#### **DUE IN CLASS**

#### STUDENTS IDENTIFICATION:

Number:	Name:
103124	Gorçalo Bários
103969	Mignel Costa
102624	Roquel Brownschuleig

### 2.1 Simple execution, without data forwarding techniques

e)	Clock cycles	18	Instructions	7	A	Average CPI	$\frac{18}{7} \approx 2,5714$
f)	Clock cycles	174		Stalls:	- Data	101	
	Instructions	61			- Structural	0	
	Aviana na CDI	2 952			D L. To	Iron 0	

A política utilizada é Tredict Branch Not Jaken, jois o xinulador arrume semple que a larch não vai rer requida (intrujão "line") e continua a dar feter da próxima instrujão ("nw"). Into orontece todos os ciclos e so no último e que executa sem stalla a instrujão "nw", sendo anulado em todos os outros ciclos ao re corregor a lood instruction ("lw \$12,0(\$1)").

#### 2.2 Application of data forwarding techniques

<b>c</b> )	Clock cycles	136	Stalls: - Data	63
	Instructions	61	- Structural	9
	Average CPI	2,230	- Branch Ta	ken 8

Speed 
$$U_{\mu} = \frac{\text{CPU-time old}}{\text{CPU-time new}} = \frac{\text{Clock-bycles old} \times \text{Cycle-time}}{\text{Clock-bycles new} \times \text{Cycle-time}} = \frac{174}{136} \approx 1,2794$$

Dodo que o CPU é o mesmo, o Cycle-time mai ser o mesmo nos duos mermos.

### 2.3 Source code optimization: minimization of data and structural hazards

a) Attach a copy of the new assembly program.

)	Clock cycles	118
	Instructions	61
	Average CPI	1,934

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

Speed Up = 
$$\frac{CPU - \text{tine old}}{CPU - \text{tine new}} = \frac{Clock - Cycles old \times Cycle - Time}{Clock - Cycles new \times Cycle - Time} = \frac{174}{118} \approx 1,4746$$

Dodo gue o CPU é o mesmo, o Cycle - Tine voi ser o mesmo nos dues versões.

#### 2.4 Source code optimization: loop unrolling

a) Attach a copy of the new assembly program.

<b>c</b> )	Clock cycles	89
	Instructions	42
	Average CPI	2,119

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

Speed Up = 
$$\frac{CPU - time old}{CPU - time new} = \frac{Clock - Cycles old \times Excles time}{Clock - Cycles new \times Cycle - Time} = \frac{174}{89} \approx 1,9551$$

Dodo que o  $CPU$  e' or meanor, o Cycle - Jine prai ser o mesnor mos duos versões.

#### 2.5 Source code optimization: branch delay slot

a) Attach a copy of the new assembly program.

<b>d</b> )	Clock cycles	101
	Instructions	61
	Average CPI	1,656

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

Speed 
$$4 = \frac{CPU\_time dd}{CPU\_time new} = \frac{Clock\_Gycles dd × Gycle\_time}{Clock\_Gycles_new} \times Gycle\_time}{Clock\_Gycles_new} \times Gycle\_time} = \frac{174}{101} \approx 1,7228$$

• Dado que o CPU l'o mesno, o Gycle—Time voi ver o mesno vos duos vervões.

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 3	36	37 3	8 3	19 4	10
	1			D					Ė					-		-				-	-																					
	2	donal \$12,\$12,\$9	Ė	F	D	Χ	X	x	Χ	Х	χ	Χ	χ	M	W																											
	3	dodd \$9, \$9,\$12		ľ	PF	0	D	X	χ	X	χ	Х	Χ	Χ	M	w																										
	4	1.11: \$5 55.1				F	F	0	D	D	D	D	х х Д	D	X	M	W																									
	5	daddi \$1, \$1, 8 lne \$6, \$5, loop M \$9, mult (\$0) lw \$12, 0 (\$1)						F	F	F	F	F	X D F	F	D	χ	M	W																								
	6	lone \$6, \$5, loop													F	D	X	M	W																							
	7	M \$9, mult (60)														F																										
_	8	lw \$12,0(\$1)															F	D	X	Μ	W																				_	
8	9																																							4		
	10																																								-	
	11																																							-		
	12		_	_				L			_																															
	13		-	-	_	-		-	_	-	_																						_	-								
	14		-	-	-	-	-	-		-																								-		-		-	-	-		
	15		-	-	-	-	-		-	-	-	-		-																			-	-	-	-						
	16		-	-		-	-	-	-	-	-			-							-		-		_					-												
	17		-	-	-	-	-	-	-	+	-					-	-	-															-			-						
	18		-	-	-	-	+	-	-		-	-				-		-																					-			
	19		-	-	-	+	+	-	-		-	-																														
	20 21		-	-	-	+	+-	H	-	+	-																															
	21		-	+	-	+	+	-	-	-	-																												-			
	23		-	-	-	+	-	+			-																															
	24		-	-	-	+	-	-		-																																
	25		-	t		+		$\vdash$	-		-																															
	26				-	-	-	-																																		
	27																																									
	28																																									
	29																																									
	30					l																																				
		1				1		_				-				-			-	-		-		-																		

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 3	38 3	39	40
1		F	D	X	M	W																																			
3	doddi \$5, \$5, 1		F	D	X	M	W																																		
				F	D	X	X	X			X	X	M	W																											
4	111. 44 44 0				F	D	X	M	W																																
5	dodd \$9, \$9, \$12					F	D	X	X	X	X	X	X	M	W																										
6	hre \$6, \$5, loop						F	D	0	D	0	0	0	χ	M	W																									
7	dold \$9, \$9, \$12 line \$6, \$5, loop NW \$9, mult (\$0)							F	F	F	X D F	F	F																												
8	lw \$12,0 (\$1)											•		F	0	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																																									
1											_									-	-					-		-		-		-	-		_				-	-	

**Table 3:** Pipeline time diagram: usage of loop unrolling minimization techniques to reduce the control hazards.

																			_			,		,				_		_	_	_										
	INS	TRUCTIONS	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	dru	? \$22, \$12, \$9 \$1, \$1, 16 \$12, 8(\$1)	r	U	X	Х	χ	X	X	X	Χ	W	W									1							-													
2	doddi	\$1, \$1, 16		F	D	Χ	M	W																																		
3	lw	\$12, 8(\$1)			F	D	χ	W	W																																	
4	11 11					F	D	X	X	Χ	X	X	W	W																												
5	down	\$23,\$13,\$9					F	D	D	D	D	D	X	X	X	χ	X	X	X	M	W																					
6	doddi	\$5, \$5, 2						F	F	F	F	F	0	X	M	W																										
7	low	\$13, 16 (\$1)											F	0	χ	M	W																									
8	dodd	\$9, \$9, \$23												F	DF	X	X	Χ	X	M	M	W																				
9	bol	\$6, \$5, 600													F	D	p	D	D	0	Χ	M	W																			
10	Inul	\$3, \$13, \$9 \$5, \$5, 2 \$13, 16 (\$1) \$9, \$9, \$23 \$6, \$5, log- \$22, \$12, \$9														F	F	F	F	F		M																				
-111	doul	\$22, \$12, \$9																			F	D	Χ	Х	Χ	χ	X	X	X	M	W											
12	-																																									
13																																										
14																																										
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.

	INSTRUCTIO	ONS	1	2	3 4	5	6	7	8	9	10	11	12 1	3 14	4 1	5 1	6 1	7 1	8 1	9 20	) 21	22	23	24 2	25 20	5 27	28	29	30 3	1 32	33	34 3	5 3	6 37	38	39	40
	1 0 \$13	0 ( \$4)	F	0	2 0	D	n	X	M I	w			-	-	1				-		-	-			+										-		
	2 doddi \$5,\$ 3 dmul \$12,4	5 1	F	EI		E	E	0	ΧI	M	M/	+	+		t	+	+		+							+											
	3 doub \$12.5	117 \$9		1 1	-	Γ	Г	F	n	·',	Ψ ∨ .	_	x )	/ >	( )	Y N	1 1/	N	+	+			1														
	3 dmul P1K14	4 0						1		0				'	'	١,,	, 4		+	+	-		-		+	+-						-					
land delay	doddi Di, D	1, 0	+	-	+					0 F	x i	ν\ -	MV		+	+	-	+	+	+	-	H	-		+	+			-	+				+	-		
Not	4 doddi \$1,\$ 5 fine \$6,\$! 3 dodd \$9,\$	A \$10			+	-		-	-	Г	F	^	A A			, ,		۸ ۱.	,	+					+						-						
	0 0000 111 4	(14)	-		+	+-		-	+	-	-	<i>U</i>	X X	N N	1	) (	\ ''	· M	A M		+-			-	+	+			-				+	+		-	
kronk delaj vlet nere — terojojo	→ 1 km \$12, 0	(\$1)		+	+	+			+	+	-	r	UL	ענ	l	) [	/ ^	1		V	+			+	-	+		-		+				+	+		
Jaragoo	8				+	+		-		+	-	+	-	+	+	+	+	+	-	+	+		-	-					+	+	+		+	-			
			-	-	+	+	H		+	-	+	+		+	+	+	+	+	+	+	+-		-	+		+		+					+				
	10				+	+	H		+	+	-	+	-	+-	+	+	+-	+	+	+	-		-		-	+	$\vdash$						+			-	
	11			-	+	+			+	+	+	-	-	+	+	+	+-	+-	+	+	+		+	+	+			+		+	+	-	+	t	-		-
	12		+-	-	+	+			-	-	+	+		+-	+	+	+	+	+	+	+		+	-	+	-	Н		+	+	+		+			+	
	13		-		-	+	-		+	-	+	-		+	+	+	+	+	+	+	+		+	+	+	+		+			+						-
	14		-		+	+			+	-	-	-	-	+	+	+	+	+	+	+			+	+	+	+		-	+	-	+-		+	-	+	+	-
	15		-	$\vdash$	+	-			+	-	+	-	-	+	+	+	+	+	+	+	+		+	+	+				-	+	+		+	+	+		
	16		-		+-	-	-		+	+	+	-	-	+	+	+	+	+	+	+	+		-	+	+					+	+		+		+	+	
	17		+		+	+			+	+	+	+	-	+	+	+	+	+	+	+				+					+		+-				+		
	18		+			+	-		+	-	-	+	-	+	+	+	+	+	+	+	+		+	+		+		+	+	-			+	+		+	
	19		-		+	+			+	+	+	-	+	+	+	+	+-	+	+	+	+		+		+	+		+		+			+			+	
	20		-		+	+	-		+	-	+	+	-	+	+	-	+	+	+	+	+		+	+	-	+			+	+			+	+	+	-	
	21		+-	-	+	+	-		+	-	-	-	-	+	+	+	+	+	+	+	+				-	-	H	-	-	-	+						-
	22					-	-		+	-	-	-	-	+	+	+	+		+	+	+		-			-			-	-	+	+	+	+			
	23				-	-	-		+	-	-	-		+	+	+	+	+	+	+	+		-	+	+	+			-	-			-		+		
	24				-	-			-		-		-	-	+	-	+	+	+	+	+-	H	-	-	-	+			+	+	+		+				-
	25				-		-		-		-	-	-	-	+	+	+	+	+	+			-		-	+	H		-	-	+	-	-				-
	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	1 15	5 10	6 17	7 1	8 1	9 2	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	М	w																																		_		
	2	doub \$12, \$12, 59		F	0	0	0	χ	Χ	X	X	X	X	X	M	W	1																								4		
	3	dodd \$9,\$9,\$12			F	F	F	0	D	D	0	D	0	0	D	0	X	N	W																-								
	4	doddi \$5,\$5,1						F	F	F	F	F	F	X D F	F	F	D	X	M	W	1																				-	-	
	5	doddi 51,51,8															F	U	X	N	\ W																				-	-	
	6	line \$6,\$5, loof														L		F	D	0	) X	1	N	W						-											-		
	7	line \$6,\$5,loop  w \$9, mult (\$0)																	F	F															-	_				-	-	-	
neva -	8	lw \$12, 0(\$1)																		ļ.,	F	[	) ;	X I	M	N													$\vdash$	-	+	-	
rova → iterosão	9																											_		_										+	-	-	
	10																				-	-							-				-							-	-	-	
	11																		-	-		L			+		+	-		4	-			-					-	-	+	+	
	12																_			_	L	-	-	-		+		-		-						-		-	-	+	+	+	
	13																-		-		-		+	+	-	-	-	-	-	-	-	-	-						-	+	-+	+	
	14																				-	-	+	-	+	-	+	+	-				-	-					+	-	+	+	
	15															-			-	-	-		+	-	+	+	+	-	-	-	+	-							+	+	+	+	
	16										_									-	-	H	+	+	+		-	+	+		+	-							+	+	+	+	
	17																		-		-	ŀ	+	+	+	+	+	+		-									+	+		+	
	18				_											_			-		-	-	÷	+	+	+	-		-	+	+	-	-	-					+	+	+	+	
	19			-	_			-		-	-		_					-				-	+	-				+	+	+	+		+							+		+	
	20		_					-		-												ŀ	H	+			+	-	+	+	+	+								+	+	+	
	21				_		-	-	-		.,		-										ŀ	+	+	+	+	+	+	+	+								-	+	+	+	
	22		-	_			-	-	-				-	-		_					-	-	H	+	+	+	+	+	+	+	+						+		+			+	
	23		-					-	-	+	-	-		-		-	-						ŀ	+	+	+		+		+	+	+								+		+	
	24		-	-	-	-	-	+	-	-		-	-		-								+	÷	+	+	+	+		+	+	+									-	+	
	25			4	-			-	+	+	-	-			-					-			H	+	+	+	+	+	+	+	+	+								+		+	
	26		-	-	-		-		-	-	-	-	-	+	-									+	+	+		+	+	+			+	+									
	27		-	-		-			+		-	-		+	-	-							H	+	+	+		+			+	+					+				+	+	
	28		-	-		-	-	+	-	-	-	-	-	+		-							-	+	+	+					+	+		-			+		+				
	29		-			+	-	+	-	+	-	-	-	-		-							-	+	+	+	+	+	+	+	+	+							+	+	+	+	
	30																																								_		

# 2.3 a)

```
.data
                  1, 3, 1, 6, 4
2 A:
          .word
          .word
                  2, 4, 3, 9, 5
4 mult:
          .word
          .code
6
          daddi
                   $1, $0, A
                               ; *A[0]
                                  ; $5 = 1 ;; i = 1
          daddi
                   $5, $0, 1
          daddi
                   $6, $0, 10
                                  ; $6 = N ; N = 10
                   $9, 0($1)
                                  ; $9 = A[0] ;; mult
          1w
10
          daddi
                   $1, $1, 8
                                 ; Set up for next word (A[1])
11
12
                   $12, 0($1)
                                 ; $12 = A[i]
          lw
13 loop:
14
          daddi
                   $5, $5, 1
                                  ; i++
                                  ; $12 = $12*$9 ;; $12 = A[i]*mult
          dmul
                   $12, $12, $9
16
          daddi
                   $1, $1, 8
                                  ; Set up for next word
17
          dadd
                   $9, $9, $12
                                  ; $9 = $9 + $12 ;; mult = mult + A[i]*mult
18
          bne
                   $6, $5, loop
                                ; Exit loop if i == N
                   $9, mult($0)
                                  ; Store result
          SW
21
          halt
                                  ; Stop the program execution
22
24 ;; Expected result: mult = f6180 (hex), 1008000 (dec)
```

## 2.4 a)

```
.data
                   1, 3, 1, 6, 4
           .word
2 A:
           .word
                   2, 4, 3, 9, 5
4 mult:
           .word
           .code
6
           daddi
                   $1, $0, A
                                  ; *A[0]
           daddi
                                    ; $5 = 1 ; ; i = 1
                   $5, $0, 1
                   $6, $0, 7
           daddi
                                    $6 = 7
9
                   $9, 0($1)
                                    ; $9 = A[0] ;; mult = A[0]
           lw
10
           lw
                                    ; $12 = A[1]
                   $12, 8($1)
11
           lw
                   $13, 16($1)
                                    ; $13 = A[2]
12
14 loop:
          dmul
                   $22, $12, $9
                                   ; $22 = $12*$9 ;; $22 = A[i]*mult
           daddi
                                    ; Set $1 for loading the next two words
                   $1, $1, 16
           1w
                   $12, 8($1)
                                    ; $12 = A[i+2] (doesn't interfere with dadd)
16
                                    ; $9 = $9 + $22 ;; mult += A[i]*mult
           dadd
                   $9, $9, $22
17
18
           dmul
                                   ; $23 = $13*$9 ;; $23 = A[i+1]*mult
                   $23, $13, $9
19
           daddi
                   $5, $5, 2
                                    ; i += 2
20
                   $13, 16($1)
                                    ; $13 = A[i+3] (doesn't interfere with dadd)
           lw
21
           dadd
                   $9, $9, $23
                                    ; $9 = $9 + $23 ;; mult += A[i+1]*mult
23
           bne
                   $6, $5, loop
                                   ; Exit loop if i == 7 (executes only three loops
24
                                    ; to make sure we reduce by a factor of 4)
25
26
           ; 9 og iterations, so we are missing 3 (A[7], A[8] and A[9])
27
                   $22, $12, $9
                                   ; $22 = A[7]*mult
           dmul
           1w
                   $14, 24($1)
                                    ; $14 = A[9] (get last word)
29
           dadd
                   $9, $9, $22
                                    ; mult += A[7]*mult
30
31
                   $23, $13, $9
                                    ; $23 = A[8]*mult
           dmul
           dadd
                   $9, $9, $23
                                   ; mult += A[8]*mult
33
34
           dmul
                   $24, $14, $9
                                   ; $24 = A[9]*mult
           dadd
                   $9, $9, $24
                                    ; mult += A[9]*mult (finally)
36
37
                   $9, mult($0)
                                    ; Store result
           SW
38
                                    ; Stop the program execution
           halt
39
40
41 ;; Expected result: mult = f6180 (hex), 1008000 (dec)
```

# 2.5 a)

```
.data
                  1, 3, 1, 6, 4
2 A:
          .word
          .word
                  2, 4, 3, 9, 5
4 mult:
          .word
          .code
6
          daddi
                  $1, $0, A
                               ; *A[0]
                                 ; $5 = 1 ;; i = 1
          daddi
                  $5, $0, 1
          daddi
                  $6, $0, 10
                                  ; $6 = N ; N = 10
                  $9, 0($1)
                                 ; $9 = A[0] ; mult
          1w
10
          daddi
                  $1, $1, 8
                                ; Set up for next word (A[1])
11
12
                  $12, 0($1)
                                 ; $12 = A[i]
          lw
13 loop:
14
          daddi
                  $5, $5, 1
                                 ; i++
                                 ; 12 = 12*9 ;; 12 = A[i]*mult
          dmul
                  $12, $12, $9
16
          daddi
                  $1, $1, 8
                                  ; Set up for next word
17
18
                  $6, $5, loop
                                ; Exit loop if i == N
          bne
          dadd
                  $9, $9, $12
                                 ; $9 = $9 + $12 ;; mult = mult + A[i]*mult
                  $9, mult($0)
                                 ; Store result
          SW
21
          halt
                                  ; Stop the program execution
22
24 ;; Expected result: mult = f6180 (hex), 1008000 (dec)
```