Lab 4 – Arquitetura de Computadores

99961-Inês Paiva & 100029-Teresa Nogueira

Exercício 1:

3.

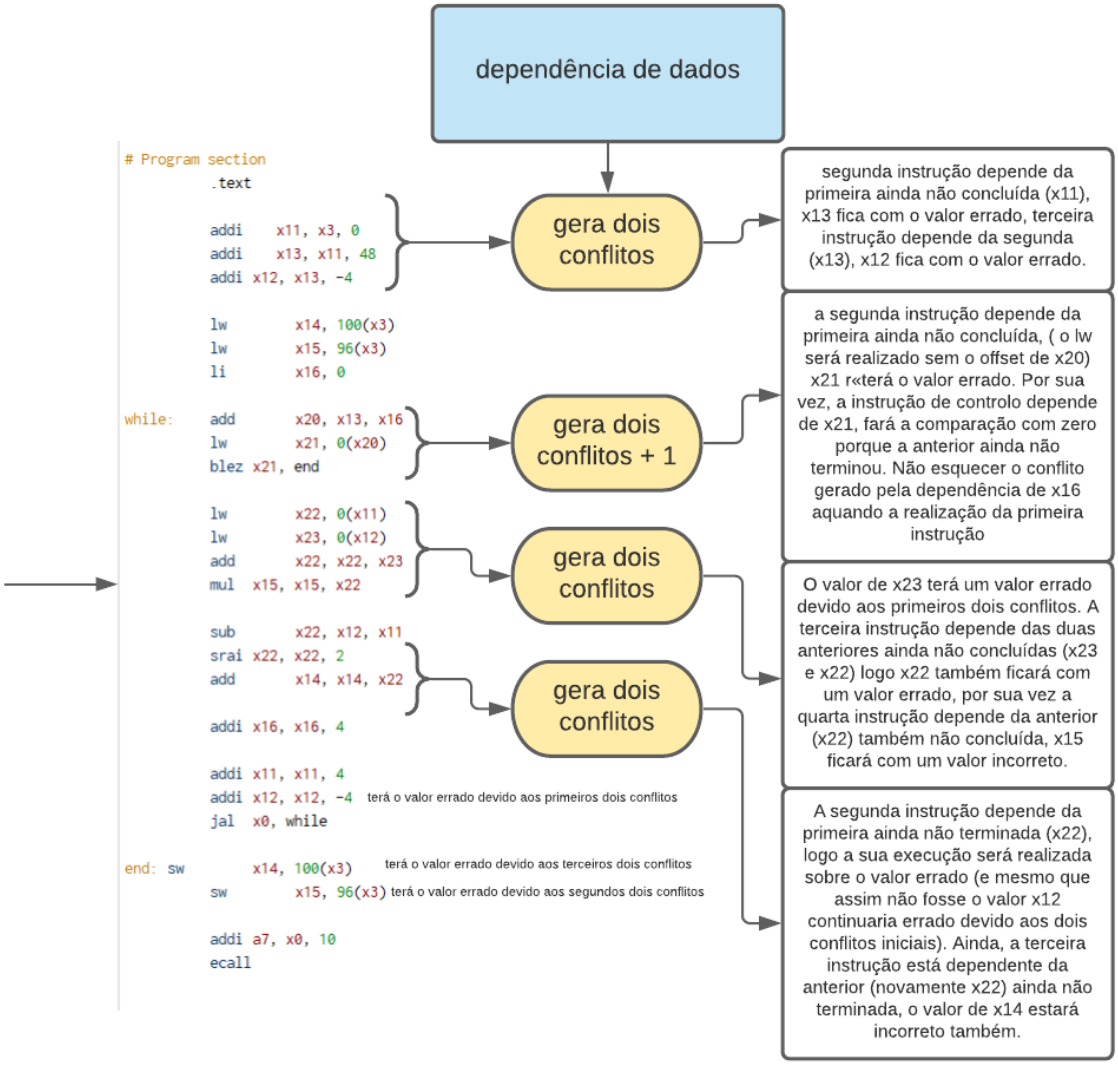
0x10000000 – a[ ]

0x10000030 – b[ ]

Q1.5. Por analise do código fornecido o endereço inicial de **a [ ]= 0x10000000h** e **b [ ]= 0x10000030h**, logo **x11= 0x10000000h** e **x13=0x10000030h**. Após colocar o *Break-Point* o registo x11 manteve o mesmo valor mas o registo **x13** ficou igual a **0x00000030h**. O valor de x13 não é igual ao esperado porque existem conflitos de dados antes do "*while*" fazendo com que o valor que se encontra em x11 quando este é chamado para calcular o valor de x13 não é o correto devido à arquitetura do processador.

Q1.6.

```
.text
addi x11, x3, 0
addi x13, x11, 48
addi x12, x13, -4
lw x14, 100(x3)
lw x15, 96(x3)
li x16, 0

while:
add x20, x13, x16
lw x21, 0(x20)
blez x21, end

lw x22, 0(x11)
lw x23, 0(x12)
add x22, x22, x23
mul x15, x15, x22

sub x22, x12, x11
srai x22, x22, 2
add x14, x14, x22

addi x16, x16, 4

addi x11, x11, 4
addi x12, x12, -4
jal x0, while

end:
sw x14, 100(x3)
sw x15, 96(x3)
addi a7, x0, 10
ecall
```



Q1.8. Agora os valores esperados para x11 e x13 correspondem à realidade, a introdução de nops (no operation) permite a execução completa da instrução anterior, à qual a próxima tem dependência, assim a segunda e a terceira operação só entram no estágio de instruction fetch quando a anterior se encontra no estágio de write back, que é o que queremos. (Esta mudança do valor de x13 deve-se ao facto de com a colocação dos *NOP'S* a leitura do x11 para calcular o valor correto de x13 ser o pretendido pois já se encontra na secção *WB* do processador.)

```
        .text

        addi    x11, x3, 0
addi     x0, x0, 0
addi     x0, x0, 0
addi     x0, x0, 0
        addi    x13, x11, 48
addi     x0, x0, 0
addi     x0, x0, 0
addi     x0, x0, 0
        addi   x12, x13, -4

        lw     x14, 100(x3)
        lw     x15, 96(x3)
        li     x16, 0
```

Q1.11.

número de ciclos de relógio: 223

número de instruções executadas: 203

Q1.12.

Instruções úteis(instruções que realizam o código): úteis fora do loop + 7*(úteis dentro do loop) + instruções até ao blez final + finais = 113

Instruções não úteis(NOP'S, stalls que garantem o funcionamento do código face ao prcessador 5 stage): não úteis fora do loop + 7*(não úteis dentro do loop) + instruções até ao blez final + finais =114

114/227 = 0,71 = 49,8%

Tendo em conta que o rácio de instruções úteis com o total de instruções é aproximadamente 50%, podemos concluir que a adaptação desenvolvida face à utilização do *processador 5-Stage in-order processor with no forwarding or hazard detection/elimination* não poderá ser considerada a mais eficiente.

Q1.13. A política de predição de salto deverá ser a de static branch prediction: predict not taken, já que não é executado o hazard detection/elimination.

Exercício 2:

Q2.3. Graças ao sistema de fowarding apenas precisamos de fazer alterações nas zonas não contempladas no encaminhamento de dados, nomeadamente, stalls após os lw, já que apenas estão disponíveis no estágio de WB e todas as instruções de controlo (já que o mecanismo não os contempla).

Q2.4.

número de ciclos de relógio: 147

número de instruções executadas: 127

Q2.5.

Instruções úteis: 113

Instruções não úteis: 30

113/143 = 79%

Tendo em conta que o rácio de instruções úteis com o total de instruções é aproximadamente 80%, podemos concluir que a adaptação desenvolvida face à utilização do *processador 5-Stage in-order processor with no hazard detection/elimination* é portanto muito mais eficiente que a desenvolvida para o processador anterior.

Q2.6.

Speedup = 223/147 = 1,52

Exercício 3:

Q3.3. Não foram introduzidas quaisquer alterações, já que os conflitos de dados são tratados pelo sistema de fowarding e os conflitos de controlo são tratados pelo sistema hazard detection/elimination, assim não é necessária a introdução de nop's (stalls) ao código fornecido, sendo este inalterado.

Q3.4.

número de ciclos de relógio: 148

número de instruções executadas: 111

Q3.5.

Instruções úteis: 113

Instruções não úteis: 0

113/113 = 100%

A execução é portanto a mais eficiente até ao momento, admitindo que não são utilizados quaisquer NOPS manualmente.

Q3.6

IPC = 0,75

Tal número é inferior a 1 já que são iniciadas várias instruções no mesmo ciclo de relógio, (relembrar o throughput performance, executa completamente **(até)** 1 instrução por ciclo de relógio) e por outro lado poderão existir instruções não acabadas (devido a possíveis conflitos de controlo).

Q3.9

	6	7	8	9	10	11	12	13	14	15	16
add x20 x13 x16	IF	ID	EX	MEM	WB						
lw x21 0(x20)		IF	ID	EX	MEM	WB					
bge x0 x21 48 <end>			IF	ID	-	EX	MEM	WB			
lw x22 0(x11)				IF	-	ID	EX	MEM	WB		
lw x23 0(x12)						IF	ID	EX	MEM	WB	
add x22 x22 x23							IF	ID	-	EX	MEM
mul x15 x15 x22								IF	-	ID	EX
sub x22 x12 x11										IF	ID
srai x22 x22 2											IF

	16	17	18	19	20	21	22	23	24	25
srai x22 x22 2	IF	ID	EX	MEM	WB					
add x14 x14 x22		IF	ID	EX	MEM	WB				
addi x16 x16 4			IF	ID	EX	MEM	WB			
addi x11 x11 4				IF	ID	EX	MEM	WB		
addi x12 x12 -4					IF	ID	EX	MEM	WB	
jal x0 0x18 <while>						IF	ID	EX	MEM	WB

AS iterações são identicas, embora ocorram em ciclos diferentes.

	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
addi x13 x11 48																																			
addi x12 x13 -4	WB																																		
lw x14 100(x3)	MEM	WB																																	
lw x15 96(x3)	EX	MEM	WB																																
addi x16 x0 0	ID	EX	MEM	WB																															
add x20 x13 x16	IF	ID	EX	MEM	WB														IF	ID	EX	MEM	WB												
lw x21 0(x20)		IF	ID	EX	MEM	WB														IF	ID	EX	MEM	WB											
bge x0 x21 48 <end>			IF	ID	-	EX	MEM	WB													IF	ID	-	EX	MEM	WB									
lw x22 0(x11)				IF	-	ID	EX	MEM	WB													IF	-	ID	EX	MEM	WB								
lw x23 0(x12)						IF	ID	EX	MEM	WB														IF	ID	EX	MEM	WB								
add x22 x22 x23							IF	ID	-	EX	MEM	WB													IF	ID	-	EX	MEM	WB					
mul x15 x15 x22								IF	-	ID	EX	MEM	WB													IF	-	ID	EX	MEM	WB				
sub x22 x12 x11										IF	ID	EX	MEM	WB														IF	ID	EX	MEM	WB			
srai x22 x22 2											IF	ID	EX	MEM	WB														IF	ID	EX	MEM	WB		
add x14 x14 x22												IF	ID	EX	MEM	WB														IF	ID	EX	MEM	WB	
addi x16 x16 4													IF	ID	EX	MEM	WB														IF	ID	EX	MEM	WB
addi x11 x11 4														IF	ID	EX	MEM	WB														IF	ID	EX	MEM
addi x12 x12 -4															IF	ID	EX	MEM	WB														IF	ID	EX
jal x0 0x18 <while>																IF	ID	EX	MEM	WB														IF	ID
sw x14 100(x3)																	IF	ID																	IF
sw x15 96(x3)																		IF																	
addi x17 x0 10																																			
ecall																																			

Q3.10.

Stalls de conflitos de dados raw: 2

Stalls introduzidos em consequência de conflitos de controlo: 0 (são realizadas duas instruções após o último jal graças ao sistema de predict not taken, após ser reconhecido o branch são invalidadas)

Q3.11. OS Stalls promovem a existência de instruções inacabadas aquando a existência de conflitos de dados e de controlo, é portanto normal que o IPC seja inferior a 1.

Exercício 4:

Q4.2. Tendo em conta os conflitos encontrados na tabela supra, reordenamos as instruções, passando a instrução addi a7, 10 para o inicio do código para poupar operações de Stall e passamos duas instruções de lw para ante da instrução de controlo blez, mitigando assim a propagação de Stalls que advinham destes mesmos lw (assim não precisam de um nop para chegar ao estágio de WB), no entanto reduzir as instruções inacabadas provenientes do jal, sendo estas um conflito de controlo permanente, uma vez que não é possível "corrigir" o código de forma a mitigá-las, graças ao invalidate (transforma a instrução carregada nos registos de pipeline num NOP, colocando todos os enables de escrita (WE) a 0).

Q4.3.

número de ciclos de relógio: 133

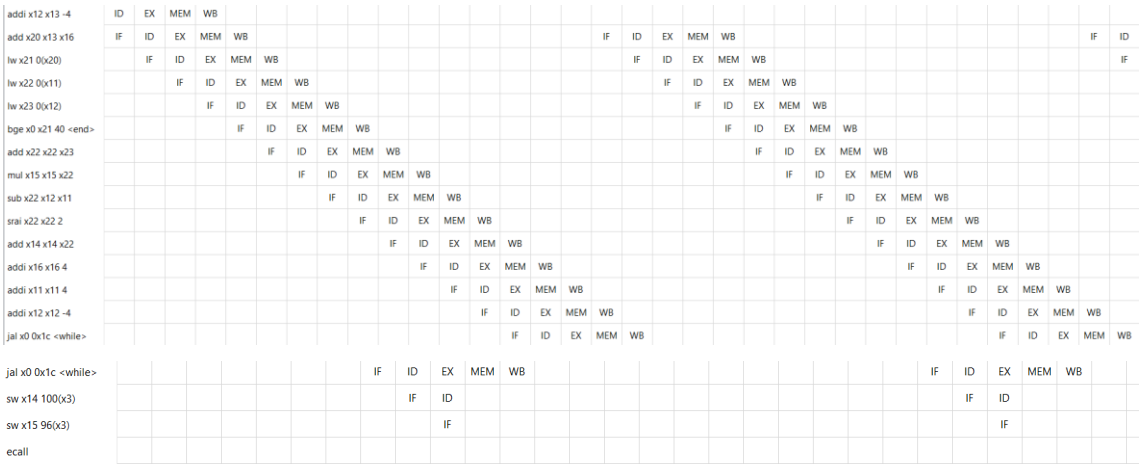
número de instruções executadas: 113

Q4.4.

IPC = 0,85

Este parametro aumentou em relação ao anterior (0,75) já que foram mitigadas as operações não realizadas devido à introdução de stalls aquando a existência de conflitos.

Q4.7.



Q4.8.

Stalls de conflitos de dados raw: 0

Stalls introduzidos em consequência de conflitos de controlo: 0 (são realizadas duas instruções após o último jal graças ao sistema de predict not taken, após ser reconhecido o branch são invalidadas)

Q4.9.

O número é inferior a 1 já que, novamente, são iniciadas várias instruções no mesmo ciclo de relógio, (relembrar o throughput performance, executa completamente **(até)** 1 instrução por ciclo de relógio), por outro lado, a operação de jal é realizada, sendo a política de predição de salto predict not taken, as duas instruções seguintes são realizadas mas invalidadas e portanto não concluídas.

Exercício surpresa

Q6.2

número de ciclos de relógio: 105

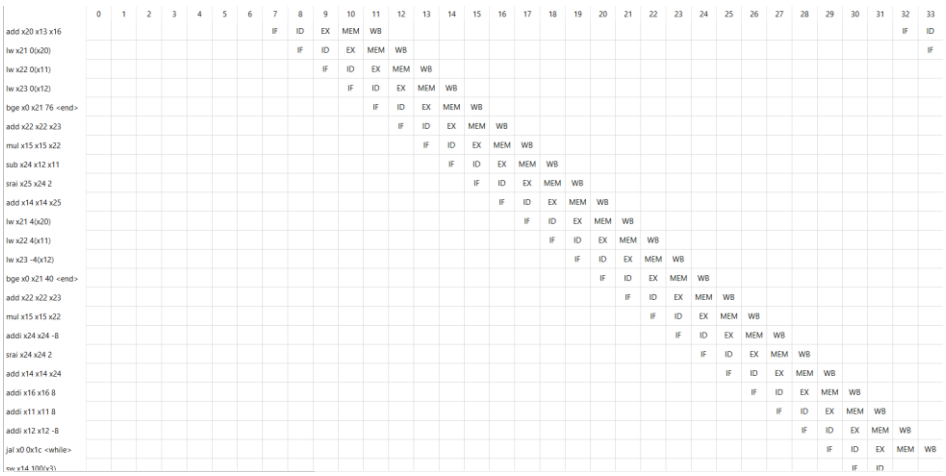
número de instruções executadas: 93

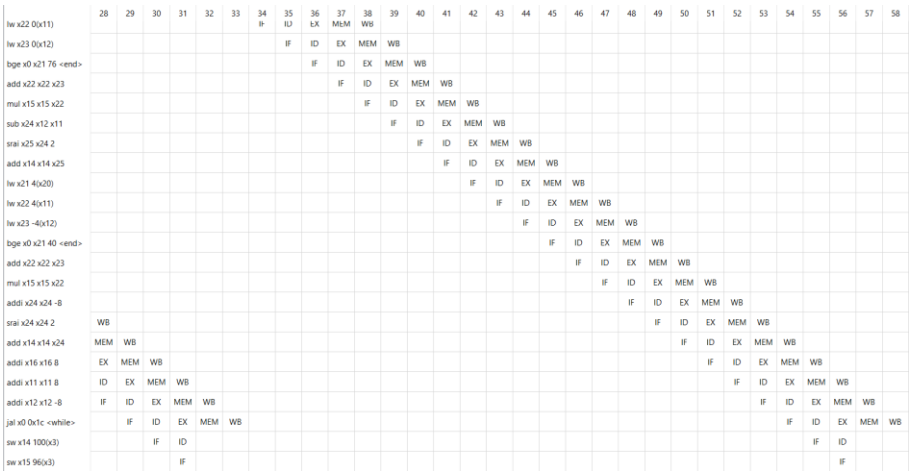
Q6.3.

IPC = 0,886

O IPC aumentou ligeiramente face ao exercício 4, embora se tenha realizado o método de unrolling, mitigando assim o número de conflitos de controlo no jal, a diminuição de instruções de forma a otimizar o código diminui também o número de instruções realizadas por ciclo, que por conseguinte resulta em apenas um pequeno aumento do IPC.

Q6.6.





	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58
lw x22 0(x11)							I+	IU	bX	MeM	WB																				
lw x23 0(x12)								IF	ID	EX	MEM	WB																			
bge x0 x21 76 <end>									IF	ID	EX	MEM	WB																		
add x22 x22 x23										IF	ID	EX	MEM	WB																	
mul x15 x15 x22											IF	ID	EX	MEM	WB																
sub x24 x12 x11												IF	ID	EX	MEM	WB															
srai x25 x24 2													IF	ID	EX	MEM	WB														
add x14 x14 x25														IF	ID	EX	MEM	WB													
lw x21 4(x20)															IF	ID	EX	MEM	WB												
lw x22 4(x11)																IF	ID	EX	MEM	WB											
lw x23 -4(x12)																	IF	ID	EX	MEM	WB										
bge x0 x21 40 <end>																		IF	ID	EX	MEM	WB									
add x22 x22 x23																			IF	ID	EX	MEM	WB								
mul x15 x15 x22																				IF	ID	EX	MEM	WB							
addi x24 x24 -8																					IF	ID	EX	MEM	WB						
srai x24 x24 2	WB																					IF	ID	EX	MEM	WB					
add x14 x14 x24	MEM	WB																					IF	ID	EX	MEM	WB				
addi x16 x16 8	EX	MEM	WB																					IF	ID	EX	MEM	WB			
addi x11 x11 8	ID	EX	MEM	WB																					IF	ID	EX	MEM	WB		
addi x12 x12 -8	IF	ID	EX	MEM	WB																					IF	ID	EX	MEM	WB	
jal x0 0x1c <while>		IF	ID	EX	MEM	WB																					IF	ID	EX	MEM	WB
sw x14 100(x3)			IF	ID																								IF	ID		
sw x15 96(x3)				IF																									IF		

Q6.7.

Stalls de conflitos de dados raw: 0

Stalls introduzidos em consequência de conflitos de controlo: 0 (são realizadas duas instruções após o último jal graças ao sistema de predict not taken, após ser reconhecido o branch são invalidadas)

Q6.8.

Tal número é inferior a 1 já que após a instrução de jal são realizadas duas instruções, posteriormente cortadas graças ao reconhecimento do salto, logo, por ciclo de relógido ficamos com instruções não acabadas, ainda, são iniciadas várias instruções no mesmo ciclo de relógio, (relembrar o throughput performance, executa completamente **(até)** 1 instrução por ciclo de relógio).

Q6.9.

Speedup = 133/105 = 1,27