STUDENTS IDENTIFICATION:

Number:	Name:
109474	Daniel Borges
110181	Duarte Cruz
110728	João Tiago Gama

2.1 Simple execution, without data forwarding techniques

d)

Clock cycles	308
Instructions	170
Average CPI	1.81

Stalls: - Data	104
- Structural	0
- Branch Taken	30

e)

The branch prediction policy adopted by the simulator is a static "not taken" prediction. This means that all branch instructions are assumed not to be taken during the fetch stage. This behavior is evidenced by the fact that the program continues fetching instructions sequentially after a branch, and only when the branch is actually taken does the pipeline experience a stall due to the need to flush and redirect the instruction flow.

2.2 Application of data forwarding techniques

c)

Clock cycles	222
Instructions	170
Average CPI	1.31

Stalls: - Data	18
- Structural	0
- Branch Taken	30

d) The obtained speedup can be computed as:

$$\text{Speedup} = \frac{T_{\text{old}}}{T_{\text{new}}} = \frac{(\frac{1}{f_{\text{old}}}) \times \#\text{Cycles}_{\text{old}}}{(\frac{1}{f_{\text{new}}}) \times \#\text{Cycles}_{\text{new}}} = \frac{f_{\text{new}}}{f_{\text{old}}} \times \frac{\#\text{Cycles}_{\text{old}}}{\#\text{Cycles}_{\text{new}}}$$

Given that:

$$\#\text{Cycles}_{\text{old}} = 308, \#\text{Cycles}_{\text{new}} = 222$$

we obtain:

$$\frac{\#\text{Cycles}_{\text{old}}}{\#\text{Cycles}_{\text{new}}} = 1.38739$$

If both processors operate at the same clock frequency ($f_{\text{new}} = f_{\text{old}}$), then:

$$\text{Speedup} = 1.38739$$

However, since the second processor includes data forwarding, its critical path is longer due to the additional forwarding logic, potentially reducing its maximum achievable clock frequency. Consequently, the actual speedup in practice will likely be lower than 1.38739.

It is not possible to precisely determine the real speedup without information about the respective clock frequencies.

2.3 Source code optimization: minimization of data and structural hazards

a) Attach a copy of the new assembly program.

c)

Clock cycles	206
Instructions	170
Average CPI	1.21

Stalls: - Data	2
- Structural	0
- Branch Taken	30

d) The obtained speedup can be calculated as follows:

$$\text{Speedup} = \frac{T_{\text{old}}}{T_{\text{new}}} = \frac{(\frac{1}{f_{\text{old}}}) \times \#\text{Cycles}_{\text{old}}}{(\frac{1}{f_{\text{new}}}) \times \#\text{Cycles}_{\text{new}}} = \frac{f_{\text{new}}}{f_{\text{old}}} \times \frac{\#\text{Cycles}_{\text{old}}}{\#\text{Cycles}_{\text{new}}}$$

Given that:

$$\#\text{Cycles}_{\text{old}} = 308, \#\text{Cycles}_{\text{new}} = 206$$

we have:

$$\frac{\#\text{Cycles}_{\text{old}}}{\#\text{Cycles}_{\text{new}}} = 1.49515$$

If both processors operate at the same clock frequency ($f_{\text{new}} = f_{\text{old}}$), then:

$$\text{Speedup} = 1.49515$$

However, since the improved processor includes data forwarding, additional combinational logic is introduced in the execution path, thereby increasing the critical path length. This typically reduces the maximum attainable clock frequency.

Consequently, although the ideal speedup (assuming equal clock rates) is 1.495, the effective speedup in practice will likely be somewhat lower, depending on the impact of forwarding logic on the cycle time.

Because the simulator does not provide explicit clock frequency information, it is not possible to determine the precise speedup considering the real hardware timing constraints.

2.4 Source code optimization: loop unrolling

a) Attach a copy of the new assembly program.

c)

Clock cycles	122
Instructions	110
Average CPI	1.11

Stalls:	- Data	2
	- Structural	0
	- Branch Taken	6

d) The obtained speedup can be expressed as:

$$\text{Speedup} = \frac{T_{\text{old}}}{T_{\text{new}}} = \frac{(\frac{1}{f_{\text{old}}}) \times \#\text{Cycles}_{\text{old}}}{(\frac{1}{f_{\text{new}}}) \times \#\text{Cycles}_{\text{new}}} = \frac{f_{\text{new}}}{f_{\text{old}}} \times \frac{\#\text{Cycles}_{\text{old}}}{\#\text{Cycles}_{\text{new}}}$$

Given that:

$$\#\text{Cycles}_{\text{old}} = 308, \#\text{Cycles}_{\text{new}} = 122$$

then:

$$\frac{\#\text{Cycles}_{\text{old}}}{\#\text{Cycles}_{\text{new}}} = 2.52459$$

Assuming both processors operate at the same clock frequency ($f_{\text{new}} = f_{\text{old}}$), the ideal speedup is:

$$\text{Speedup} = 2.52459$$

However, the improved processor implements data forwarding, which introduces additional combinational logic in the datapath. This increases the critical path delay, thereby potentially reducing the maximum achievable clock frequency.

As a result, the effective speedup in a real hardware implementation would be lower than 2.52459, since the longer critical path would require a slower clock to maintain correct operation.

Because the simulator does not model or expose timing details such as the actual clock frequency, it is not possible to accurately quantify the true performance gain beyond the ideal cycle-based estimation.

Anexo 1: Assembly Code 2.3 a)

```
    .data
A:  .word 1,2,3,4,5,6,7,8,9
    .word 10,11,12,13,14,15,16
B:  .word 11,22,33,44,55,66,77
    .word 88,99,111,122,133
    .word 144,155,166
C:  .word 0,0,0,0,0,0,0,0,0
    .word 0,0,0,0,0,0,0

    .text
main:
    la s3, A          # base of A
    la s4, B          # base of B
    la s5, C          # base of C
    addi s1, zero, 0   # i = 0
    addi s2, zero, 16  # value of N

loop:
    lw t1, 0(s3)
    lw t2, 0(s4)
    addi s1, s1, 1
    addi s3, s3, 4
    mul t3, t2, t1
    addi s4, s4, 4
    add t3, t3, t1
    sw t3, 0(s5)
    addi s5, s5, 4
    bne s1, s2, loop

end:
    addi a7, zero, 10
    ecall           # Exit (syscall)
```

Anexo 2: Assembly Code 2.4a)

```
    .data
A:  .word 1,2,3,4,5,6,7,8,9
    .word 10,11,12,13,14,15,16
B:  .word 11,22,33,44,55,66,77
    .word 88,99,111,122,133
    .word 144,155,166
C:  .word 0,0,0,0,0,0,0,0,0
    .word 0,0,0,0,0,0,0

    .text
main:
    la s3, A          # base of A
    la s4, B          # base of B
    la s5, C          # base of C
    addi s1, zero, 0   # i = 0
    addi s2, zero, 16  # value of N

loop:
    lw t2, 0(s4)
    lw t1, 0(s3)
    addi t3, t2, 1
    mul t3, t3, t1
    sw t3, 0(s5)

    lw t2, 4(s4)
    lw t1, 4(s3)
    addi t3, t2, 1
    mul t3, t3, t1
    sw t3, 4(s5)

    lw t2, 8(s4)
    lw t1, 8(s3)
    addi t3, t2, 1
    mul t3, t3, t1
    sw t3, 8(s5)

    lw t2, 12(s4)
    lw t1, 12(s3)
    addi t3, t2, 1
    mul t3, t3, t1
    sw t3, 12(s5)

    addi s1, s1, 4
    addi s3, s3, 16
    addi s4, s4, 16
    addi s5, s5, 16
    bne s1, s2, loop

end:
    addi a7, zero, 10
    ecall           # Exit (syscall)
```

Table 1: 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 x6 0 x19	F	D	X	M	W														F	D	X	M	W																	
2	lw x7 0 x20		F	D	X	M	W																																		
3	mul x28 x7 x6			F	D	-	-	X	M	W																															
4	add x28 x28 x6				F	-	-	D	-	-	X	M	W																												
5	sw x28 0 x21							F	-	-	D	-	-	X	M	W																									
6	addi x9 x9 1										F	-	-	D	X	M	W																								
7	addi x19 x19 4													F	D	X	M	W																							
8	addi x20 x20 4														F	D	X	M	W																						
9	addi x21 x21 4															F	D	X	M	W																					
10	bne x9 x18 -36 <loop>																F	D	X	M	W																				
11	addi x17 x0 10																	F	D																						
12	ecall																		F																						
13																																									
14																																									
15																																									
16																																									
17																																									
18																																									
19																																									
20																																									
21																																									
22																																									
23																																									
24																																									
25																																									
26																																									
27																																									
28																																									
29																																									
30																																									

Table 2: 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 x6 0 x19	F	D	X	M	W									F	D	X	M	W																						
2	lw x7 0 x20		F	D	X	M	W																																		
3	mul x28 x7 x6			F	D	-	X	M	W																																
4	add x28 x28 x6				F	-	D	X	M	W																															
5	sw x28 0 x21						F	D	X	M	W																														
6	addi x9 x9 1							F	D	X	M	W																													
7	addi x19 x19 4								F	D	X	M	W																												
8	addi x20 x20 4									F	D	X	M	W																											
9	addi x21 x21 4										F	D	X	M	W																										
10	bne x9 x18 -36 <loop>											F	D	X	M	W																									
11	addi x17 x0 10												F	D																											
12	ecall													F																											
13																																									
14																																									
15																																									
16																																									
17																																									
18																																									
19																																									
20																																									
21																																									
22																																									
23																																									
24																																									
25																																									
26																																									
27																																									
28																																									
29																																									
30																																									

Table 3: 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 x6 0 x19	F	D	X	M	W								F	D	X	M	W																							
2	lw x7 0 x20		F	D	X	M	W																																		
3	addi x9 x9 1			F	D	X	M	W																																	
4	addi x19 x19 x4				F	D	X	M	W																																
5	mul x28 x7 x6					F	D	X	M	W																															
6	addi x20 x20 4						F	D	X	M	W																														
7	add x28 x28 x6							F	D	X	M	W																													
8	sw x28 0 x21								F	D	X	M	W																												
9	addi x21 x21 4									F	D	X	M	W																											
10	bne x9 x18 -36 <loop>										F	D	X	M	W																										
11	addi x17 x0 10											F	D																												
12	ecall												F																												
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 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	lw x7 0 x20	F	D	X	M	W																													F	D	X	M	W		
2	lw x6 0 x19		F	D	X	M	W																																		
3	addi x28 x7 1			F	D	X	M	W																																	
4	mul x28 x28 x6				F	D	X	M	W																																
5	sw x28 0 x21					F	D	X	M	W																															
6	lw x7 4 x20						F	D	X	M	W																														
7	lw x6 4 x19							F	D	X	M	W																													
8	addi x28 x7 1								F	D	X	M	W																												
9	mul x28 x28 x6									F	D	X	M	W																											
10	sw x28 4 x21										F	D	X	M	W																										
11	lw x7 8 x20											F	D	X	M	W																									
12	lw x6 8 x19												F	D	X	M	W																								
13	addi x28 x7 1													F	D	X	M	W																							
14	mul x28 x28 x6														F	D	X	M	W																						
15	sw x28 8 x21															F	D	X	M	W																					
16	lw x7 12 x20																F	D	X	M	W																				
17	lw x6 12 x19																	F	D	X	M	W																			
18	Addi x28 x7 1																		F	D	X	M	W																		
19	Mul x28 x28 x6																			F	D	X	M	W																	
20	Sw x28 12 x21																				F	D	X	M	W																
21	Addi x9 x9 4																					F	D	X	M	W															
22	addi x19 x19 16																						F	D	X	M	W														
23	Addi x20 x20 16																							F	D	X	M	W													
24	Addi x21 x21 16																								F	D	X	M	W												
25	bne x9 x18 -96 <loop>																									F	D	X	M	W											
26	addi x17 x0 10																										F	D													
27	ecall																											F													
28																																									
29																																									
30																																									
31																																									
32																																									
33																																									