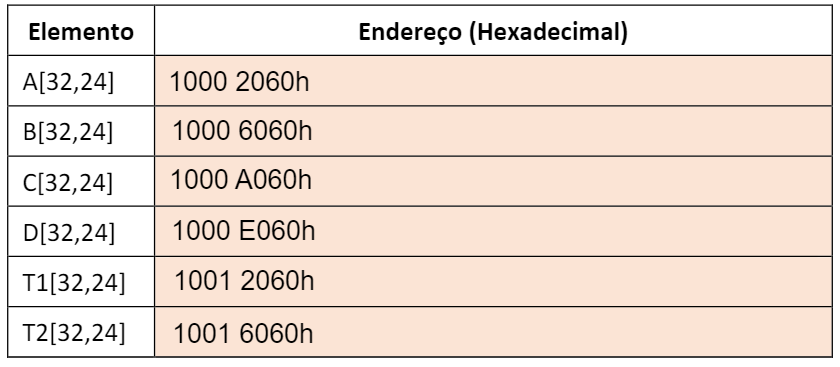
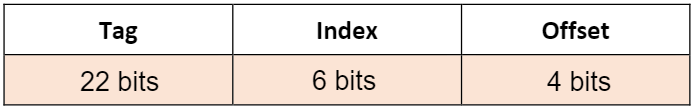
Laboratório 5 – Memória Cache

Exercício 1:

1.4)



1.6)

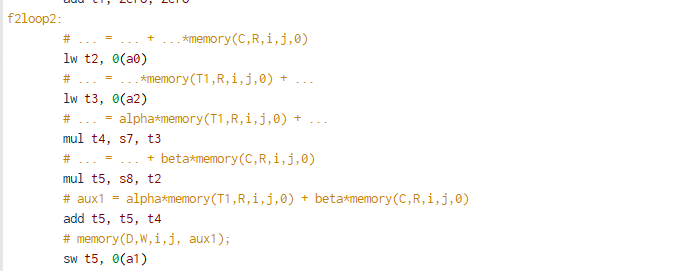


1.7)



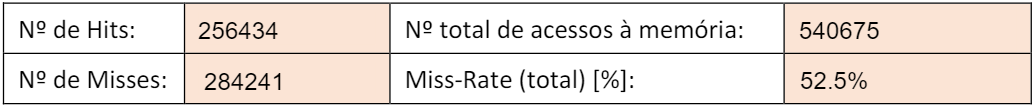
1.8)

Para este programa são necessárias pelo menos 3 vias de associatividade para guardar o conteúdo das 3 matrizes utilizadas durante o mesmo (uma via para cada matriz). Isto para que seja possível diminuir o número de misses por cada iteração do loop apresentado em baixo, ou seja, caso houvesse apenas 1 via, o número de misses ia aumentar consideravelmente pois sempre que se analisasse o endereço de outra matriz, ia dar sempre dar miss. Caso houvesse 2 vias, sempre que fosse ler o endereço da 3ª ou 1ª matriz ia dar miss.

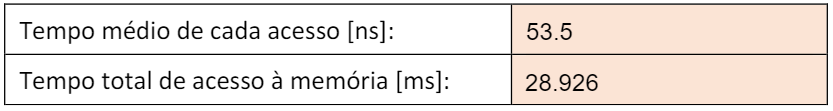


Exercício 2:

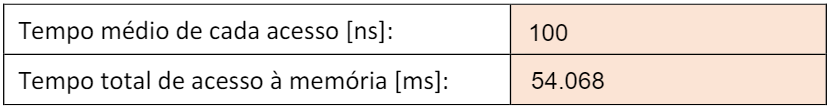
2.5)



2.6)



2.7)



2.8)

Através da análise dos dados do 2.6 e do 2.7, podemos verificar uma diferença bastante significativa de quase o dobro do tempo, no tempo total de acesso à memória. Assim, podemos concluir que é benéfico a utilização da cache para diminuir bastante a latência em relação a acesso de memória.

Exercício 3

3.2)

Alterar os ciclos “for” iria diminuir bastante o miss rate devido a forma como os dados da matriz estão implementados na memoria cache, isto é, como os dados relativos a cada matriz estão dispostos na cache, linha a linha, ao alterar-mos o código para ler os dados também linha a linha em vez de coluna a coluna iria ser bastante vantajoso para a eficiência do programa.

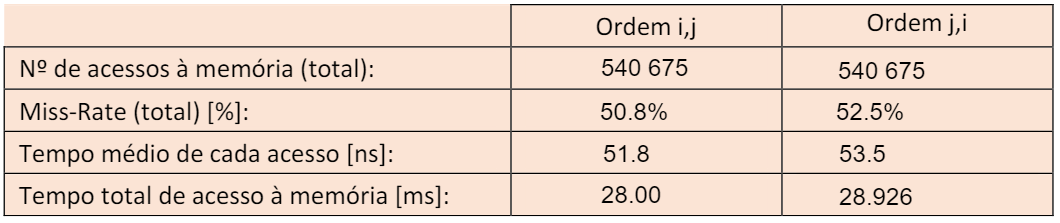
3.3)

A cache tem capacidade para 4 words por matriz, por isso, a cada 4 números da matriz lidos na memória, iria haver 1 miss. Assim, passávamos a ter uma hit-rate de 75% (3 hits para 1 miss), em vez do que acontecia ao ler coluna a coluna.

3.4)

Podemos assim concluir, que a ordem mais favorável seria a que acabamos de analisar (leitura dos dados linha a linha através da inversão do loop), pois iria aumentar bastante a hit-rate e consideravelmente a eficiência do programa em relação ao tempo total de acesso à memoria.

3.5)

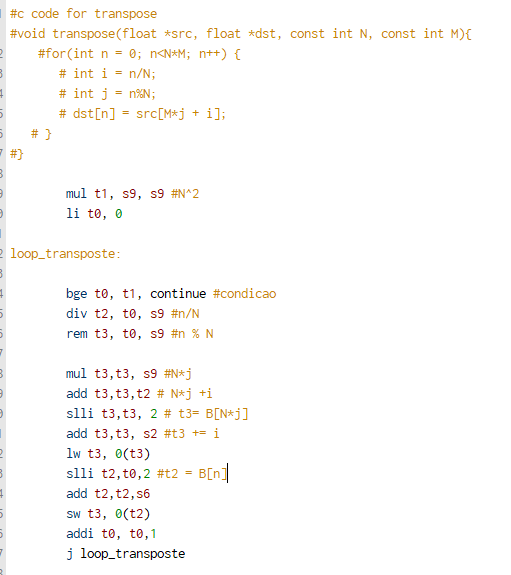


3.6)

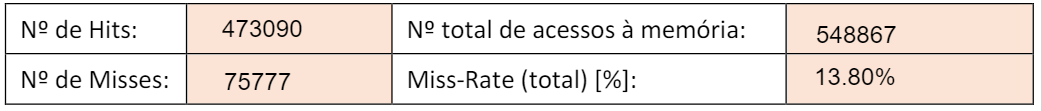
Apesar da pequena diminuição do miss-rate em relação à ordem ”j,i” , não era esperado uma diminuição muito mais significativa pois a maioria dos misses do programa acontecem na primeira parte.

Exercício 4:

4.1)



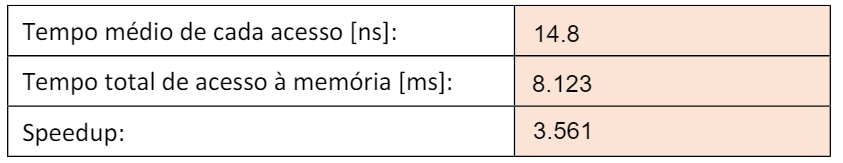
4.3)



4.4)

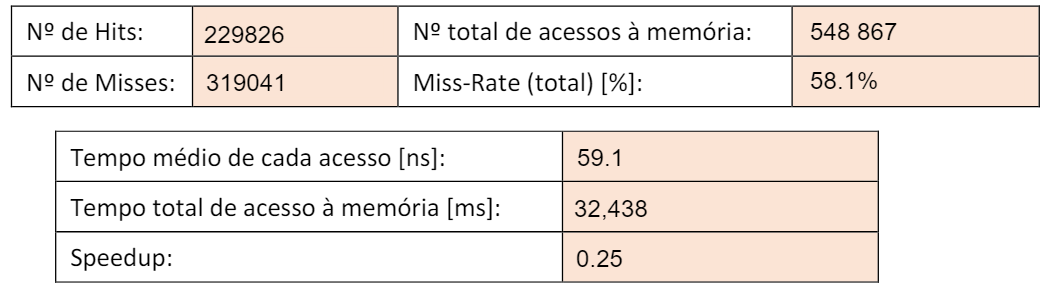
A miss-rate diminui bastante comparado as outras versões antigas do programa, isto porque ao fazer a transposta estamos a inverter as colunas com as linhas, permitindo assim ao que a cache tenha muito menos misses.

4.5)

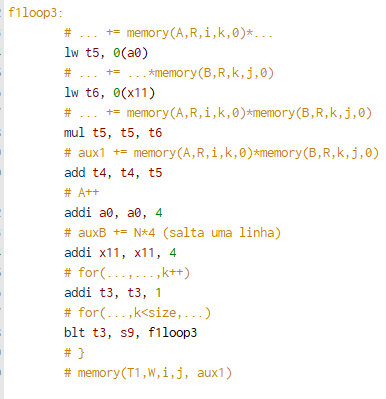


Exercício 5:

5.2)

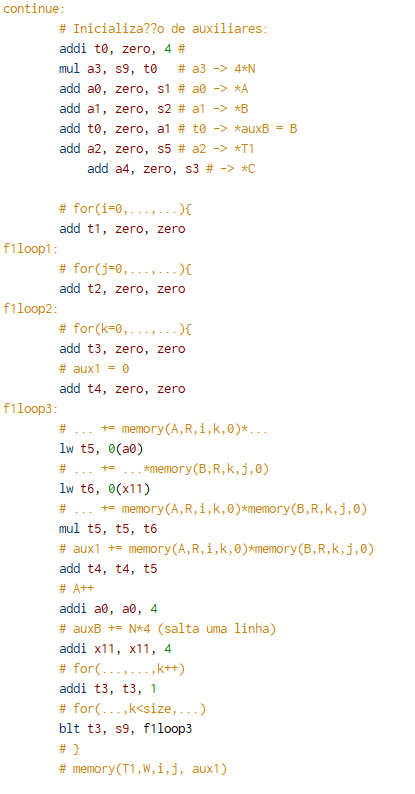


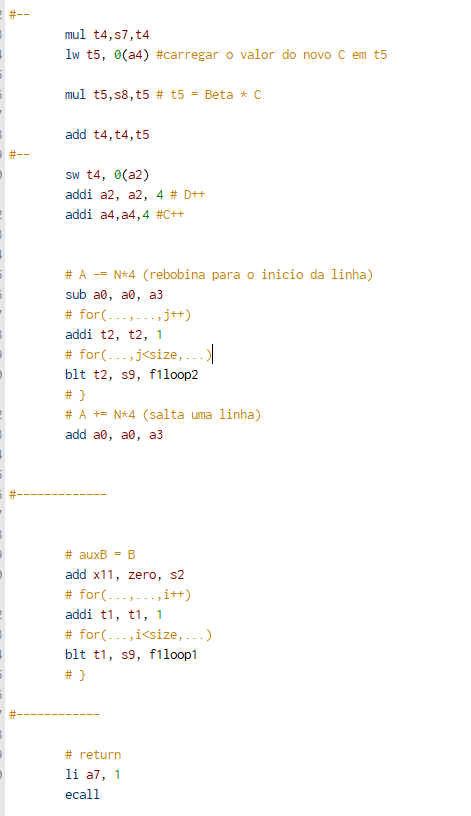
5.3)



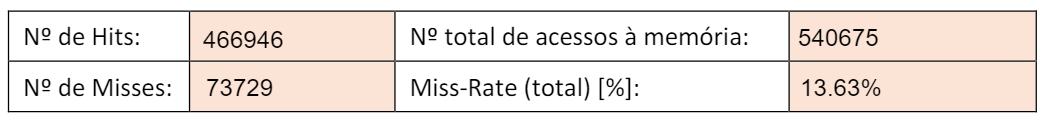
Podemos observar no código anterior que há vários acessos à memoria e como há apenas uma via de mapeamento, as tags vão diferentes para cada matriz o que implica que haja mais misses enquanto o loop esta a ser a executado.

Exercicio do lab  
6.1)





6.3)



6.4)

Ao retirar a matriz auxiliar, podemos verificar que o numero de acessos diminui consideravelmente, e uma diminuição ligeira no miss-rate em relação ao exercício 4.

6.5)

