Architetture dei Sistemi di Elaborazione
02GOLOV [M-Z]

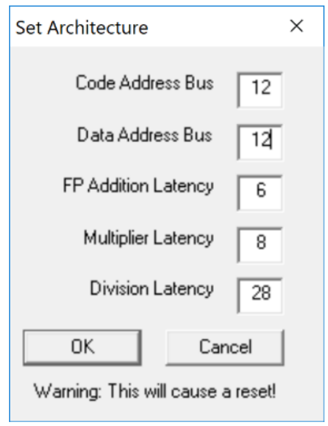
Laboratory 2

Delivery date:
31/10/2019

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 < 30; i++){
        v5[i] = (v1[i]*v2[i]) + v3[i];
        v6[i] =(v3[i]*v4[i])/v5[i]:

}
```

Assume that the vectors v1[], v2[], v3[], and v4[] are allocated previously in memory and contains 30 double precision floating point values; **assume that v5[] will not contain 0 values**. Additionally, the vectors v5[], v6[] are empty vectors also allocated in memory.

a. Using the simulator and the *Base Configuration*, compute how many clock cycles take the program to execute.

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

$$\text{speedup}_{\text{overall}} = \frac{\text{execution time}_{\text{old}}}{\text{execution time}_{\text{new}}} = \frac{1}{(1-\text{fraction}_{\text{enhanced}}) + \dfrac{\text{fraction}_{\text{enhanced}}}{\text{speedup}_{\text{enhanced}}}}$$

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`	1415	1.041	1.081	1.45
`By simulation`	1415	1.032	1.067	1.30

3) Write an assembly program (**`program_3.s`**) for the winMIPS64 architecture able to compute the hamming distance between two consecutive elements of a data array X[] previously allocated in memory (the array X contains 30 8-bit elements). Given two consecutive elements X[i] and X[i+1], the Hamming distance is defined as the number of bits set to 1 in (X[i] ^ X[i+1]).

    If the Hamming distance is:
    - even, the variable `even_counter` must be incremented
    - odd, the variable `odd_counter` must be incremented

    These two variables are allocated in memory and initially empty.

4) Considering the following *winMIPS64* architecture:
    - Code address bus: 12
    - Data address bus: 12
    - Pipelined FP arithmetic unit (latency): 4 stages
    - Pipelined multiplier unit (latency): 8 stages
    - divider unit (latency): not pipelined unit, 12 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.*

a. calculate by hand, how many clock cycles take the program to execute?

Number of clock cycles:	1448

b. compute the same calculation using the *winMIPS64* simulator.

Number of clock cycles:	1478 e 652

Compare the results obtained in the points 4.a and 4.b., and provide some explanation in the case the results are different.

Eventual explanation:

Ho considerato il peggior caso per calcolare il numero di cicli:
considerando che ogni elemento del vettore sia pari a 255, quindi che la funzione "counter_ones" si ripeta ogni volta per 8 volte. Si ottiene 1448
6+30*[5+8*4+1+4+3+2+1]+1+1 = 48*30+8 = 1448.

Ho riportato nel numero di clock sia il valore con un vettore contente solo 255 (1478) e sia un valore calcolato da un vettore casuale (652)