**Alunos: lab7, lccao, sgfl, vlma**

**6#**

1. Dado quo array {7, 6, 5, 4, 1, 2} está carregado na memória na posição 8092 hexadecimal, o código seguinte opera um sort no array ordenando-o em ordem crescente. Essa ordenação ocorre da seguinte forma: A primeira posição do array é comparada com todas as outras e, caso seja menor que a posição comparada, eles são trocados entre si. Em seguida a segunda posição é comparada com as próximas e assim sucessivamente até a última posição.

1. Por volta de 440 ciclos.
2. Por volta de 416 ciclos.
3. Foram adicionados nops após instruções de adição e após a de comparação (slt).

#include <reg.h>

.data

A: .word 7, 6, 5, 4, 1, 2

.text

.globl start

.ent start

start: lui a0,0x8002

ori a0, a0, 0x00b0

addi a1, $0,0x6

nop

xor t0,t0,t0

CW1: beq t0,a1,EW1

addi t1,t0,1

nop

nop

CW2: beq t1, a1, EW2

add t2, t0, t0

add t2, t2, t2

add t2, a0, t2

add t3, t1, t1

add t3, t3, t3

add t3, a0, t3

lw t4, 0(t2)

nop

lw t5, 0(t3)

nop

slt t6, t4, t5

nop

nop

nop

bne t6, $0, IGN

nop

beq t4, t5, IGN

sw t5, 0(t2)

sw t4, 0(t3)

IGN: addi t1, t1, 1

nop

beq $0, $0, CW2

EW2: addi t0, t0, 1

nop

beq $0, $0, CW1

EW1: beq $0, $0, FIM

FIM: nop

break

.end start

1. Sim, implementando previsão de desvio, e possivelmente com uma ajuda do compilador na reordenação das previsões para evitar dependências.

**7#**

1. Por volta de 307 ciclos.
2. Utilizaremos Fowarding para evitar o conflito de dados, com isso a otimização é que não será necessário instruções nops após a escrita em registradores, exceto caso seja após instruções de carregamento de memória. Com isso foi otimizado em 22 ciclos. 285 no total.
3. Para esse algoritmo, pensamos em usar uma técnica de previsão que sempre prevê que o char a ser analisado não é um “a” dada a probabilidade menor de sua ocorrência. Então o desvio não iria ocorrer. Caso a previsão esteja errada, iremos contar com a volta para o beq e o desvio ocorrendo. No fim das contas da 150 ciclos.