Revisão: exercício

Case Study 1 – pg 143

2.1

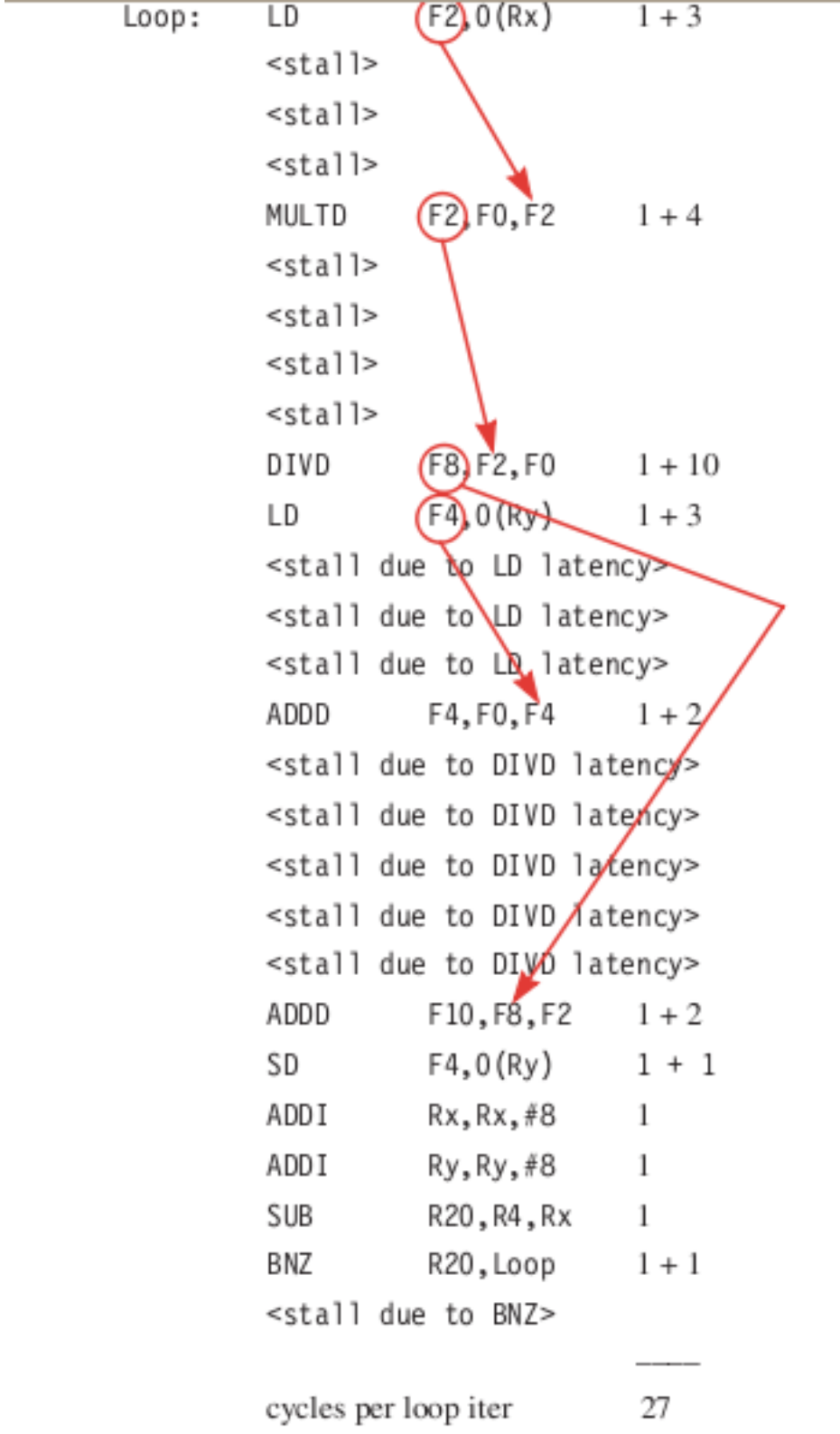
- Cada instrução ocupa os recursos mesmo que não esteja usando!
- As instruções subsequentes não começam até que a instrução anterior tenha terminado totalmente de executar (WB)
- Este é o pior caso

```
Loop: LD      F2,0(Rx)  1+3
      MULTD   F2,F0,F2  1+4
      DIVD    F8,F2,F0  1 + 10
      LD      F4,0(Ry)  1+3
      ADDD    F4,F0,F4  1+2
      ADDD    F10,F8,F2 1+2
      SD      F4,0(Ry)  1 + 1
      ADDI    Rx,Rx,#8  1
      ADDI    Ry,Ry,#8  1
      SUB     R20,R4,Rx 1
      BNZ     R20,Loop  1+1

                        ____
cycles per loop iter    37
```

2.2

- Se o processador detecta as relações de produtor/consumidor, só estas devem esperar



2.3

- LD, MULD, DIVD todos tem dependência de dados... nada se pode fazer
- O LD depois do DIVD não depende de nenhum dado não processado, então pode ser executado no segundo pipe sem problemas ao mesmo tempo
- O ADDD também não depende do DIVD mas precisa terminar o LD terminar, stall
- O próximo ADDD precisa esperar o ADD anterior e o DIVD terminar, como o DIVD tem maior latência, espera-se o DIVD
- O SD não depende de ninguem e vai para o segundo pipe
- O ADDI que não depende do SD executa e o ADDI que pedende vai para o segundo pipe (stall de 1 ciclo)
- SUB não tem delay, BNZ executa sem problemas

	Execution pipe 0		Execution pipe 1	
Loop:	LD F2,0(Rx)	;	<nop>	
	<stall for LD latency>	;	<nop>	
	<stall for LD latency>	;	<nop>	
	<stall for LD latency>	;	<nop>	
	MULTD F2,F0,F2	;	<nop>	
	<stall for MULTD latency>	;	<nop>	
	<stall for MULTD latency>	;	<nop>	
	<stall for MULTD latency>	;	<nop>	
	<stall for MULTD latency>	;	<nop>	
	DIVD F8,F2,F0	;	LD F4,0(Ry)	
	<stall for LD latency>	;	<nop>	
	<stall for LD latency>	;	<nop>	
	<stall for LD latency>	;	<nop>	
	ADDD F4,F0,F4	;	<nop>	
	<stall due to DIVD latency>	;	<nop>	
	<stall due to DIVD latency>	;	<nop>	
	<stall due to DIVD latency>	;	<nop>	
	<stall due to DIVD latency>	;	<nop>	
	<stall due to DIVD latency>	;	<nop>	
	<stall due to DIVD latency>	;	<nop>	
	ADDD F10,F8,F2	;	SD F4,0(Ry)	
	ADDI Rx,Rx,#8	;	ADDI Ry,Ry,#8	
	SUB R20,R4,Rx	;	BNZ R20,Loop	
	<stall due to BNZ>	;	<nop>	

cycles per loop iter 24

2.4

- Se houver uma interrupção entre N e N+1 a instrução N+1 não pode completar, mesmo que seja mais rapida que N. Pode-se, no entanto, assumir que todas as intruções do pipe terminam de executar antes da interrupção ser tratada.
- Não podem haver hazards WAW! Se N+1 escrever em um registrador N não pode escrever depois dele.
- As instruções com maiores latências podem ser ultrapassadas mais facilmente. Por exemplo um DIVD vai completar muito depois de um LD

2.5

	Execution pipe 0		Execution pipe 1	
Loop:	LD F2,0(Rx)	;	LD F4,0(Ry)	
	<stall for LD latency>	;	<stall for LD latency>	
	<stall for LD latency>	;	<stall for LD latency>	
	<stall for LD latency>	;	<stall for LD latency>	
	MULTD F2,F0,F2	;	ADDD F4,F0,F4	
	<stall for MULTD latency>	;	<stall for ADDD latency>	
	<stall for MULTD latency>	;	<stall for ADDD latency>	
	<stall for MULTD latency>	;	SD F4,0(Ry)	
	<stall for MULTD latency>	;	<nop>	
	DIVD F8,F2,F0	;	<nop>	
	<stall for DIVD latency>	;	<nop>	
	<stall for DIVD latency>	;	<nop>	#ops: 11
	<stall for DIVD latency>	;	<nop>	#nops: $(22 \times 2) - 11 = 33$
	<stall for DIVD latency>	;	<nop>	
	<stall for DIVD latency>	;	<nop>	
	<stall for DIVD latency>	;	<nop>	
	<stall for DIVD latency>	;	<nop>	
	<stall for DIVD latency>	;	<nop>	
	ADDI Rx,Rx,#8	;	ADDI Ry,Ry,#8	
	SUB R20,R4,Rx	;	<nop>	
	ADDD F10,F8,F2	;	BNZ R20,Loop	
	<stall due to BNZ>	;	<stall due to BNZ>	

cycles per loop iter 22

2.6

- a) 33 desperdícios x 2 x 22 oportunidades)
  - 33/44 = 0.75
- b) Loop unrolling
  - unroll 2 iterações do loop
  - 22 ciclos por iteração do loop
  - 2 iterações implicam em 11 ciclos por processador
  - O Speedup seria de 22/11 = 2