|  |  |
| --- | --- |
| **Architetture dei Sistemi di Elaborazione 02GOLOV [A-L]** | Delivery date:  October 29, 2020 |
| **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 < 5; i++){

for (j = 0; j < 5; j++) {

v3[i][j] = v1[i][j] \* v2[i][j];

v5[i][j] = v3[i][j] \* v4[i][j];

}

}

Assume that v1, v2 and v4 are two 5x5 matrixes allocated previously in memory and containing double precision floating-point values. Additionally, the matrixes v3,v5 are initially empty and allocated in memory.

* + 1. Using the simulator and the *Base Configuration*, compute how many clock cycles take the program to execute.

1. Using the WinMIPS64 simulator, validate experimentally the Amdahl’s law, defined as follows:



1. Using the program developed before: **program\_2.s**
2. 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

* + - * 1. Configuration 2

Starting from the *Base Configuration*, change only the Multiplier latency to 4

* + - * 1. 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** |  | 1 | 1.496 | 1 |
| **By simulation** |  | 1 | 1,493 | 1 |

1. Write an assembly program (**program\_3.s**) for the winMIPS64 architecture that implements the following piece of code:

**unsigned char** a[30];

**unsigned char** b[30];

**unsigned char** res[30];

**for** (i = 0; i < 30; i++){

**while** (b[i] > 0)

{

**if** (isOdd (b[i])) {

res[i] = res[i] + a[i];

}

a[i] = a[i] \* 2;

b[i] = b[i] / 2;

}

}

Assume vectors a and b are previously allocated in memory. Populate the vectors with values chosen by you. Assume also that res is and empty vector in memory. The function isOdd returns 0 when the number is even, 1 when odd. **Please note that the function should be replaced with the proper piece of code (function call not required)**.

Which is the operation implemented by the above code?

**Your Answer: res[i]=a[i]\*b[i]**

1. 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*.

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

|  |  |
| --- | --- |
| Number of clock cycles: | 786 |

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

|  |  |
| --- | --- |
| Number of clock cycles: | 786 |

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: