

Considerare la seguente architettura MIPS64:

- Integer ALU: 1 clock cycle
- Data memory: 1 clock cycle
- FP multiplier unit: pipelined 6 stages
- FP arithmetic unit: pipelined 3 stages
- FP divider unit: not pipelined unit that requires 6 clock cycles
- branch delay slot: 1 clock cycle, and the branch delay slot disabilitato
- forwarding abilitato
- è possibile completare lo stage EXE di una istruzione in modo out-of-order.

- Facendo riferimento al frammento di codice riportato, si mostrino le tempistiche relative all'esecuzione ciascuna istruzione e si calcoli il numero totale di clock cycles necessari per eseguire completamente il programma.

```
for (i = 0; i < 100; i++) {
    v5[i] = (v1[i]/v2[i]) / v3[i] + v4[i];
}
```

[illegible]

B1

2305

Name, Student ID

Considerato un programma basato su loop, ed assumendo che il processore utilizzato sia un MIPS64 che implementa multiple-issue e speculation:

- Issue di 2 istruzioni per clock cycle
- Istruzioni jump richiedono 1 issue
- Esegui il commit di 2 istruzioni per clock cycle
- Le unità funzionali hanno le seguenti caratteristiche:
 - i. 1 Memory address 1 clock cycle
 - ii. 1 Integer ALU 1 clock cycle
 - iii. 1 Jump unit 1 clock cycle
 - iv. 1 FP multiplier unit, which is pipelined: 8 stages
 - v. 1 FP divider unit, which is not pipelined: 8 clock cycles
 - vi. 1 FP Arithmetic unit, which is pipelined: 4 stages
- La predizione di salto è sempre corretta
- Non ci sono cache misses
- Essitono 2 CDB (Common Data Bus).

○ Si complete la tabella mostrando il comportamento del processore durante le 3 iniziali iterazioni

# iterazione	Instruction	ISSUE	EXE	MEM	CDBx2	COMMITx2
1	l.d f1,v1(r1)	1	2	3	4	5
1	l.d f2,v2(r1)	1	3	4	5	6
1	mul.d f1,f1,f1	2	5	—	13	14
1	mul.d f2,f2,f2	2	6	—	14	15
1	div.d f5,f1,f2	3	15	—	23	24
1	s.d f5,v3(r1)	3	4	—	—	24
1	daddui r1,r1,-8	4	5	—	6	25
1	bnez r1,loop	5	7	—	—	25
2	l.d f1,v1(r1)	6	7	8	9	26
2	l.d f2,v2(r1)	6	8	9	10	26

3 Febbraio 2020 – ASE – Architetture moderne

B1

Name, Student ID

2	mul.d f1,f1,f1	7	10	—	18	27
2	mul.d f2,f2,f2	7	11	—	19	27
2	div.d f5,f1,f2	8	23	—	31	32
2	s.d f5,v3(r1)	8	9	—	—	32
2	daddui r1,r1,-8	9	10	—	11	33
2	bnez r1,loop	10	12	—	—	33
3	l.d f1,v1(r1)	11	12	13	14	34
3	l.d f2,v2(r1)	11	13	14	15	34
3	mul.d f1,f1,f1	12	15	—	23	35
3	mul.d f2,f2,f2	12	16	—	24	35
3	div.d f5,f1,f2	13	31	—	39	40
3	s.d f5,v3(r1)	13	14	—	—	40
3	daddui r1,r1,-8	14	15	—	16	41
3	bnez r1,loop	15	17	—	—	41