DUE IN CLASS

STUDENTS IDENTIFICATION:

Number:	**Name:**
102802	Fábio Mata
103392	Nuno Gonçalves
102664	Pedro Sousa

2.1 Simple execution, without data forwarding techniques

e)

Clock cycles	18	Instructions	7	Average CPI	$\frac{18}{7} = 2.571$

f)

Clock cycles	174
Instructions	61
Average CPI	2.852

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

g) A política de previsão de branch utilizada é "predict not taken", uma vez que após a instrução branch é executada a instrução seguinte do programa, e não a que consta no target do branch. Já que a instrução branch se situa no fim do loop, o jump vai ser feito em todas as iterações excepto na última, pelo que a instrução sw apenas não é anulada neste caso.

2.2 Application of data forwarding techniques

c)

Clock cycles	136
Instructions	61
Average CPI	2.230

Stalls:	- Data	63
	- Structural	9
	- Branch Taken	80

d)

$CC_{old} = 174$

$CC_{new} = 136$

$$\text{Speedup} = \frac{time_{old}}{time_{new}} = \frac{CC_{old} \times \cancel{Clock\text{-}Time}}{CC_{new} \times \cancel{Clock\text{-}Time}} = \frac{CC_{old}}{CC_{new}} = \frac{174}{136} \simeq 1.279$$

mesmo CPU

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)

$CC_{old} = 174$
$CC_{new} = 118$

$$\text{Speedup} = \frac{\text{Time}_{old}}{\text{Time}_{new}} = \frac{CC_{old} \times \cancel{\text{Clock-Time}}}{CC_{new} \times \cancel{\text{Clock-Time}}} \underset{\downarrow \text{ mesmo CPU}}{=} \frac{174}{118} \approx 1.475$$

2.4 Source code optimization: loop unrolling

a) Attach a copy of the new assembly program.

c)

Clock cycles	89
Instructions	43
Average CPI	2.070

Stalls: - Data	57
- Structural	9
- Branch Taken	2

d)

$CC_{old} = 174$
$CC_{new} = 89$

$$\text{Speedup} = \frac{\text{Time}_{old}}{\text{Time}_{new}} = \frac{CC_{old} \times \cancel{\text{Clock-Time}}}{CC_{new} \times \cancel{\text{Clock-Time}}} \underset{\downarrow \text{ mesmo CPU}}{=} \frac{174}{89} \approx 1.955$$

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)

$CC_{old} = 174$
$CC_{new} = 101$

$$\text{Speedup} = \frac{\text{Time}_{old}}{\text{Time}_{new}} = \frac{CC_{old} \times \cancel{\text{Clock-Time}}}{CC_{new} \times \cancel{\text{Clock-Time}}} \underset{\downarrow \text{ mesmo CPU}}{=} \frac{174}{101} \approx 1.723$$

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	X	M	W																																			
2	dmul $12,$12,$9		F	D	X	X	X	X	X	X	X	X	M	W																											
3	dadd $9,$9,$12			F	D	D	X	X	X	X	X	X	X	M	W																										
4	daddi $5,$5,1				F	F	D	D	D	D	D	D	D	X	M	W																									
5	daddi $1,$1,8						F	F	F	F	F	F	F	D	X	M	W																								
6	bne $6,$5,loop													F	D	X	M	W																							
7	sw $9,mult($0)														F																										
8	lw $12,0($1)															F	D	X	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	X	M	W																																			
2	daddi $5,$5 , 1		F	D	X	M	W																																		
3	dmul $12,$12,$9			F	D	X	X	X	X	X	X	X	M	W																											
4	daddi $1,$1, 8				F	D	X	M	W																																
5	dadd $9,$9,$12					F	D	X	X	X	X	X	X	M	W																										
6	bne $6, $5, loop						F	D	D	D	D	D	D	X	M	W																									
7	sw $9, mult ($0)							F	F	F	F	F	F																												
8	lw $12 ,0($1)													F	D	X	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.

	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	dmul $16, $12, $9	F	D	X	X	X	X	X	X	X	M	W																													
2	lw $13, 16($1)		F	D	X	M	W																																		
3	daddi $1, $1, 24			F	D	X	M	W																																	
4	lw $14, 0($1)				F	D	X	M	W																																
5	dadd $9, $9, $16					F	D	X	X	X	X	M	W																												
6	dmul $17, $13, $9						F	D	D	D	D	X	X	X	X	X	X	X	M	W																					
7	lw $12, 8($1)							F	F	F	F	D	X	M	W																										
8	dadd $9, $9, $17											F	D	X	X	X	X	X	X	M	W																				
9	dmul $18, $14, $9												F	D	D	D	D	D	D	X	X	X	X	X	X	X	M	W													
10	daddi $5, $5, 3													F	F	F	F	F	F	D	X	M	W																		
11	dadd $9, $9, $18																			F	D	X	X	X	X	X	X	M	W												
12	bne $6, $5, loop																				F	D	D	D	D	D	D	X	M	W											
13	sw $9, mult($0)																					F	F	F	F	F	F														
14	dmul $16, $12, $9																											F	D	X	X	X	X	X	X	X	M	W			
15																																									
16																																									
17																																									
18																																									
19																																									
20																																									
21																																									
22																																									
23																																									
24																																									
25																																									
26																																									
27																																									
28																																									
29																																									
30																																									

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	X	M	W																															
2	daddi $5,$5,1		F	F	F	F	F	D	X	M	W																														
3	dmul $12,$12,$9							F	D	X	X	X	X	X	X	X	M	W																							
4	daddi $1,$1,8								F	D	X	M	W																												
5	bne $6,$5,loop									F	D	X	M	W																											
6	dadd $9,$9,$12										F	D	X	X	X	X	X	M	W																						
7	lw $12, 0($1)											F	D	D	D	D	D	X	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	X	X	X	X	X	X	X	M	W																										
3	dadd $9, $9, $12			F	F	F	D	D	D	D	D	D	D	D	D	X	M	W																							
4	daddi $5, $5, 1						F	F	F	F	F	F	F	F	F	D	X	M	W																						
5	daddi $1, $1, 8															F	D	X	M	W																					
6	bne $6, $5, loop																F	D	D	X	M	W																			
7	sw $9, mult($0)																	F	F																						
8	lw $12, 0($1)																			F	D	X	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)

```
                .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       ;
        dadd    $9, $9, $12     ; $9 = $9 + $12  ;; mult = mult +
A[i]*mult

        bne     $6, $5, loop    ; Exit loop if i == N

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

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

2.4 a)

```
                .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[i - 1] ;; mult
        lw $12, 8($1)            ; $12 = A[i]

loop:
        dmul $16, $12, $9        ; $16 = $12*$9 ;; $16 = A[i]*mult
        lw $13, 16($1)           ; $13 = A[i + 1]
        daddi $1, $1, 24         ; $1 += 3
        lw $14, 0($1)            ; $14 = A[i - 1]
        dadd $9, $9, $16         ; $9 = $9 + $16 ;; mult = mult +
A[i]*mult

        dmul $17, $13, $9        ; $17 = $9*$13 ;; $17 = A[i]*mult
        lw $12, 8($1)            ; $12 = A[i]
        dadd $9, $9, $17         ; $9 = $9 + $17 ;; mult = mult +
A[i]*mult

        dmul $18, $14, $9        ; $18 = $14*$9 ;; $18 = A[i]*mult
        daddi $5, $5, 3          ; i += 3
        dadd $9, $9, $18         ; $9 = $9 + $18 ;; mult = mult +
A[i]*mult

        bne $6, $5, loop         ; Exit loop if i == N

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

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

2.5 a)

```
                .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)
```