|  |  |
| --- | --- |
| **Computer Architectures**  **02LSEOV** | Delivery date:  October 16th 2025 |
| **Laboratory**  **2** | The expected delivery of lab\_02.zip must include:   * **program\_1.s** * This file, filled with information and possibly compiled in a pdf format. |

Please configure the WinMIPS64 simulator with the *Initial Configuration* provided below c):

* *Integer ALU: 1 clock cycle*
* *Data memory: 1 clock cycle*
* Code address bus: 12
* Data address bus: 12
* FP arithmetic unit: pipelined, 4 clock cycles
* FP multiplier unit: pipelined, 6 clock cycles
* FP divider unit: not pipelined, 30 clock cycles

1. Write an assembly program (**program\_1.s**) for the *WinMIPS64* architecture described before being able to implement the following high-level code:

for (i = 31; i >= 0; i--){

v4[i] = v1[i]\*v1[i] – v2[i];

v5[i] = v4[i]/v3[i] – v2[i];

v6[i] = (v4[i]-v1[i])\*v5[i];

}

Assume that the vectors v1[], v2[], and v3[] have been previously allocated in memory and contain 32 double-precision **floating-point values;** also assume that v3[] doesnot contain 0 values. Additionally, the vectors v4[], v5[], v6[] are empty vectors also allocated in memory.

**Calculate** the data memory footprint of your program:

|  |  |
| --- | --- |
| Data | Number of bytes |
| V1 |  |
| V2 |  |
| V3 |  |
| V4 |  |
| V5 |  |
| V6 |  |
| Total |  |

Are there any issues? Yes, where and why? No? Do you need to change something?

|  |
| --- |
| Your answer: |

ATTENTION: WinMIPS64 has a limitation regarding the maximum length of the string when declaring a vector. It is therefore recommended to split the elements of the vectors into multiple lines: this also increases readability.

Example: my\_fancy\_vector: .byte 8, 12 ,2, 9

.byte 49,77, 28

.byte ……

* + - Calculate the CPU performance equation (CPU time) of the above program by assuming a clock frequency of 15 MHz:

By definition:

* + - * CPI is equal to the number of clock cycles required by the related functional unit to execute the instruction (EX stage).
      * IC*i* is the number of times an instruction is repeated in the referenced source code.
    - Recalculate the CPU performance equation assuming that you can triple the speed by just one unit of your choice between the FP multiplier or the FP divider:
      * FP multiplier unit: 6 🡪 2 clock cycles

*or*

* + - * FP divider unit: 30 🡪 10 clock cycles

Table 1: CPU time by hand

|  |  |  |  |
| --- | --- | --- | --- |
|  | Initial CPU time (a) | CPU time  (b – MUL speeded up) | CPU time  (b – DIV speeded up) |
| program\_1.s |  |  |  |

* + - Using the simulator, calculate the CPU time again and fill in the following table:

Table 2: CPU time using the simulator

|  |  |  |  |
| --- | --- | --- | --- |
|  | Initial CPU time (a) | CPU time  (b – MUL speeded up) | CPU time  (b – DIV speeded up) |
| program\_1.s |  |  |  |

Are there any differences? If so, where and why? If not, please provide some comments in the box below:

|  |
| --- |
| Your answer: |

* + - Using the simulator and the *Initial Configuration*, enable the Forwarding option and compute how many clock cycles the program takes to execute.

Table 3: forwarding enabled

|  |  |  |
| --- | --- | --- |
|  | Number of clock cycles | IPC (Instructions Per Clock) |
| program\_1.s |  |  |

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 4: **Program performance for different processor configurations**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Program | Forwarding | | Branch Target Buffer | | Delay Slot | | Forwarding + Branch Target Buffer | |
|  | IPC | CC | IPC | CC | IPC | CC | IPC | CC |
| program\_1.s |  |  |  | |  | |  | |

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



1. Using the program developed before: **program\_1.s**
2. Modify the processor architectural parameters related to multicycle instructions (Menu🡪Configure🡪Architecture) in the following way:
   * + - 1. Configuration 1

Starting from the *Initial Configuration*, change the FP addition latency to 3

* + - * 1. Configuration 2

Starting from the *Initial Configuration*, change the FP multiplier latency to 4

* + - * 1. Configuration 3

Starting from the *Initial Configuration*, change the FP division latency to 10

Compute both manually (using the Amdahl’s Law) and with the simulator the speed-up for any one of the previous processor configurations. Compare the obtained results and complete the following table.

Table 5: **program\_1.s speed-up computed by hand and by simulation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc. Config.**    **Speed-up comp.** | Initial config.  [c.c.] | Config. 1 | Config. 2 | Config. 3 |
| **By hand** |  |  |  |  |
| **By simulation** |  |  |  |  |

IMPORTANT NOTE: By now, you should have noticed that WinMIPS tends to differ from the notation provided during the lessons. One of these is that it tends to enter the execution state even if the operands are not yet available. This can affect the number of clock cycles required to execute the program since the instruction, entering the execution phase, frees up the decoding phase, which another instruction can then use.

1. Consider the *Initial Configuration*.

Then, assume that:

* + branch delay slot is not enabled
  + data forwarding is enabled
  + the EX stage could also be completed in an out-of-order fashion

1. Given the following source code, esteem the number of clock cycles (CC) needed for completion using notation given during the lessons (so, not using WinMIPS). In this regard, fill in the table on the next page with the pipeline stages for a single iteration of the loop using the same notation as the first two lines. Then, use the information obtained to fill in the table on this page.

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* C \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; for (i = 10; i > 0; i--) {

; v4[i] = (v1[i]+v1[i]+v2[i])/(v1[i]\*v2[i])

; }

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MIPS64 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

|  |  |  |
| --- | --- | --- |
| .data | Comments | Clock cycles |
| v1: .double 43, 23, 57, 34, 79, 6, 30, 44, 82, 18 |  |  |
| v2: .double 37, 12, 94, 59, 31, 77, 58, 21, 62, 1 |  |  |
| v3: .double 44, 94, 56, 67, 23, 78, 37, 14, 9, 52  v4: .space 80 |  |  |
|  |  |
|  |  |
| .text |  |  |
| daddui r1,r0,0 | r1 ← pointer |  |
| daddui r2,r0,10 | r2 ← 10 |  |
| cyc: l.d f1,v1(r1) | f1 ← v1[i] |  |
| l.d f2,v2(r1) | f2 ← v2[i] |  |
| l.d f3,v3(r1) | f3 ← v3[i] |  |
| add.d f5,f1,f1 | f5 ← v1[i]+v1[i] |  |
| mul.d f6,f1,f2 | f6 ← v1[i]\*v2[i] |  |
| add.d f5,f5,f2 | f5 ← f5 + v2[i] |  |
| div.d f7,f5,f6 | f7 ← f5/f6 |  |
| s.d f7,v4(r1) | v4[i] ← f7 |  |
| daddui r1,r1,8 | r1 ← r1+8 |  |
| daddi r2,r2,-1 | r2 ← r2-1 |  |
| bnez r2, cyc |  |  |
| halt |  |  |
|  |  |  |
| Total number of CC to run the entire program: |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| daddui r1,r0,0 | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| daddui r2,r0,10 |  | F | D | E | M | W |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f1,v1(r1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f2,v2(r1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f3,v3(r1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| mul.d f5,f1,f3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| div.d f6,f1,f2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| add.d f7,f5,f6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| s.d f7,v4(r1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| daddui r1,r1,8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| daddi r2,r2,-1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| bnez r2, cycle |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| halt |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

HINT: Use letters other than E for FP functional units or enter a subscript. For example, you can use EA for the FP adder, EM for the FP multiplier, and ED for the FP divider.

1. Repeat the same operations performed above assuming that data forwarding is disabled. If necessary, duplicate the table in the next page to have the necessary space to complete the required work.

|  |  |  |
| --- | --- | --- |
| .data | Comments | Clock cycles |
| v1: .double 43, 23, 57, 34, 79, 6, 30, 44, 82, 18 |  |  |
| v2: .double 37, 12, 94, 59, 31, 77, 58, 21, 62, 1 |  |  |
| v3: .double 44, 94, 56, 67, 23, 78, 37, 14, 9, 52  v4: .space 80 |  |  |
|  |  |
|  |  |
| .text |  |  |
| daddui r1,r0,0 | r1 ← pointer |  |
| daddui r2,r0,10 | r2 ← 10 |  |
| cyc: l.d f1,v1(r1) | f1 ← v1[i] |  |
| l.d f2,v2(r1) | f2 ← v2[i] |  |
| l.d f3,v3(r1) | f3 ← v3[i] |  |
| add.d f5,f1,f1 | f5 ← v1[i]+v1[i] |  |
| mul.d f6,f1,f2 | f6 ← v1[i]\*v2[i] |  |
| add.d f5,f5,f2 | f5 ← f5 + v2[i] |  |
| div.d f7,f5,f6 | f7 ← f5/f6 |  |
| s.d f7,v4(r1) | v4[i] ← f7 |  |
| daddui r1,r1,8 | r1 ← r1+8 |  |
| daddi r2,r2,-1 | r2 ← r2-1 |  |
| bnez r2, cyc |  |  |
| halt |  |  |
|  |  |  |
| Total number of CC to run the entire program: |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| daddui r1,r0,0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| daddui r2,r0,10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f1,v1(r1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f2,v2(r1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| l.d f3,v3(r1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| mul.d f5,f1,f3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| div.d f6,f1,f2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| add.d f7,f5,f6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| s.d f7,v4(r1) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| daddui r1,r1,8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| daddi r2,r2,-1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| bnez r2, cycle |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| halt |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |