INSTITUTO SUPERIOR TÉCNICO

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

COMPUTER ORGANIZATION

LEIC-A, LEIC-T

Third Lab Assignment: Instruction Level Parallelism

Version 1.0

2023/2024

DUE IN CLASS

STUDENTS IDENTIFICATION:

Number:	**Name:**
99552	Rodrigo Dias
102848	Diogo Rodrigues
103156	Francisco Noel

2.1 Simple execution, without data forwarding techniques

e)

Clock cycles	18

Instructions	6

Average CPI	3,0

f)

Clock cycles	174
Instructions	61
Average CPI	2,852

Stalls: - Data	101
- Structural	0
- Branch Taken	8

g) A técnica usada é o static prediction not taken. Após dar fetch da instrução BNE, começa-se a dar fetch da instrução SW (durante a execução do branch), assumindo que o salto não é realizado. Caso o salto seja realizado, a instrução SW é anulada.

2.2 Application of data forwarding techniques

c)

Clock cycles	136
Instructions	61
Average CPI	2,230

Stalls: - Data	63
- Structural	9
- Branch Taken	8

d)

$$CR = \frac{\text{nº ciclos}}{t_{exec}} \qquad CPI = \frac{\text{nº ciclos}}{\text{nº inst}} \Rightarrow t = \frac{CPI \cdot \text{nº inst}}{CR}$$

$$\text{SPEED UP} = \frac{\frac{CPI_{2.1} \times \#inst_{2.1}}{CR_{2.1}}}{\frac{CPI_{2.2} \times \#inst_{2.2}}{CR_{2.2}}} = \frac{2,852}{2,230} = 1,2789$$

2.3 Source code optimization: minimization of data and structural hazards

a) Attach a copy of the new assembly program.

c)

Clock cycles	118
Instructions	61
Average CPI	1,934

Stalls: - Data	36
- Structural	9
- Branch Taken	8

d)

$$SPEED\ UP = \frac{\frac{CPI_{2.1} \times \#inst_{2.1}}{CR_{2.1}}}{\frac{CPI_{2.3} \times \#inst_{2.3}}{CR_{2.3}}} = \frac{2,852}{1,934} = 1,4747$$

2.4 Source code optimization: loop unrolling

a) Attach a copy of the new assembly program.

c)

Clock cycles	103
Instructions	43
Average CPI	2,395

Stalls:	- Data	81
	- Structural	9
	- Branch Taken	2 (8/4 = 2)

↘ fator de 4

d)

$$SPEED\ UP = \frac{t_{old}}{t_{new}} = \frac{\frac{CPI_{2.1} \times n^{o}\ inst_{2.1}}{CR}}{\frac{CPI_{2.4} \times n^{o}\ inst_{2.4}}{CR}} = \frac{2,852 \times 61}{2,395 \times 43} = 1,6893$$

2.5 Source code optimization: branch delay slot

a) Attach a copy of the new assembly program.

d)

Clock cycles	101
Instructions	61
Average CPI	1,656

Stalls:	- Data	27
	- Structural	9
	- Branch Taken	0

e)

$$SPEED\ UP = \frac{\frac{CPI_{2.1} \times \#inst_{2.1}}{CR_{2.1}}}{\frac{CPI_{2.5} \times \#inst_{2.5}}{CR_{2.5}}} = \frac{2,852}{1,656} = 1,7222$$

Table 1: 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
1	lw $12, 0($1)	F	D	E	M	W																																			
2	dmul $12, $12, $9		F	D	M_0	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M	W																											
3	dadd $9, $9, $12			F	D	D	E	E	E	E	E	E	E	M	W																											
4	daddi $5, $5, 1				F	F	D	D	D	D	D	D	D	E	M	W																										
5	daddi $1, $1, 8						F	F	F	F	F	F	F	D	E	M	W																									
6	bne $6, $5, loop													F	D	E	M	W																						
7	sw $9, mult($0)														F																									
8	lw $12, 0($1)															F	D	E	M	W																				
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 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
1	lw $12, 0($1)	F	D	E	M	W																																			
2	daddi $5, $5, 1		F	D	E	M	W																																		
3	dmul $12, $12, $9			F	D	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M	W																											
4	daddi $1, $1, $8				F	D	E	M	W																															
5	daddl $9, $9, $12					F	D	E	E	E	E	E	E	M	W																									
6	bne $6, $5, loop						F	D	D	D	D	D	D	E	M	W																								
7	sw $9, mult($0)							F	F	F	F	F	F																											
8	lw $12, 0($1)													F	D	E	M	W																						
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: usage of loop unrolling minimization techniques to reduce the control hazards.

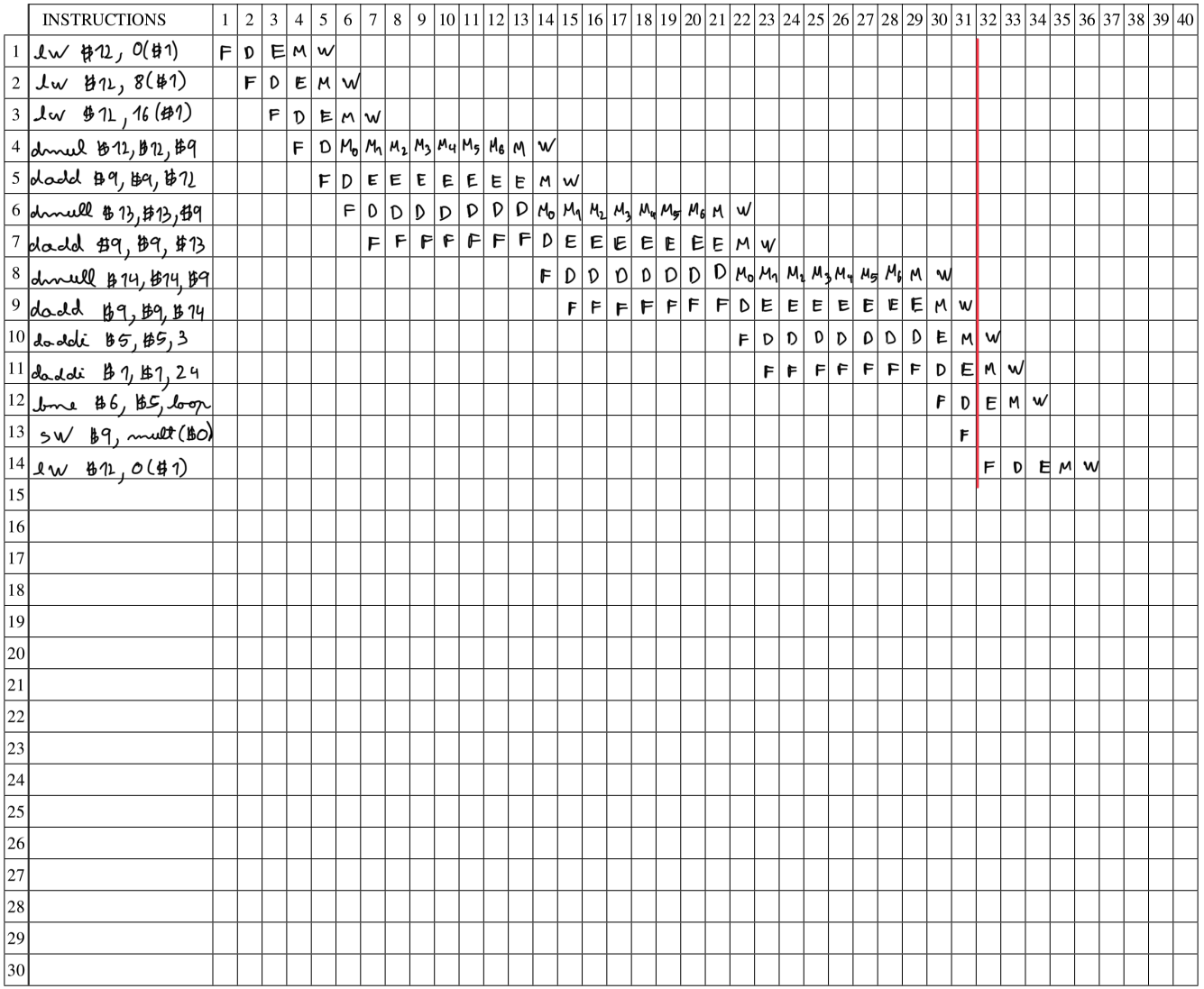

Table 4: Pipeline time diagram: usage of branch delay slot techniques to reduce the control hazards.

	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
1	lw $12, 0($1)	F	D	D	D	D	D	E	M	W																															
2	daddi $5, $5, 1		F	F	F	F	F	D	E	M	W																														
3	dmul $12, $12, $9							F	D	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M	W																								
4	daddi $1, $1, 8								F	D	E	M	W																												
5	bne $6, $5, loop									F	D	E	M	W																										
6	dadd $9, $9, $12										F	D	E	E	E	E	E	M	W																				
7	lw $12, 0($1)											F	D	D	D	D	D	E	M	W																			
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, 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
1	lw $12, 0($1)	F	D	X	M	W																																			
2	dmul $12,$12,$9		F	D	D	D	M0	M1	M2	M3	M4	M5	M6	M	W																										
3	dadd $9, $9, $12			F	F	F	D	D	D	D	D	D	D	D	D	E	M	W																						
4	daddi $5, $5, 1						F	F	F	F	F	F	F	F	F	D	E	M	W																						
5	daddi $7, $1, 8															F	D	E	M	W																				
6	bne $6, $5, loop																F	D	D	E	M	W																		
7	sw $9, mult($0)																	F	F																						
8	lw $12, 0($1)																			F	D	E	M	W																
9																																								
10																																								
11																																								
12																																								
13																																								
14																																								
15																																								
16																																								
17																																								
18																																								
19																																								
20																																								
21																																								
22																																								
23																																								
24																																								
25																																								
26																																								
27																																								
28																																								
29																																								
30																																								

2.3 a)

ASM 2.3.s

```
1                       .data
2   A:                  .word   1, 3, 1, 6, 4
3                       .word   2, 4, 3, 9, 5
4   mult:   .word   0
5
6           .code
7           daddi   $1, $0, A       ; *A[0]
8           daddi   $5, $0, 1       ; $5 = 1 ;; i
9           daddi   $6, $0, 10      ; $6 = N ;; N = 10
10          lw      $9, 0($1)       ; $9 = A[0]  ;; mult
11          daddi   $1, $1, 8       ;
12
13  loop:   lw      $12, 0($1)      ; $12 = A[i]
14          daddi   $5, $5, 1       ; i++
15          dmul    $12, $12, $9    ; $12 = $12*$9 ;; $12 = A[i]*mult
16          daddi   $1, $1, 8       ;
17          dadd    $9, $9, $12     ; $9 = $9 + $12  ;; mult = mult + A[i]*mult
18
19          bne     $6, $5, loop    ; Exit loop if i == N
20
21          sw      $9, mult($0)    ; Store result
22          halt
23
24  ;; Expected result: mult = f6180 (hex), 1008000 (dec)
25
26
```

2.4 a)

2.4.s

```
1                       .data
2   A:                  .word   1, 3, 1, 6, 4
3                       .word   2, 4, 3, 9, 5
4   mult:   .word   0
5
6           .code
7           daddi   $1, $0, A       ; *A[0]
8           daddi   $5, $0, 1       ; $5 = 1 ;; i
9           daddi   $6, $0, 10      ; $6 = N ;; N = 10
10          lw      $9, 0($1)       ; $9 = A[0]  ;; mult
11          daddi   $1, $1, 8       ;
12
13  loop:   lw      $12,  0($1)     ; $12 = A[i]
14          lw      $13,  8($1)     ; $13 = A[i+1]
15          lw      $14, 16($1)     ; $14 = A[i+2]
16
17          dmul    $12, $12,  $9   ; $12 = $12*$9 ;; $12 = A[i]*mult
18          dadd    $9, $9, $12     ; $9 = $9 + $12  ;; mult = mult + A[i]*mult
19
20          dmul    $13, $13, $9
21          dadd    $9, $9, $13
22
23          dmul    $14, $14, $9
24          dadd    $9, $9, $14
25
26
27
28          daddi   $5, $5, 3       ; i+=3
29          daddi   $1, $1, 24      ;
30
31
32
33
34          bne     $6, $5, loop    ; Exit loop if i == N
35
36          sw      $9, mult($0)    ; Store result
37          halt
38
39  ;; Expected result: mult = f6180 (hex), 1008000 (dec)
40
41
```

2.5 a)

2.5.s

```
.data
A:          .word   1, 3, 1, 6, 4
            .word   2, 4, 3, 9, 5
mult:       .word   0

            .code
            daddi   $1, $0, A       ; *A[0]
            daddi   $5, $0, 1       ; $5 = 1 ;; i
            daddi   $6, $0, 10      ; $6 = N ;; N = 10
            lw      $9, 0($1)       ; $9 = A[0]  ;; mult
            daddi   $1, $1, 8       ;

loop:       lw      $12, 0($1)      ; $12 = A[i]
            daddi   $5, $5, 1       ; i++
            dmul    $12, $12, $9    ; $12 = $12*$9 ;; $12 = A[i]*mult
            daddi   $1, $1, 8       ;

            bne     $6, $5, loop    ; Exit loop if i == N
            dadd    $9, $9, $12     ; $9 = $9 + $12  ;; mult = mult + A[i]*mult

            sw      $9, mult($0)    ; Store result
            halt

;; Expected result: mult = f6180 (hex), 1008000 (dec)
```