INSTITUTO SUPERIOR TÉCNICO

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

COMPUTER ORGANIZATION

LEIC-A, LEIC-T

First Lab Assignment: Instruction Level Parallelism

Version 1.4

STUDENTS IDENTIFICATION:

Number:	**Name:**

2015/2016

1 Introduction

1.1 Objectives

The main purpose of this assignment is to provide the students with a straight and close contact to the operation of a pipelined computer architecture, as well as to the techniques that are commonly applied in order to maximize the efficiency of the developed computer programs. For that purpose, a dedicated simulation environment will be adopted: the WinMIPS64 [1], version 1.57.

1.2 Environment

WinMIPS64 is a simulator and a visual debugger of a subset of the MIPS64 Instruction Set Architecture (ISA). It was developed at Dublin City University School of Computing (Ireland) for educational purposes and it is capable of executing small programs that comply with the supported subset of the MIPS64 ISA. A detailed description of its operation is available in the provided user manual [2].

The main reason for adopting WinMIPS64 is its educational value in helping to understand the inner workings of pipelines. It provides a graphical interface that allows users to observe the execution of instructions through the multiple stages of the pipeline. Furthermore, the user has the capability of seeing how stalls are introduced and handled by the CPU, inspecting the status of registers and memory, and controlling step by step the execution of instructions.

However, WinMIPS64 is not fully compatible with the 32-bit architecture adopted in the textbook [3] (MIPS32). As a result, the instruction set used in this assignment is slightly different from the one presented in the textbook and in the theory classes. In particular, two main differences must be considered in order to properly understand the application program introduced in Section 1.3:

- *Registers:* in WinMIPS64, all registers have 64 bits in size. There are 32 integer registers (referred to as *$0-$31*) and 32 floating-point registers (referred to as *f0-f31*).
- *Instructions:* WinMIPS64 implements the operations using the integer instructions that depicted in the following table (see [2] for more details):

MIPS64 Instruction	**Description**	**MIPS32 Equivalent**
`daddi` *reg*, *reg*, *imm*	Add 64-bit immediate	`addi` *reg*, *reg*, *imm*
`dadd` *reg*, *reg*, *reg*	Add 64-bit integers	`add` *reg*, *reg*, *reg*
`dmul` *reg*, *reg*, *reg*	Multiply 64-bit integers	`mul` *reg*, *reg*, *reg*

1.3 Application program

A given mathematics library makes use of the following algorithm written in pseudo-code. This algorithm computes the compensated factorial of *N-1* and the Lucas number for $n = N$. The outcome values are stored in variables *fact* and *lucas*, respectively.

```
#define N 10
double A[N-1] = {0, 1, 3, 0, 7, 0, 1, 5, 3};
int64 x=2, y=1, lucas, fact=1, i=1;
double *pA = &A[0];

do {
  fact = fact * i + *pA;
  lucas=x+y; x=y; y=lucas;
  pA++;
  i++;
} while(i!=N);
```

Figure 1 lists the MIPS64 source code that implements this mathematical function (see file `prog.s`). To provide an example of how this function is used, the program initializes the data vectors with a few sampled values. Each value is represented with a 64-bit integer.

```
        .data
fact:   .word    0x0
lucas:  .word    0x0
A:      .word    0, 1, 3, 0, 7,
        .word    0, 1, 5, 3

        .code
        daddi    $1, $0, A      ; $1 = Index for A
        daddi    $2, $0, 2      ; $2 = 2 ;; x
        daddi    $3, $0, 1      ; $3 = 1 ;; y
        daddi    $5, $0, 1      ; $5 = 1 ;; i
        daddi    $6, $0, 10     ; $6 = N ;; N =10
        daddi    $12, $0, 1     ; $12 = 1 ;; fact=1

loop:   lw       $16, 0($1)     ; $16 = A[i-1]
        dmul     $12, $12, $5   ; $12 = fact *i
        dadd     $12, $12, $16  ; $12 = fact *i + A[i-1]

        dadd     $4, $2, $3     ; $4 = lucas = x + y
        daddi    $2, $3, 0      ;
        daddi    $3, $4, 0      ;

        daddi    $1, $1, 8      ; pA++
        daddi    $5, $5, 1      ; i++
        bne      $6, $5, loop   ; Exit loop if i == N

        sw       $12, fact($0)  ; Store factorial result
        sw       $4, lucas($0)  ; Store lucas result
        halt
```

Figure 1: Program source code.

2 Procedure

2.1 Simple execution, without data forwarding techniques

a) Download the source code file `prog.s` from the course webpage. Copy it into your working directory and initiate the WinMIPS64 simulator, by executing the program `winmips64.exe`[1].

The WinMIPS64 main window is composed of a menu bar and seven frames, showing different aspects of the simulation: Pipeline, Code, Data, Registers, Statistics, Cycles and Terminal. There is also a status bar to notify the user that the simulator is running as soon as the simulation has been started.

b) Configure the simulator, in order to prevent data forwarding, by making sure that the following check boxes are **unchecked**:

`Configure` → `Enable Forwarding`
`Configure` → `Enable Delay Slot`

c) Open the downloaded program, by pursuing the following steps:

`File` → `Open` → `prog.s`

[1] In a Linux environment, this program may be executed by making use of the 'wine' platform emulator and by issuing the command: `wine winmips64.exe`

d) Initiate the simulation, either by issuing the command:
`Execute → Run to` (shortkey = F4)
or by running the program by single-steps:
`Execute → Single Cycle` (shortkey = F7)

e) Select an arbitrarily loop iteration (avoid the first and the last ones) of the executed program. For each instruction of such iteration represent, in Table 1, the several executed stages of the pipeline: F, D, Xn, M, W. Do not forget to represent every *Stalls* that may occur.

f) Summarize the program execution profile, by filling the following form:

Clock cycles	
Instructions	
Average CPI	

Stalls:	- RAW	
	- Structural	
	- Branch Taken	

g) By analysing the program execution, characterize the branch prediction policy that is adopted by this simulator. Justify.

2.2 Application of data forwarding techniques

Although WinMIPS64 pipeline simulator attempts to mimic as far as possible the pipeline processor architecture that is described in Chapter 4 of [4], in certain aspects some alternative strategies have been implemented. One such example is observed with *Stalls*: they are handled where they arise in the pipeline, not necessarily in the ID stage. Several consequences arise from this strategy:

- floating-point instructions are allowed to issue out of ID into their own pipelines (if available), where they either proceed or stall, waiting for their operands to become available;
- instructions are allowed to complete out-of-order, leading to the introduction of WAR hazards;
- structural hazards often happen at the MEM stage bottleneck, as more than one instruction attempt to exit of their execute stage pipelines at the same time.

a) Configure the WinMIPS64 simulator in order to activate data forwarding, by making sure that the following check box is **checked**:
`Configure → Enable Forwarding`

b) Repeat the previous section procedure and represent, in Table 2, the execution of the same iteration of the program loop, by representing, for each instruction, the several executed stages of the pipeline: F, D, Xn, M, W. Do not forget to represent every *Stalls* that may occur.

c) Summarize the program execution profile, by filling the following form:

Clock cycles	
Instructions	
Average CPI	

Stalls:	- RAW	
	- Structural	
	- Branch Taken	

d) Evaluate the obtained *speedup*, when conpared with the base setup, considered in section 2.1.

2.3 Source code optimization: minimization of data and structural hazards

a) One common approach to reduce the still existing data and structural hazards is to apply re-order techniques [4] to the instruction sequence of the program. By keeping the simulator's data forwarding option asserted, analyze the time diagram of the previous section and apply the necessary re-ordering optimization techniques in order to minimize the Structural and RAW *Stalls*. Make sure that the resulting output is kept unchanged.

b) Represent, in Table 3, the execution of the selected iteration of the program loop, by representing, for each instruction, the several executed stages of the pipeline: F, D, Xn, M, W. Do not forget to represent every *Stalls* that may occur.

c) Summarize the program execution profile, by filling the following form:

Clock cycles	
Instructions	
Average CPI	

Stalls:	- RAW	
	- Structural	
	- Branch Taken	

d) Compute the obtained *speedup*, when conpared with the base setup, considered in section 2.1.

2.4 Source code optimization: loop unrolling

a) One approach that is usually adopted to reduce the control hazards is to apply loop unrolling techniques [4] to the program instruction sequence. By analysing the time diagram of the previous section, apply the loop unrolling technique in order to reduce, by a factor of 2, the amount of resulting control hazards.

b) Represent, in Table 4, the execution of the selected iteration of the program loop, by representing, for each instruction, the several executed stages of the pipeline: F, D, Xn, M, W. Do not forget to represent every *Stalls* that may occur.

c) Summarize the program execution profile, by filling the following form:

Clock cycles	
Instructions	
Average CPI	

Stalls:	- RAW	
	- Structural	
	- Branch Taken	

d) Compute the obtained *speedup*, when conpared with the base setup, considered in section 2.1.

2.5 Source code optimization: branch delay slot

a) One alternative approach that is frequently made available by most pipeline processor implementations to reduce the control hazards penalty is based on the usage of the *Branch Delay Slot* [4]. By analysing the time diagram of the program sequence considered in section 2.3, repeat the application of the re-order techniques to the program considered in section 2.3 in order to take advantage of the branch delay slot.

b) Before executing the modified file, configure the simulator in order to take advantage of the *Branch Delay Slot*, by making sure that the following check box is also **checked**:

`Configure` → `Enable Delay Slot`

c) Represent, in Table 5, the execution of the selected iteration of the program loop, by representing, for each instruction, the several executed stages of the pipeline: F, D, Xn, M, W. Do not forget to represent every *Stalls* that may occur.

d) Summarize the program execution profile, by filling the following form:

Clock cycles	
Instructions	
Average CPI	

Stalls:	- RAW	
	- Structural	
	- Branch Taken	

e) Compute the obtained *speedup*, when conpared with the base setup, considered in section 2.1.

References

[1] WinMIPS64. Webpage. "`http://indigo.ie/~mscott`", September 2013.

[2] Mike Scott. *WinMIPS64 Simple Tutorial*, 2008.

[3] John Hennessy and David Patterson. *Computer Architecture - A Quantitative Approach*. Morgan Kaufmann, 4^{th} edition, 2006.

[4] David Patterson and John Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann, 4^{th} edition, 2011.

Table 1: Pipeline time diagram, without data forwarding techniques.

	INSTRUCTIONS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
1																																																			
2																																																			
3																																																			
4																																																			
5																																																			
6																																																			
7																																																			
8																																																			
9																																																			
10																																																			
11																																																			
12																																																			
13																																																			
14																																																			
15																																																			
16																																																			
17																																																			
18																																																			
19																																																			
20																																																			
21																																																			
22																																																			
23																																																			
24																																																			
25																																																			
26																																																			
27																																																			
28																																																			
29																																																			
30																																																			

Table 2: Pipeline time diagram, with data forwarding techniques.

	INSTRUCTIONS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
1																																																			
2																																																			
3																																																			
4																																																			
5																																																			
6																																																			
7																																																			
8																																																			
9																																																			
10																																																			
11																																																			
12																																																			
13																																																			
14																																																			
15																																																			
16																																																			
17																																																			
18																																																			
19																																																			
20																																																			
21																																																			
22																																																			
23																																																			
24																																																			
25																																																			
26																																																			
27																																																			
28																																																			
29																																																			
30																																																			

Table 3: Pipeline time diagram, with minimization techniques to reduce the data and structural hazards.

	INSTRUCTIONS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
1																																																			
2																																																			
3																																																			
4																																																			
5																																																			
6																																																			
7																																																			
8																																																			
9																																																			
10																																																			
11																																																			
12																																																			
13																																																			
14																																																			
15																																																			
16																																																			
17																																																			
18																																																			
19																																																			
20																																																			
21																																																			
22																																																			
23																																																			
24																																																			
25																																																			
26																																																			
27																																																			
28																																																			
29																																																			
30																																																			

Table 4: Pipeline time diagram: usage of loop unrolling minimization techniques to reduce the control hazards.

	INSTRUCTIONS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
1																																																			
2																																																			
3																																																			
4																																																			
5																																																			
6																																																			
7																																																			
8																																																			
9																																																			
10																																																			
11																																																			
12																																																			
13																																																			
14																																																			
15																																																			
16																																																			
17																																																			
18																																																			
19																																																			
20																																																			
21																																																			
22																																																			
23																																																			
24																																																			
25																																																			
26																																																			
27																																																			
28																																																			
29																																																			
30																																																			

Table 5: Pipeline time diagram: usage of branch delay slot techniques to reduce the control hazards.

	INSTRUCTIONS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
1																																																			
2																																																			
3																																																			
4																																																			
5																																																			
6																																																			
7																																																			
8																																																			
9																																																			
10																																																			
11																																																			
12																																																			
13																																																			
14																																																			
15																																																			
16																																																			
17																																																			
18																																																			
19																																																			
20																																																			
21																																																			
22																																																			
23																																																			
24																																																			
25																																																			
26																																																			
27																																																			
28																																																			
29																																																			
30																																																			