Computer Architectures
Exam of 18.6.2020 - part I

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

Question #1

The Tomasulo architecture for superscalar processors with dynamic scheduling and speculation uses reservation stations.
You are requested to

1. Explain what reservation stations are and where they are placed in the Tomasulo architecture, listing the modules they are connected to
2. Describe the hardware structure of a reservation station
3. Summarize when data/information are written/updated in a reservation station
4. explain the advantages stemming from the introduction of the reservation stations in the Tomasulo architecture.

Possible answer:

1. Reservation stations are hardware structures connected to execution units. They store instructions to be executed by them and waiting for operands. They are connected to the execution unit they relate to, to the Register File (where they fetch operands from), and to the Common Data Bus (CDB).
2. Each reservation station stores different fields
    - Op: stores the operation to be performed
    - Vj and Vk: values of the 2 operands, if available
    - Qj and Qk: identifiers of the instruction which will produce an operand; instructions are identified by the corresponding slot in the ROB
    - A: for load/store instructions, only. It stores the address
    - Busy: flag stating whether the reservation is free
3. A reservation station is first written by the issue unit, which allocates an instruction to it. If no reservation stations for that execution unit are available, a structural hazard occurs. If an operand is not yet available after the issue, the reservation station monitors the CDB until its value is produced. As soon as all operands are available, the instruction is sent to the execution unit, which executes it.
4. The reservation stations allow for an efficient out-of-order instruction execution, matching the data dependencies.

Computer Architectures
Exam of 18.6.2020 - part I

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, filling the following tables, determine the pipeline behavior in each clock period, as well as the total number of clock periods required to execute the fragment. The value of the constant k is written in f10 before the beginning of the code fragment.

```
; ********************* MIPS64 ***********************
;  for (i = 0; i < 10; i++) {
;       v5[i] = (v1[i]*v2[i]) + (v3[i]*v4[i])/k;
;  }
```

```
                .data
v1:     .double “10 values”
v2:     .double “10 values”
v3:     .double “10 values”
v4:     .double “10 values”
v5:     .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)
       l.d  f4,v4(r1)
       mul.d  f6,f1,f2
       mul.d  f7,f3,f4
       div.d  f8, f7, f10
       add.d f9, f6, f8
       s.d  f9,v5(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 <= v4[i]	1
f6 <= v1[i]*v2[i]	4
f7 <= v3[i]*v4[i]	1
f8 <= v3[i]*v4[i]/k	6
f9 <= v1[i]*v2[i] + v3[i]*v4[i]/k	2
v5[i] <= f9	1
r1 <= r1 + 8	1
r2 <= r2 - 1	1
	2
	1
total	236

Computer Architectures
Exam of 18.6.2020 - part I

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

main:	daddui r1,r0,0	F	D	E	M	W																									5
	daddui r2,r0,10		F	D	E	M	W																								1
loop:	l.d f1,v1(r1)			F	D	E	M	W																							1
	l.d f2,v2(r1)				F	D	E	M	W																						1
	l.d f3,v3(r1)					F	D	E	M	W																					1
	l.d f4,v4(r1)						F	D	E	M	W																				1
	mul.d f6,f1,f2							F	D	E	E	E	E	M	W																4
	mul.d f7,f3,f4								F	D	E	E	E	E	M	W															1
	div.d f8, f7, f10									F	D				E	E	E	E	E	E	M	W									6
	add.d f9, f6, f8										F	D									E	E	M	W							2
	s.d f9,v5(r1)											F									D	E		M	W						1
	daddui r1,r1,8																				F	D	E		M	W					1
	daddi r2,r2,-1																					F	D		E	M	W				1
	bnez r2,loop																						F			D	E	M	W			2
	halt																														1