Computer Architectures
Exam of 28.1.2020 - part I – ver. B1

First name, Last name, ID.....

Question #1

Let consider the Instruction Level Parallelism (ILP).
You are requested to

1. Explain what ILP is and why it is important in pipelined architectures
2. Define what static and dynamic instruction re-scheduling are, listing the advantages / disadvantages they involve
3. Summarize what Loop Unrolling is and list the advantages / disadvantages it involves
4. Provide an example of loop unrolling, writing a piece of code before and after its application.

Hints for answers

1. ILP stands for Instruction Level Parallelism. It is the amount of parallelism (i.e., independence) between instructions. It helps achieving a better CPI in pipelined architectures, since it allows different instructions to be executed in parallel. Hence, it is a characteristic of the code.
2. Re-scheduling is the activity of changing the order of instructions executed by the processor. This can be done
   1. statically, if it is done by the compiler. In this way the compiler becomes more complex, and the code is optimized for a given processor architecture. Since not all dependencies are known at compile time (e.g., those involving memory locations are not), the achievable performance is lower than for dynamic re-scheduling.
   2. Dynamic re-scheduling is implemented at run time by the processor, whose hardware clearly becomes more complex. However, the code is still portable and the achievable performance is higher.
3. Loop unrolling is a static technique for reorganizing the code when it includes loops. It involves repeating the body of a loop several time (if iterations are independent one from each other), thus reducing the number of iterations.
   In this way the performance may increase because
   - The number of branch instructions reduces
   - The size of the basic block corresponding to the iteration body increases, increasing the ILP.

   On the other size, this requires smarter compilers and may significantly increase the code size.

4.

```
        LD R6, R0, 10
LOOP:   LD R3, 0(R1)
        LD R4, 0(R2)
        DADDUI R5, R3,R4
        DADDUI R1, R1, 8
        DADDUI R2, R2, 8
        DADDUI R6,R6,-1
        BNEZ R6, LOOP
```

Unrolled:

```
        LD R6, R0, 5
LOOP:   LD R3, 0(R1)
        LD R4, 0(R2)
        DADDUI R5, R3,R4
        DADDUI R1, R1, 8
        DADDUI R2, R2, 8
        LD R3, 0(R1)
        LD R4, 0(R2)
        DADDUI R5, R3,R4
        DADDUI R1, R1, 8
        DADDUI R2, R2, 8
        DADDUI R6,R6,-1
        BNEZ R6, LOOP
```

Computer Architectures
Exam of 28.1.2020 - part I – ver. B1

First name, Last name, ID.....

Question #2

Let consider a MIPS64 architecture including the following functional units (for each unit the number of clock periods to complete one instruction is reported):

- Integer ALU: 1 clock period
- Data memory: 1 clock period
- FP arithmetic unit: 2 clock periods (pipelined)
- FP multiplier unit: 4 clock periods (pipelined)
- FP divider unit: 6 clock periods (unpipelined)

You should also assume that

- The branch delay slot corresponds to 1 clock cycle, and the branch delay slot is not enabled
- Data forwarding is enabled
- The EXE phase can be completed out-of-order.

You should consider the following code fragment and, using the table in the following page (where each column corresponds to a clock period), and determine the pipeline behavior in each clock period, as well as the total number of clock periods required to execute the fragment, reporting the result in the right column in the table below. The value of the constant k is written in f5 before the beginning of the code fragment.

```
; ********************* MIPS64 ***********************
;  for (i = 0; i < 10; i++) {
;       v4[i] = v1[i]/v2[i] - v3[i]^2;
;  }
```

```
                .data
V1:     .double "10 values"
V2:     .double "10 values"
V3:     .double "10 values"
V4:     .double "10 values"



                .text
main:  daddui r1,r0,0
       daddui r2,r0,10
loop:  l.d  f1,v1(r1)
       l.d  f2,v2(r1)
       l.d  f3,v3(r1)
       div.d  f4, f1, f2
       mul.d  f5,f3,f3
       sub.d f6, f4, f5
       s.d  f6,v4(r1)
       daddui  r1,r1,8
       daddi  r2,r2,-1
       bnez  r2, loop
       halt
```

	comments	Clock cycles
	r1← pointer	5
	r2 <= 20	1
	f1 <= v1[i]	1
	f2 <= v2[i]	1
	f3 <= v3[i]	1
	f4 <= v1[i] / v2[i]	6
	f5 <= v3[i]^2	0
	f7 <= v1[i] / v2[i] - v3[i]^2	2
	v4[i] <= f6	1
	r1 <= r1 + 8	1
	r2 <= r2 - 1	1
		2
		1
total		176

Computer Architectures
Exam of 28.1.2020 - part I – ver. B1

First name, Last name, ID.....

daddui r1,r0,0	F	D	E	M	W																										
daddui r2,r0,10		F	D	E	M	W																									
l.d f1,v1(r1)			F	D	E	M	W																								
l.d f2,v2(r1)				F	D	E	M	W																							
l.d f3,v3(r1)					F	D	E	M	W																						
div.d f4, f1, f2						F	D	E	E	E	E	E	E	M	W																
mul.d f5,f3,f3							F	D	E	E	E	E	M	W																	
sub.d f6, f4, f5									F	D				E	E	M	W													
s.d f6,v4(r1)										F				D		E	M	W												
daddui r1,r1,8														F		D	E	M	W											
daddi r2,r2,-1																F	D	E	M	W										
bnez r2, loop																	F		D	E	M	W								
halt																			F	D	E	M	W								