Universidade Federal do Espírito Santo

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Arquitetura de Computadores I – Turmas 01 e 02 (EARTE) – 2021/2
Prof. Rodolfo da Silva Villaça – rodolfo.villaca@ufes.br

GRUPO: Dionatas Santos, Maria Julia Nolasco, Otávio Sales

Laboratório VII – Conflitos no Pipeline

1. Objetivos

Após completar as atividades deste laboratório, você irá:

- Conhecer como funciona o pipeline da CPU MIPS;
- Verificar o ganho de desempenho da CPU pipeline com relação à CPU monociclo;
- Verificar a influência dos conflitos no desempenho da CPU pipeline;
- Verificar com alguns conflitos podem ser evitados pela reorganização de instruções.

2. Atividades

Para executar esta atividade você poderá utilizar os simuladores MARS e DrMIPS[1]. Considere o programa *fibonacci.asm* a seguir para executar esta atividade. O programa deveria calcular a famosa sequência de Fibonacci e armazenar os valores de cada passo do algoritmo na memória (*Vout*)e o último valor calculado em *res*.

```
.data
num: .word 10
res: .word 0
Vout: .space 400

.text
#carregando registradores com valores aleatorios
la $s1, Vout
lw $a0, num
li $t0, 1
li $t1, 1
li $t2, 1

loop:
ble $a0, $t0, end
sw $t2, 8($s1)
addi $s1, $s1, 4
addi $t0, $t0, 1
addi $14, $0, -2
move $t3, $t2
add $t2, $t2, $t1
move $t1, $t3
b loop

end:
sw $t2, 8($s1)
sw $t2, res
```

Tarefa 1

Avalie e, se necessário, corrija o programa Fibonacci.asm no DrMIPS com a **CPU monociclo** (unicycle.cpu) para valores de *num* = {10, 20, 30, 40} e preencha a tabela a seguir com a variável *res* no final da execução e a quantidade de instruções executadas.

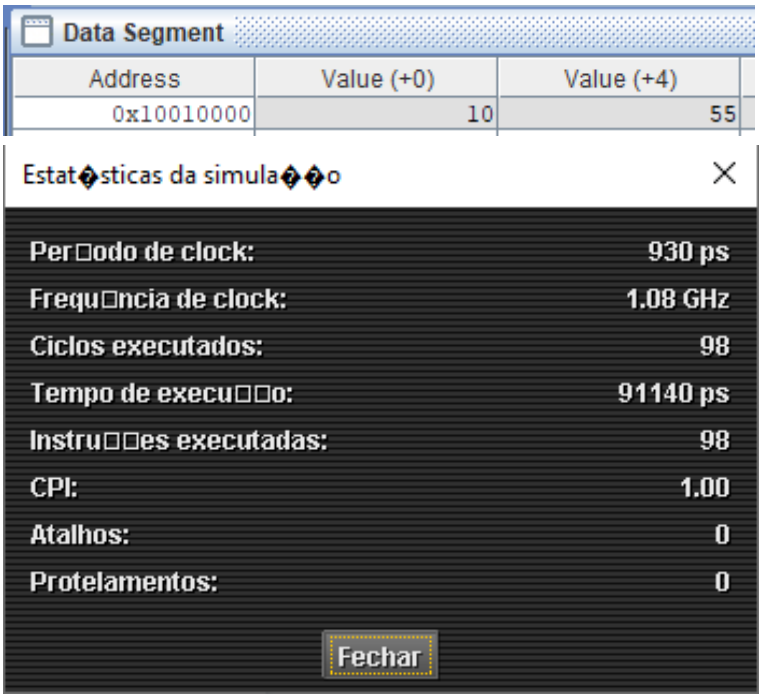
Dica: Use a ferramenta de geração de estatísticas de execução para auxiliar na resposta.

1 https://github.com/brunonova/drmips

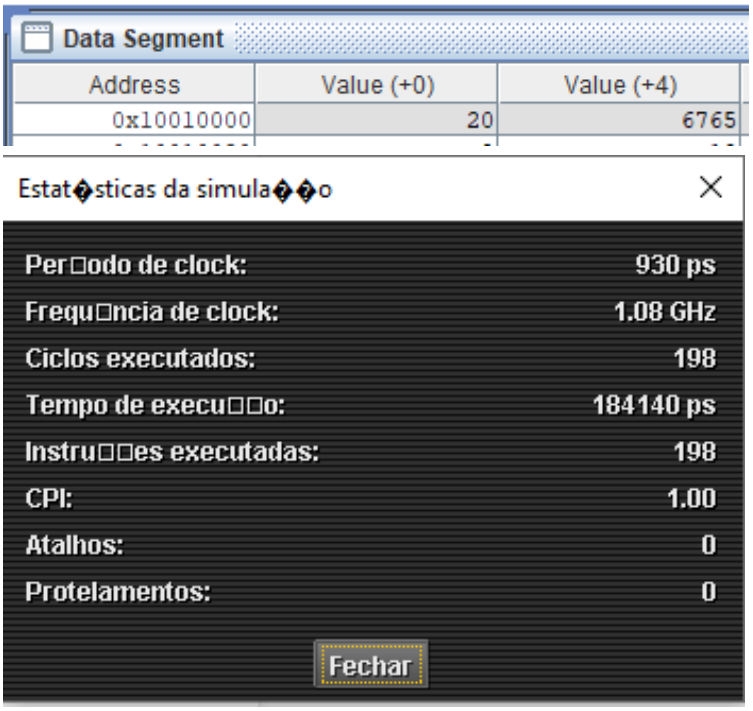
num	*res*	*#instrucoes*
10	55	98
20	6765	198
30	932040	298
40	102334155	398

Endereço do segmento de texto res Value (+4) e Quantidade de instruções

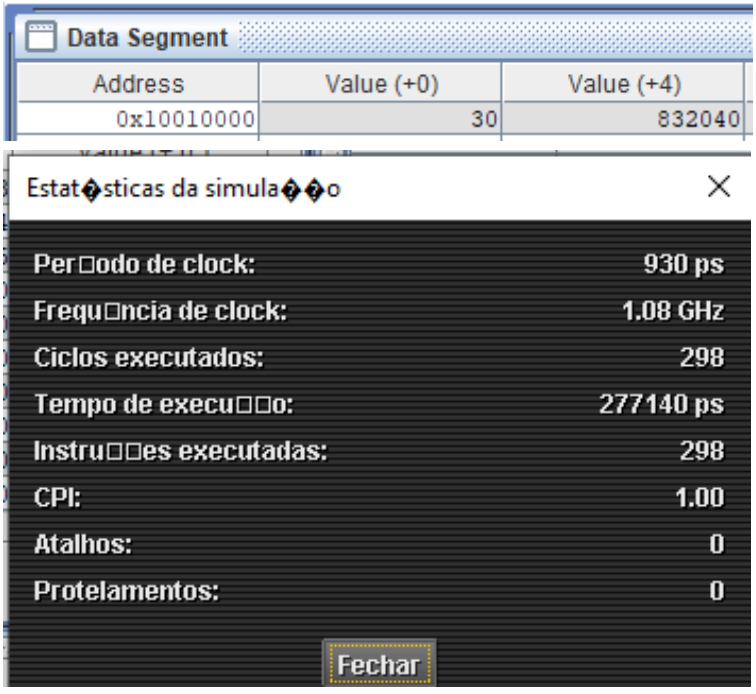
num = 10



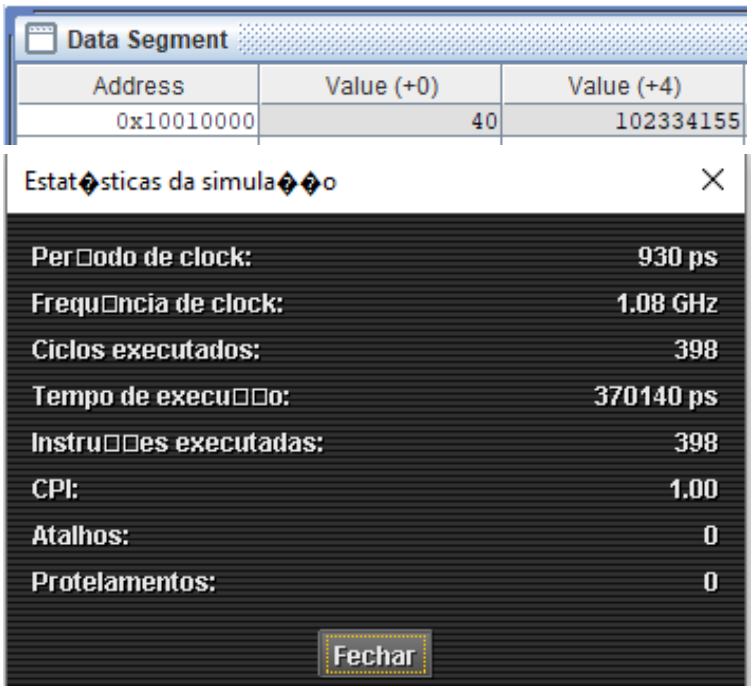
num=20


Data Segment
Address
Value (+0)
Value (+4)
0x10010000
20
6765
Estat□sticas da simula□□o
Per□odo de clock:
930 ps
Frequ□ncia de clock:
1.08 GHz
Ciclos executados:
198
Tempo de execu□□o:
184140 ps
Instru□□es executadas:
198
CPI:
1.00
Atalhos:
0
Protelamentos:
0
Fechar

num=30


Data Segment
Address
Value (+0)
Value (+4)
0x10010000
30
832040
Estat□sticas da simula□□o
Per□odo de clock:
930 ps
Frequ□ncia de clock:
1.08 GHz
Ciclos executados:
298
Tempo de execu□□o:
277140 ps
Instru□□es executadas:
298
CPI:
1.00
Atalhos:
0
Protelamentos:
0
Fechar

num=40



Tarefa 2

A seguir você deverá executar o seu código numa **CPU pipeline de 5 estágios**, conforme projeto apresentado no Capítulo 4 do livro texto (usar o modelo **pipeline-no-hazard-detection.cpu** no DrMIPS).

Analise o código em busca de possíveis conflitos que causariam problemas na execução no pipeline original. Liste quais são estes possíveis conflitos e a sua causa (dependência dados, mudança de fluxo). Analise e identifique quais podem ser resolvidos por reorganização das instruções durante a compilação.

Apresente o código reorganizado.

Possíveis conflitos [Hazards, a partir da página 335 do livro texto]

Hazards de dependência de dados em instruções de move

move $t3, $t2

add $t2, $t2, $t1

move $t1, $t3

Hazards de mudança de fluxo

ble $a0, $t0, end

b loop

Ambos problemas poderiam ser resolvidos com o uso de bolhas, no caso do “move” em específico no segundo move , deveria ser adicionado uma bolha após a primeira instrução de move, sendo assim atrasado (em 1 clock por exemplo).

Foi necessário corrigir o código

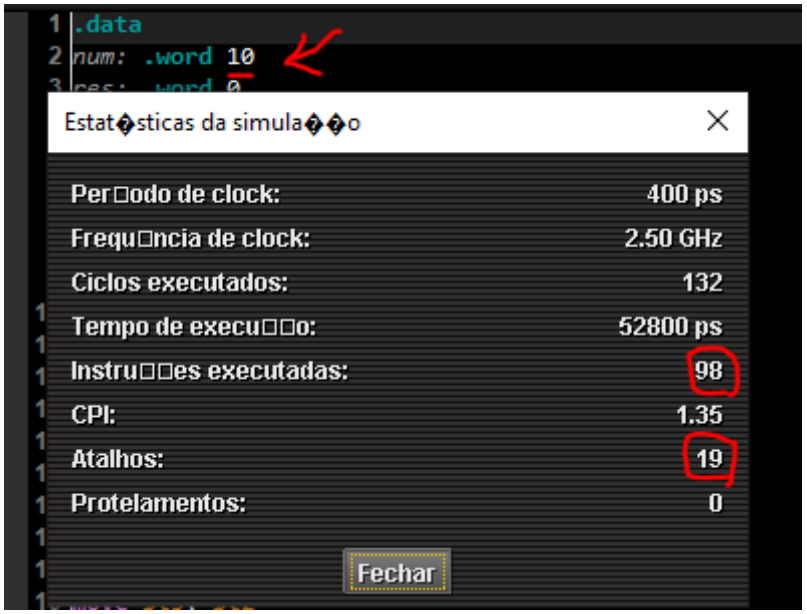
```
.data
num: .word 10       #aqui foi mudado para 20, 30 e 40
res: .word 0
Vout: .space 400
.text

#carregando registradores com valores aleatórios
la $s1, Vout
lw $a0, num         #num = valor do numero da fibonacci
li $t0, 1           #contador
li $t1, 1
li $t2, 1

loop:
ble $a0, $t0, end  #aqui é um while(count<=num)
sw $t2, 8($s1)     #mem[$1 + 8] = $t2 ; mem[$1 + 8] == Vout[1]
addi $s1, $s1, 4   #aqui avança +4 (uma posição no vetor)
addi $t0, $t0, 1   #count ++
addi $14, $0, -2
move $t3, $t2      #$t3 = $t2
add $t2, $t2, $t1  #$t2 = $t2 + $t1
move $t1, $t3      #$t1 = $t3
b loop

end:
#sw $t2, 8($s1)
sw $t3, res #foi mudado o "sw $t2, res"
```

Tarefa 3

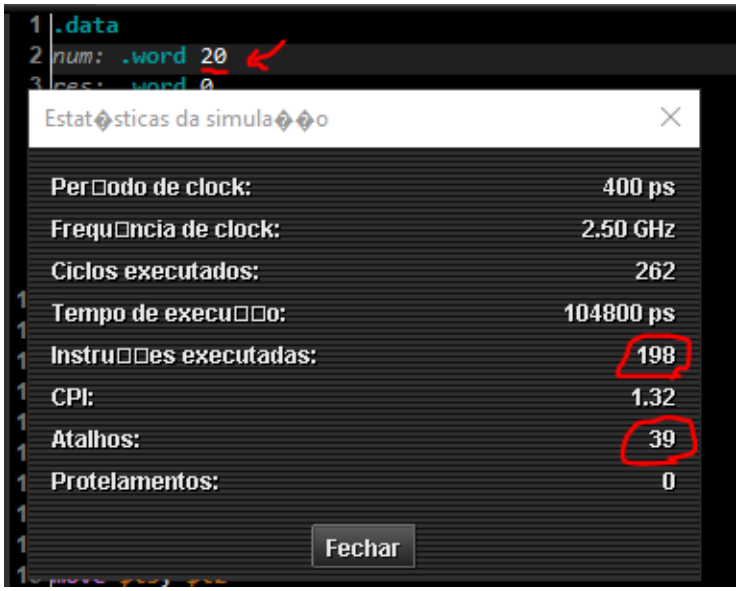
Execute **o seu programa** fibonacci.asm (**corrigido**) na **CPU pipeline (pipeline.cpu)** do simulador DrMIPS e preencha a tabela abaixo, mostrando: a variável res no final da execução, a quantidade de instruções executadas e o número de bolhas geradas durante a execução do seu programa:

num	*res*	#instruçoes	#bolhas
10	55	98	19
20	6765	198	39
50	Overflow	498	99
90	Overflow	898	179

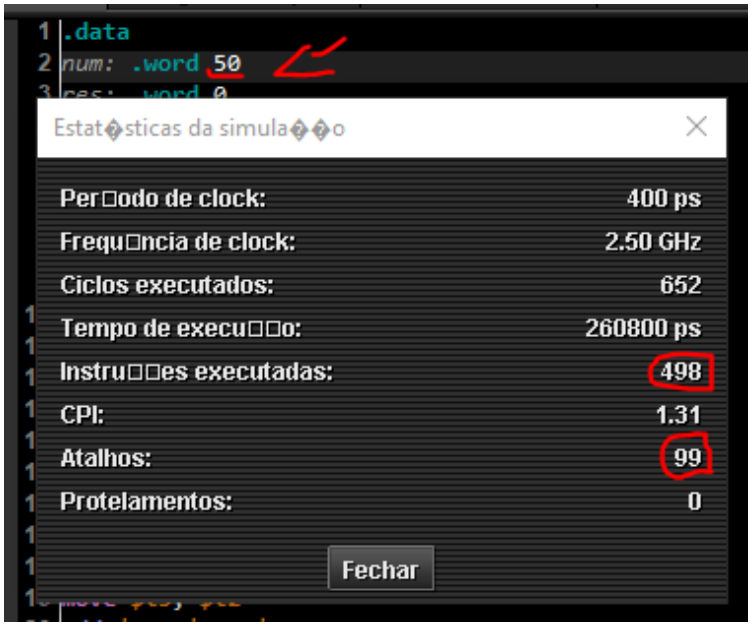
Com 10


.data
num: .word 10
Estat□sticas da simula□□o
Per□odo de clock: 400 ps
Frequ□ncia de clock: 2.50 GHz
Ciclos executados: 132
Tempo de execu□□o: 52800 ps
Instru□□es executadas: 98
CPI: 1.35
Atalhos: 19
Protelamentos: 0
Fechar

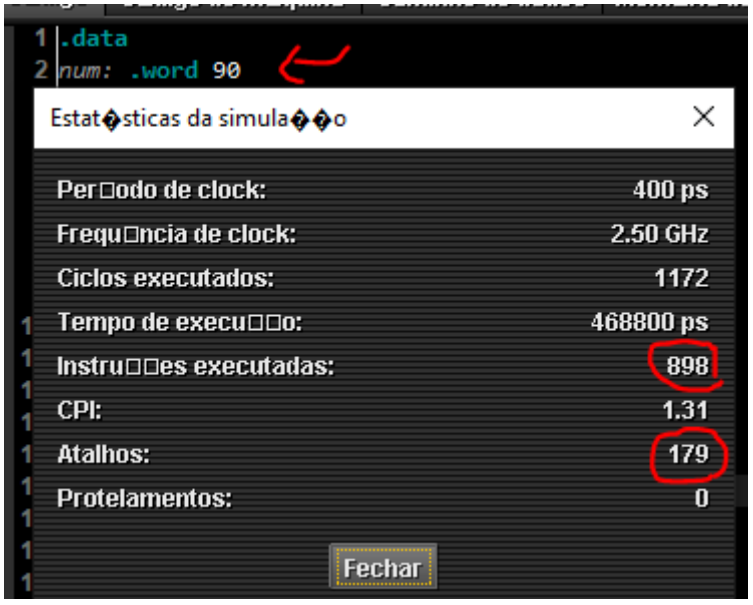
Com 20


.data
num: .word 20
Estat□sticas da simula□□o
Per□odo de clock: 400 ps
Frequ□ncia de clock: 2.50 GHz
Ciclos executados: 262
Tempo de execu□□o: 104800 ps
Instru□□es executadas: 198
CPI: 1.32
Atalhos: 39
Protelamentos: 0
Fechar

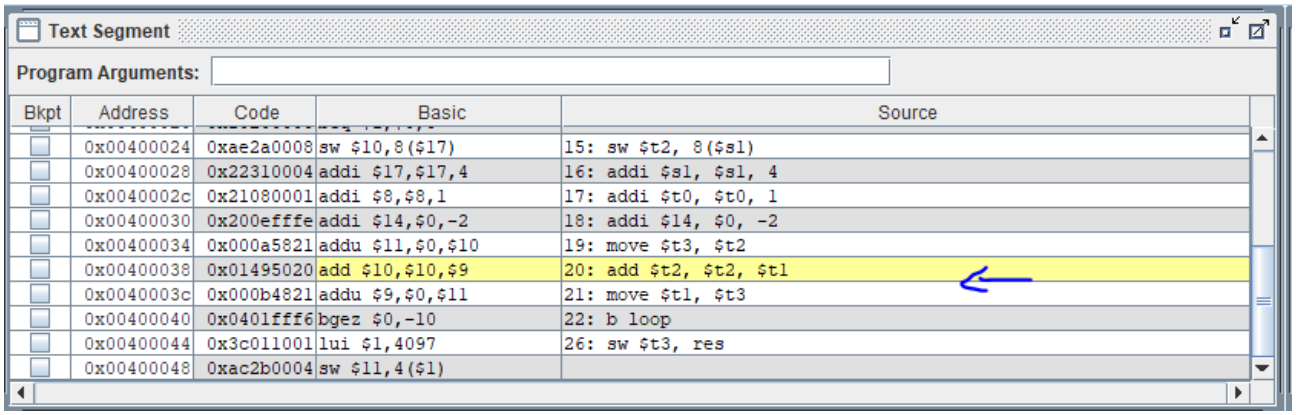
Com 50

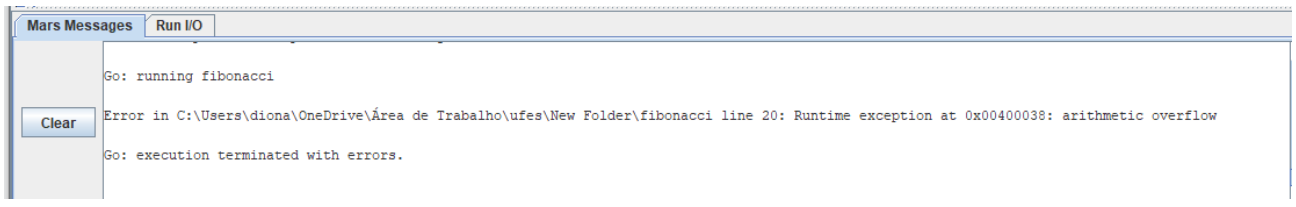


Com 90



Com 50 e 90 = Overflow





Tarefa 4

Calcule o ganho de desempenho da CPU pipeline em comparação com a CPU monociclo para as mesmas entradas. Justifique sua resposta!

Desempenho depende de 3 fatores:

-número de instruções

-tempo de ciclo de clock(período)

-número de clocks por instrução(cpi)

*Sendo desempenho = 1/tempo de execução

num=10	Número de Instruções	Período	CPI	Tempo Execução	Desempenho (1/s)
CPU Monociclo	98	930 ps	1.00	91140 ps	$1{,}0972\times 10^{-5}$
CPU Pipeline	98	400 ps	1.35	52800 ps	$1{,}8939\times 10^{-5}$

num=30	Número de Instruções	Período	CPI	Tempo Execução	Desempenho (1/s)
CPU Monociclo	298	930 ps	1.00	277140 ps	$3{,}0682\times 10^{-6}$
CPU Pipeline	298	400 ps	1.32	156800 ps	$6{,}3775\times 10^{-6}$

num=40	Número de Instruções	Período	CPI	Tempo Execução	Desempenho (1/s)
CPU Monociclo	398	930 ps	1.00	370140 ps	$2{,}7016\times 10^{-6}$
CPU Pipeline	398	400 ps	1.31	208800 ps	$4{,}7892\times 10^{-6}$

Como podemos ver pelas tabelas acima os tempos são os mesmos independentemente do *num* e o tempo da CPU Monociclo é maior do que o da CPU Pipeline para o mesmo *num* e o mesmo número de instruções. Além disso, mesmo que a CPI da CPU Pipeline diminua conforme o *num* aumenta, essa diminuição não é muito significativa. Em todos os casos podemos ver que a CPU Pipeline tem um desempenho melhor do que a CPU Monociclo.