739*0,25)/ 1750=**0,106**	(476*0,25)/ 1750=**0,068**	(432*0,25)/1750= **0,062**	(476*(17 /90))/ 1750 =**0,051**	(476*(2/15))/1750= **0,036**
mult.s	(1880*0,25)/ 1750=**0,269**	(980*0,25)/ 1750=**0,14**	(922*0,25)/1750= **0,132**	(980*(17 /90))/ 1750 =**0,106**	(980*(2/15))/1750= **0,075**
series.s	(550*0,25)/ 1750=**0,079**	(233*0,25)/ 1750=**0,033**	(234*0,25)/1750= **0,033**	(233*(17 /90))/ 1750 =**0,025**	(233*0,6)/ 1750= **0,08**
program_0.s	(1281*0,25)/ 1750=**0,146**	(1158*0,25)/ 1750=**0,165**	(1046*0,2 5)/1750= **0,149**	(1158*0, 4333)/ 1750 =**0,287**	(1158* (2/15))/ 1750= **0,088**
TOTAL Time (@ 1.75kHz) **[secondi]**	***0,6***	***0,406***	***0,376***	***0,469***	***0,279***