DUE IN CLASS

STUDENTS IDENTIFICATION:

Number:	Name:
103124	Gonçalo Bórios
103969	Miguel Costa
102624	Raquel Braunschweig

2.1 Simple execution, without data forwarding techniques

e)

Clock cycles	18	Instructions	7	Average CPI	$\frac{18}{7} \approx 2{,}5714$

f)

Clock cycles	174
Instructions	61
Average CPI	2,852

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

g) A política utilizada é Predict Branch Not Taken, pois o simulador assume sempre que a branch não vai ser seguida (instrução "bne") e continua a dar fetch da próxima instrução ("sw"). Isto acontece todos os ciclos e só no último é que executa sem stalls a instrução "sw", sendo anulada em todos os outros ciclos ao se carregar a load instruction ("lw $12, 0($1)").

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)

$$\text{Speed Up} = \frac{CPU\_Time_{old}}{CPU\_Time_{new}} = \frac{Clock\_Cycles_{old} \times \cancel{Cycle\_Time}}{Clock\_Cycles_{new} \times \cancel{Cycle\_Time}} = \frac{174}{136} \approx 1{,}2794$$

• Dado que o CPU é o mesmo, o Cycle-Time vai ser o mesmo nas duas versões.

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)

$$\text{Speed Up} = \frac{CPU\_time_{old}}{CPU\_time_{new}} = \frac{Clock\_Cycles_{old} \times \cancel{Cycle\_Time}}{Clock\_Cycles_{new} \times \cancel{Cycle\_Time}} = \frac{174}{118} \approx 1{,}4746$$

• Dado que o CPU é o mesmo, o Cycle-Time vai ser o mesmo nas duas versões.

2.4 Source code optimization: loop unrolling

a) Attach a copy of the new assembly program.

c)

Clock cycles	89
Instructions	42
Average CPI	2,119

Stalls:	- Data	55
	- Structural	9
	- Branch Taken	2

d)

$$\text{Speed Up} = \frac{CPU\_time_{old}}{CPU\_time_{new}} = \frac{Clock\_Cycles_{old} \times \cancel{Cycle\_Time}}{Clock\_Cycles_{new} \times \cancel{Cycle\_Time}} = \frac{174}{89} \approx 1{,}9551$$

• Dado que o CPU é o mesmo, o Cycle-Time vai ser o mesmo nas duas versões.

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)

$$\text{Speed Up} = \frac{CPU\_time_{old}}{CPU\_time_{new}} = \frac{Clock\_Cycles_{old} \times \cancel{Cycle\_Time}}{Clock\_Cycles_{new} \times \cancel{Cycle\_Time}} = \frac{174}{101} \approx 1{,}7228$$

• Dado que o CPU é o mesmo, o Cycle-Time vai ser o mesmo nas duas versões.

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

nova iteração

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																																									

branch delay slot

nova iteração

Table 5: Pipeline time diagram, without data forwarding techniques.

nova → iteração

	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 = 1
        daddi   $6, $0, 10      ; $6 = N ;; N = 10
        lw      $9, 0($1)       ; $9 = A[0]  ;; mult
        daddi   $1, $1, 8       ; Set up for next word (A[1])

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       ; Set up for next word
        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                    ; Stop the program execution

;; 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 = 1
        daddi   $6, $0, 7       ; $6 = 7
        lw      $9, 0($1)       ; $9 = A[0]  ;; mult = A[0]
        lw      $12, 8($1)      ; $12 = A[1]
        lw      $13, 16($1)     ; $13 = A[2]

loop:   dmul    $22, $12, $9    ; $22 = $12*$9 ;; $22 = A[i]*mult
        daddi   $1, $1, 16      ; Set $1 for loading the next two words
        lw      $12, 8($1)      ; $12 = A[i+2] (doesn't interfere with dadd)
        dadd    $9, $9, $22     ; $9 = $9 + $22  ;; mult += A[i]*mult

        dmul    $23, $13, $9    ; $23 = $13*$9 ;; $23 = A[i+1]*mult
        daddi   $5, $5, 2       ; i += 2
        lw      $13, 16($1)     ; $13 = A[i+3] (doesn't interfere with dadd)
        dadd    $9, $9, $23     ; $9 = $9 + $23  ;; mult += A[i+1]*mult

        bne     $6, $5, loop    ; Exit loop if i == 7 (executes only three loops
                                ; to make sure we reduce by a factor of 4)

        ; 9 og iterations, so we are missing 3 (A[7], A[8] and A[9])
        dmul    $22, $12, $9    ; $22 = A[7]*mult
        lw      $14, 24($1)     ; $14 = A[9] (get last word)
        dadd    $9, $9, $22     ; mult += A[7]*mult

        dmul    $23, $13, $9    ; $23 = A[8]*mult
        dadd    $9, $9, $23     ; mult += A[8]*mult

        dmul    $24, $14, $9    ; $24 = A[9]*mult
        dadd    $9, $9, $24     ; mult += A[9]*mult (finally)

        sw      $9, mult($0)    ; Store result
        halt                    ; Stop the program execution

;; 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 = 1
        daddi   $6, $0, 10      ; $6 = N ;; N = 10
        lw      $9, 0($1)       ; $9 = A[0]  ;; mult
        daddi   $1, $1, 8       ; Set up for next word (A[1])

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       ; Set up for next word

        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                    ; Stop the program execution

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