UNIVERSIDADE FEDERAL DE PELOTAS
CENTRO DE DESENVOLVIMENTO TECNOLÓGICO
CIÊNCIA DA COMPUTAÇÃO
ENGENHARIA DE COMPUTAÇÃO
Arquitetura e Organização de Computadores II

Exercícios de Revisão para a Prova 1 - Respostas

1- Quantos níveis pode ter uma hierarquia de memória?

Quantos forem necessários.

2- O que é um bloco?

Unidade mínima de informação, contendo um certo
número de palavras de memória.

3- O que é a taxa de acertos também chamada de hit ratio?

Corresponde à fração dos acessos à memória encontrados
no nível superior (com frequência, é usada como medida de
desempenho do sistema de memória)

4- Se o hit ratio for de 0.78, quanto será seu miss ratio?

0.22

5- Descreva o que é penalidade por falta.

É o tempo necessário para substituir um dos blocos do nível superior pelo bloco
do nível inferior que contém a informação desejada, mais o
tempo para enviar a informação ao processador

6 - Qual a maior vantagem e desvantagem do uso de associatividade na cache?

O aumento do grau de associatividade resulta, em geral, na
redução da taxa de faltas

• A desvantagem é o aumento do tempo de tratamento do acerto e do custo em hardware, pois mais bits precisam ser armazenados (tag) e mais comparações precisam ser feitas para encontrar o bloco

7 - Cite as vantagens e desvantagens dos seguintes mapeamentos:

a) Direto

Vantagens

• Não há necessidade de algoritmo de substituição

• Hardware simples e de baixo custo

• Alta velocidade de operação

• Desvantagens

• Desempenho cai se acessos consecutivos são feitos a palavras com mesmo índice

• hit ratio inferior ao de caches com mapeamento associativo

• O hit ratio aumenta com o aumento da cache, aproximando-se de caches com mapeamento associativo

b) Totalmente associativo e em qual cenário esta estratégia é mais utilizado.

Vantagem:

• Máxima flexibilidade no posicionamento de qualquer palavra (ou linha) da memória principal em qualquer palavra (ou linha) da cache

• Desvantagens:

• Custo em hardware da comparação simultânea de todos os endereços armazenados na cache

• Algoritmo de substituição (em hardware) para selecionar uma linha da cache como consequência de um miss

• Utilizado apenas em memórias associativas de pequeno tamanho

c) Completamente associativo:

• comparadores são compartilhados por todos os conjuntos

• algoritmo de substituição só precisa considerar linhas dentro de um Conjunto

• Conjunto-associativa N-way X mapeamento direto:
• Dado tem atraso extra do multiplexador
• Dado vem DEPOIS da decisão Hit/Miss e da seleção do conjunto
• No mapeamento direto a linha da cache está disponível ANTES da decisão Hit/Miss
• Possível assumir um hit e continuar. Recuperar depois se for miss

8- Cite quais são as fontes de misses, porque acontecem e as suas respectivas soluções.

Compulsórios (cold start ou chaveamento de processos, primeira referência): primeiro acesso a uma linha
• De Conflito (ou Colisão): Múltiplas linhas de memória acessando o mesmo conjunto da cache conjunto-associativa ou mesma linha da cache com mapeamento direto
• Solução 1: aumentar tamanho da cache
• Solução 2: aumentar associatividade
• De Capacidade: Cache não pode conter todas as linhas acessadas pelo programa
• Solução: aumentar tamanho da cache
• Invalidação: outro processo (p.ex. I/O) atualiza a memória

9- Explique as vantagens e desvantagens das estratégias de escrita write through e write back:

Esquema Write Through
• Vantagens
• Fácil de implementar

• Um cache miss nunca resulta em uma escrita na memória principal

• Memória sempre possui o dado mais recente, mantendo sempre a coerência dos dados

• Desvantagens

• Escritas são mais lentas

• Cada escrita requer em acesso a memória principal

• Necessita de maior largura de banda de memória

• Esquema Write Back

• Vantagens

• As escritas ocorrem na velocidade da cache

• Várias escritas em um bloco da cache resultam em apenas uma escrita na memória principal

• Menor largura de banda e menor consumo de energia

• Desvantagens

• Implementação mais complexa

• Dados na cache e na memória principal podem estar inconsistentes

• Uma leitura pode gerar uma escrita na memória principal

10- O que é o "bit sujo" no esquema write back?

Como muitas vezes o bloco armazenado na cache não está consistente com a memória principal, um "bit se sujo" é usado para indicar esta inconsistência

11- Explique quando um write miss ocorre e as estratégias utilizadas para lidar com este evento.

Quando o processador solicita uma escrita o endereço

solicitado não está armazenado na cache.

• Duas opções em caso de write miss:
• Alocação de escrita
• As perdas de escrita atuam como perdas de leitura
• Alocação sem escrita
• As perdas de escrita não afetam a cache
• O bloco é modificado apenas na memória principal

12- Descreva o problema que haverá ao se utilizar um sistema com uma cache juntamente do esquema de escrita write through com buffer de escrita cuja frequência das escritas é muito maior que o ciclo de escrita da memória DRAM. Como poderia-se resolver este problema? (Justifique)

O buffer de escrita vai sofrer overflow. Resolve-se inserindo uma cache L2 para salvar as informações de forma mais rápida.

13. Porque não podemos utilizar apenas memórias SRAM para a construção da hierarquia de memória. Explique as vantagens e desvantagens desta tecnologia de implementação.

Memórias SRAM possuem tempos de acesso bem menores que memórias DRAM e memórias secundárias, porém, possuem um custo muito maior de fabricação, inviabilizando a possibilidade de ser memórias SRAM com grandes capacidades. Além disso, as SRAMs possuem área e consumo muito maiores também.

Vantagens de SRAMs: são mais rápidas que DRAMs e memórias secundárias, além de não precisarem de refresh.
Desvantagens de SRAMs: baixa densidade, custo mais alto de fabricação e maior consumo de energia.

14. Explique porque nos sistemas computacionais atuais o uso de uma hierarquia de memória é fundamental?

Do ponto de vista ideal, deseja-se ter acesso a uma quantidade ilimitada de memória com um tempo de acesso muito pequeno. Porém, na prática, devido a restrições tecnológicas, é necessário utilizar hierarquia de memória para fornecer ao usuário uma maior capacidade de memória juntamente com um tempo de acesso desejável. Dessa forma com as memórias mais próximas do processador (Cache) sendo mais rápidas e menores, e as mais distantes (RAM) possuindo maior capacidade e uma velocidade muito menor.

15. Explique o funcionamento de uma cache, e porque ela aumenta a eficiência do sistema.

Quando o processador precisa ler determinada informação da memória (seja de dados ou de instruções), realiza um acesso à cache para tentar obter a informação desejada, caso contrário, tal informação precisa ser buscada no nível de memória inferior na hierarquia. O processo de escrita na memória varia um pouco dependendo do esquema adotado (write-through ou write-back), mas o ganho de eficiência é justificado pelos mesmos motivos que nas situações de leitura da memória.

No momento que um dado é recentemente escrito/lido, a chance dele ser acessado novamente nos próximos instantes é muito grande. Aproveitando esta característica e o fato de que o tempo de buscar um dado na cache é muito menor do que de se buscar na RAM devido as técnologias utilizadas, pode-se obter grandes ganhos de tempo de acesso a memória ao se utilizar a cache..

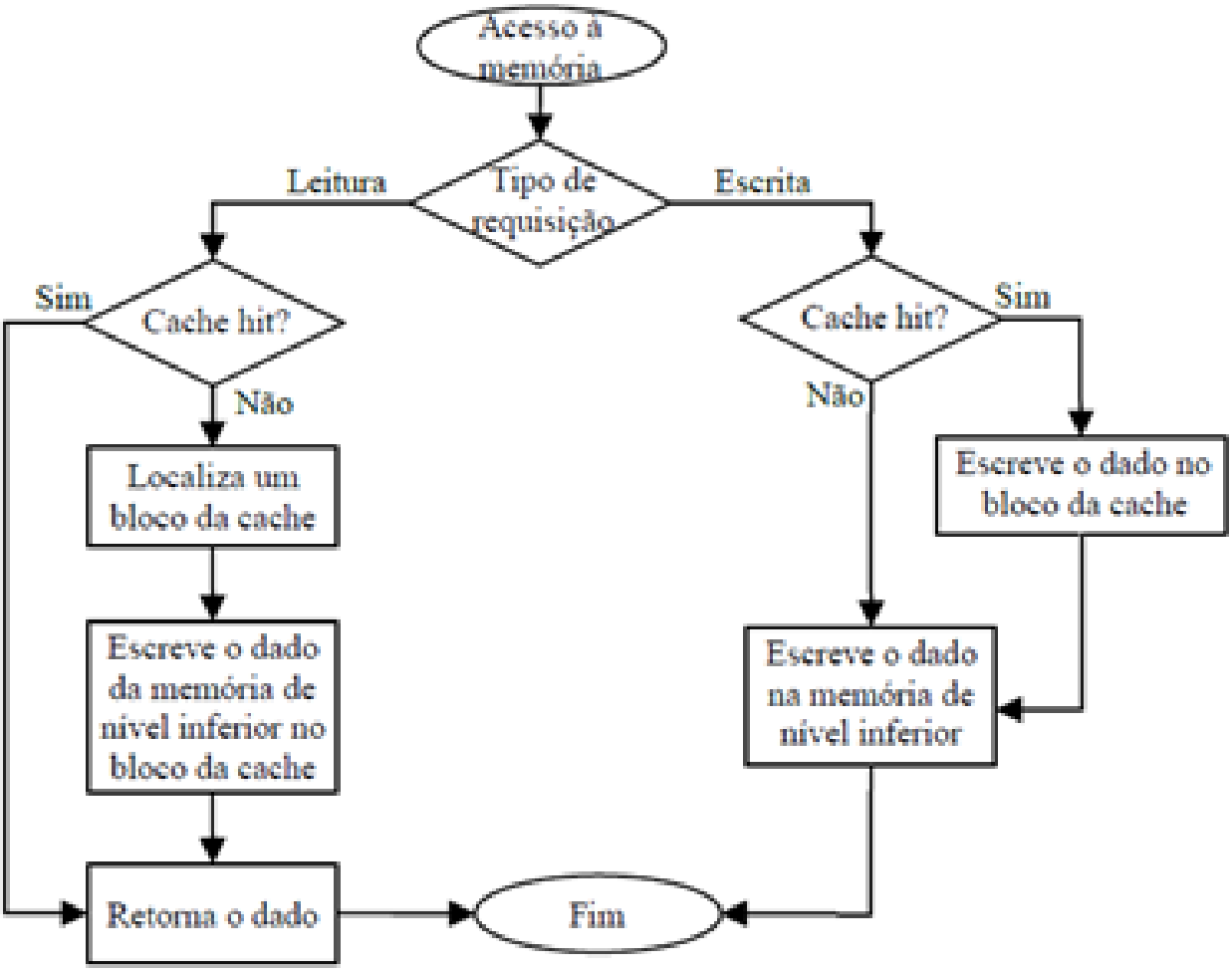
16. Considerando um sistema de cache, responda as seguintes questões:

(a) Durante uma operação de escrita na cache o gerenciador da memória cache pode utilizar uma das duas estratégias possíveis: write-through ou write-back. Explique o funcionamento das duas estratégias.

Write-through: a informação é escrita tanto no bloco da cache quanto no bloco da memória situada no nível inferior da hierarquia (a memória principal, no caso da cache).

Write-back: a informação é escrita somente no bloco da cache. O bloco modificado é escrito no componente inferior da hierarquia somente no momento da sua substituição na cache.

(b) Crie o fluxograma de uma cache write-through sem alocação de escrita.



17. Considerando um sistema de cache, responda as seguintes questões:

(a) Uma cache consistindo em quatro blocos de uma word cada, pode apresentar os seguintes mapeamentos: cache totalmente associativa, cache conjunto associativa com dois blocos por conjunto e cache diretamente mapeada. Explique como funciona cada modo de mapeamento neste sistema de cache (exemplifique).

Cache totalmente associativa: *o conteúdo de cada endereço de memória referenciado pelo processador pode ser mapeado para qualquer bloco da cache. Por exemplo: supondo a cache inicialmente vazia e que o endereço de memória 13D esteja sendo referenciado pelo processador; o conteúdo dessa posição de memória seria mapeado para qualquer um dos quatro blocos da cache. Se, na sequência, o endereço de memória 37D fosse referenciado pelo processador, o conteúdo desse endereço de memória seria mapeado para um bloco ainda não ocupado da cache, e assim por diante.*

Cache conjunto-associativa com dois blocos por conjunto: *o conteúdo de cada endereço de memória referenciado pelo processador pode ser mapeado para qualquer bloco dentro do conjunto correspondente da cache. Por exemplo: supondo a cache inicialmente vazia e que o endereço de memória 13D esteja sendo referenciado pelo processador; o conteúdo dessa posição de memória seria mapeado para um dos dois blocos do conjunto de índice 1 da cache, pois 13 mod 2 = 1. Mas, se, na sequência, o endereço de memória 37D fosse referenciado pelo processador, o conteúdo desse endereço de memória também seria mapeado para o conjunto de índice 1 da cache, pois 37 mod 2 = 1; porém, a informação seria mapeada para o outro bloco do conjunto, já que tal bloco estaria vazio no momento.*

Cache mapeada diretamente: *o conteúdo de cada endereço de memória referenciado pelo processador pode ser mapeado somente para um bloco da cache (e sempre para o mesmo bloco). Por exemplo, supondo a cache inicialmente vazia e que o endereço de memória 13D esteja sendo referenciado pelo processador; o conteúdo dessa posição de memória seria mapeado para o bloco de índice 1 da cache, pois 13 mod 4 = 1. Mas, se, na sequência, o endereço de memória 37D fosse referenciado pelo processador, o conteúdo desse endereço de memória também seria mapeado para o bloco de índice 1 da cache, pois 37 mod 4 = 1, havendo, portanto, uma falta por conflito de dados.*

(b) Algumas das estratégias para redução do miss rate são: utilizar tamanho de blocos maiores, utilizar caches de tamanhos maiores e maior associatividade. Explique cada uma destas estratégias e como elas podem reduzir o miss rate.

Tamanhos de bloco maiores: *permitem explorar a localização espacial, uma vez que, ao ocorrer uma falta, as informações de várias posições adjacentes do nível de memória inferior*

são lidas, de modo que existe uma (grande) possibilidade de que tais conteúdos adjacentes sejam referenciados em breve, o que contribuiria para diminuir a taxa de faltas.

Caches de tamanhos maiores: *fornecem mais opções de blocos para mapear o conteúdo de posições do nível de memória inferior, de modo que isso contribui para diminuir a taxa de faltas, já que o índice de faltas por conflitos de dados tende a diminuir. Além disso, é possível armazenar mais dados na cache.*

Maior associatividade: *quanto maior o grau de associatividade de uma cache, mais opções de blocos haverá para mapear o conteúdo de posições do nível inferior de memória, uma vez que cada uma dessas posições será mapeada para um determinado conjunto, que possuirá mais posições quanto maior for a associatividade da cache; isso contribui para que o índice de faltas por conflitos de dados seja menor, o que leva à redução da taxa de faltas.*

(c) Discuta as desvantagens associadas a cada um destas estratégias.

Tamanhos de bloco maiores: *a taxa de faltas pode crescer se o bloco representar uma fração considerável do tamanho total da cache, pois o número de blocos que podem ser mantidos na cache será menor, o que pode gerar uma grande competição por espaço na cache. Como resultado, um bloco vai ser retirado da cache antes que muitas de suas palavras tenham sido acessadas. A localidade espacial entre as palavras de um bloco diminui com o aumento do tamanho do bloco. Outro problema associado ao aumento do tamanho do bloco é o aumento do custo da falta, que é determinada pelo tempo necessário à busca de um bloco no nível imediatamente inferior na hierarquia*

Caches de tamanhos maiores: *apresenta maior lentidão no acesso, pois os circuitos da memória são maiores.*

Maior associatividade: *quanto maior o grau de associatividade de uma cache, maior será a quantidade de bits destinada a tags, o que faz com que boa parte da capacidade total de bits da cache não seja usada para dados.*

18- Os itens A e B contém uma lista de referências para endereços de memória de 5 bits.

A. 0, 31, 12, 7, 31, 13, 12, 31, 7, 31, 7, 12

B. 1, 17, 1, 17, 5, 25, 29, 1, 25, 29, 17, 1

Para cada uma das referências da Tabela 1, realizar os mapeamentos direto (itens a, b) e associativo em quatro (somente item a) e duas vias (somente item b) de uma cache com 8 blocos. Também listar se cada referência requisitada apresenta cache hit/miss calculando a taxa de ausência de endereços. Assumir que inicialmente a cache está vazia. Se houver necessidade de substituição de blocos na cache, utilize o algoritmo LRU (Last Recently Used – último recentemente usado) para definir qual endereço será substituído na memória cache.

1) Direto:

a)

Compulsory Misses: 5
Total Cache Queries: 12
Capacity Misses: 0
Total Misses: 5
Conflict Misses: 0
Miss Rate: 41.67 %
Cache Hits: 7
Hit Rate: 58.33 %

b)

Compulsory Misses: 5
Total Cache Queries: 12
Capacity Misses: 0
Total Misses: 5
Conflict Misses: 0
Miss Rate: 41.67 %
Cache Hits: 7
Hit Rate: 58.33 %

2) Assoc 4 a)

Compulsory Misses: 5
Total Cache Queries: 12
Capacity Misses: 0
Total Misses: 5
Conflict Misses: 0
Miss Rate: 41.67 %
Cache Hits: 7
Hit Rate: 58.33 %

3) Assoc 2 b)

Compulsory Misses: 4
Total Cache Queries: 12
Capacity Misses: 0
Total Misses: 7
Conflict Misses: 3
Miss Rate: 58.33 %
Cache Hits: 5
Hit Rate: 41.67 %

19. Quantos conjuntos possui a cache L1 4-way de 64 Kbytes de um processador Ultra Sparc III com blocos de 32 palavras de 64bits?

Tamanho de bloco = 32 * 64 bits = 2048 bits = 256 bytes = 0,25Kbyte

N Conjuntos = 64 Kbytes / 0,25 Kbytes = 256 blocos / 4 (4-way) = 64 conjuntos

20. Projete uma cache de dados de tamanho de 1MB mapeamento conjunto associativo de 2 vias que utiliza endereços de 32 bits e 128 bytes por bloco e utiliza uma memória endereçada à bytes. Calcule:

(a) Quantos bits são utilizados para o offset, índice e tag?

Já que o endereçamento da memória é feito por bytes e cada bloco contém 128 bytes, são necessários 7 bits para o offset, a fim de permitir a localização de um byte dentro de um bloco.

Número total_blocos = 1MB/128B=2^20/128= 8192 blocos

Como é uma cache conjunto-associativa de duas vias, o número de conjuntos é:

Nº de conjuntos= 8192 / 2 = 4096 conjuntos

Como 4096=2^12, são necessários 12 bits para o índice. Logo, a tag terá um número de bits de:

Nº de bits_tag= 32 - 12 - 7 = 13 bits

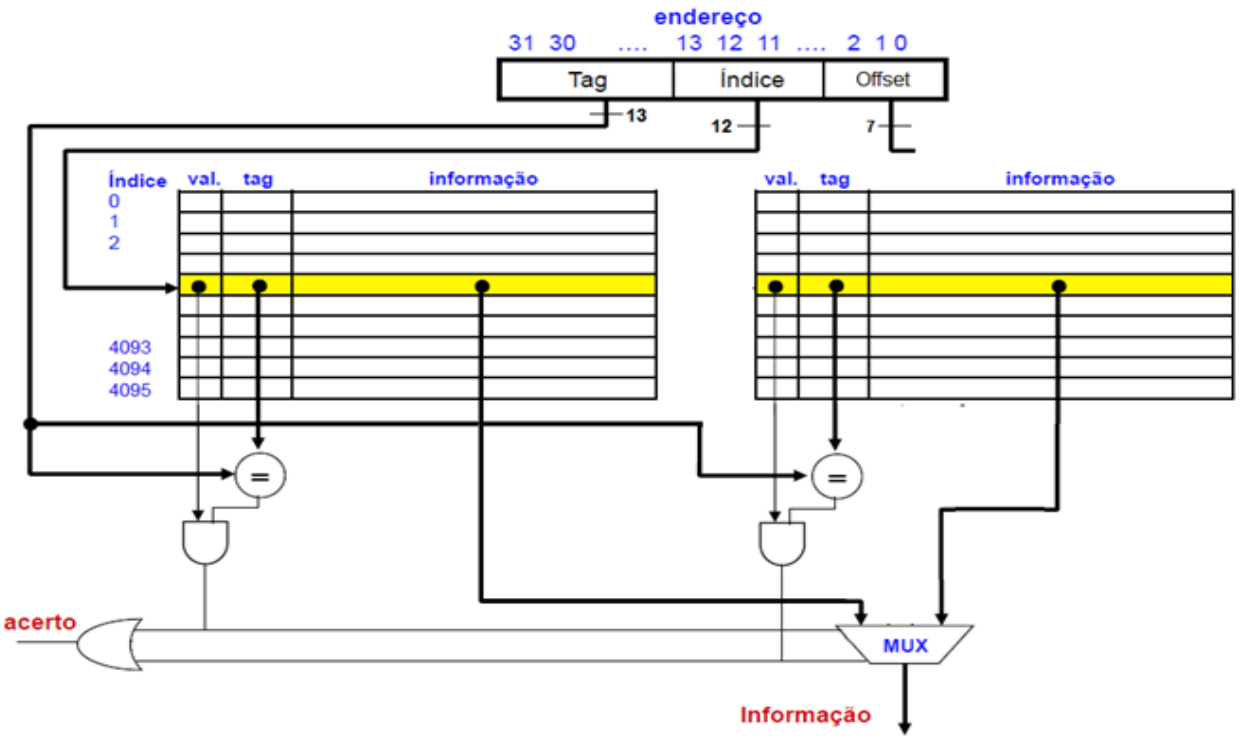
b) Qual o tamanho total desta cache?

Tamanho_total_cache = 8192 * [128 + (13 + 1)] = 1062912 bytes = 8503296 bits (dividiu bytes por 8)

c) Para onde seria mapeado o endereço 912 nesta cache (considere a cache vazia)?

Como 912/128=7,125 e 7mod(4,096) = 7, o endereço 912 seria mapeado para o conjunto 7.

(d) Construa um diagrama desta cache.



21- Qual é o tempo de acesso a cache de um computador cujo tempo médio efetivo de acesso a cada byte da cache é 1.3ns para aplicação Z que quando executada acessa cada byte da Cache 75 vezes?

TCE =TEMPO EFETIVO CACHE

TCE = (N+1 * Tc)/n

N*TCE = N+1*Tc

75*1.3 = 76 * TC = (75*1.3)/76

TC= 1.28ns

22. Através de um experimento científicos com uma aplicação X, provou-se que a cache de um determinado sistema possui um tempo efetivo de acesso à cache de 2ns para aquela aplicação. Sabe-se que o tempo de acesso à cache neste sistema é de 0.5ns e o tempo de acesso à memória principal é de 15ns.

A- Qual é o número de acessos à cache neste experimento?

TCE = TC + TM/N

(TCE - TC)*N = TM

(2ns - 0,5ns) * N = 15ns

1,5nsN = 15ns

N= 15/1,5 = 10

B- Testando o mesmo sistema com uma aplicação Y, obteve-se um tempo efetivo de acesso à cache de 0.8ns. Quantos acessos à cache foram feitos na execução desta aplicação?

(0.8-0.5)*N = 15ns

N = 15/0,3 = 50

C- O que pode-se afirmar sobre as aplicações X e Y? Justifique o que pode provocar sua afirmação.

Que a aplicação Y aproveita muito mais a localidade temporal do que a aplicação X no dado sistema. Assim, possivelmente a aplicação Y utiliza dados semelhantes mais vezes em um curto período de tempo do que a cache X, havendo possivelmente algum laço de instruções na aplicação Y.

D- Se for feito um upgrade neste sistema e a memória principal fosse trocada por uma memória cujo tempo de acesso é de 12ns. Qual seria o tempo efetivo de acesso à cache para a aplicação Y?

(TCE-0.5)*50 = 15ns

TCE = 12/50 + 0,5

TCE = 0,74ns

E- Depois deste upgrade, qual seria o tempo efetivo de acesso à cache se fosse inserido uma nova cache L2 no sistema com tempo de acesso de 2ns?

(TCE-0.5)*50 = 15ns

TCE = 2/50 + 0,5

TCE = 0,54ns

23. Para uma cache de dados que armazena 4096Kb de dados e que possui endereços de 32 bits, armazena duas palavras por bloco e utiliza uma memória endereçada à bytes. Calcule:

Caso seja diretamente mapeada:

(a) Quantos bits são utilizados para o offset, índice e tag?

Já que o endereçamento da memória é feito por bytes e cada bloco contém 64 bytes, são necessários 6 bits para o offset, a fim de permitir a localização de um byte dentro de um bloco.

Número total_blocos = 4096K = 2^12/64 (64b = 2 palavras) = 64k

Nº de Blocos= 64k

Como 64k=2^16, são necessários 6 bits para o índice. Logo, a tag terá um número de bits de:

Nº de bits_tag= 32 - 16 - 6 = 10 bits

b) Qual o tamanho total da cache?

(2 palavras = 64) ((64 + 10 + 1)) * (64k) = 4800kb = 692B

C) Qual foi o aumento no tamanho da cache em %?

14,7%

Caso seja Associativa com conjunto de 4 posições:

(a) Quantos bits são utilizados para o offset, índice e tag?

Número total_blocos = 4096K/64/4=(2^22/64)/4 (4=num conj) = 16k conj

Nº de Conj = 16k

Como 16K=2^14, são necessários 14 bits para o índice. Logo, a tag terá um número de bits de:

Nº de bits_tag= 32 - 14 - 6 = 12 bits

b) Qual o tamanho total da cache?

(2 palavras = 64b) ((64 + 12 + 1)) * (4*16k) = 5568kb

77bits

c) Qual foi o aumento no tamanho da cache em %?

26,4%

24. A área de memória disponível para implementação de uma cache L2 é 256 Kbytes. Considerando que a memória a ser endereçada possui 256 Mbytes (2^{28}) e a cache deve trabalhar com blocos de 8 palavras de 16 bits calcule para a técnica direta, totalmente associativa e conjunto associativa (16 conjuntos):

Assoc 4:

– Divisão de bits do endereço: 21 (Tag) 4 (Conjunto) 3 (Palavra)

– Aproveitamento efetivo da área da cache (relação entre dados e controle):

00% (só dados na cache, controle fica nas MAs)

– Número de linhas da cache:

– Quntidade e tamanho em Kbytes das memórias associativas (quando necessário)

– Divisão de bits do endereço: 21 (Tag [restante?]) 4 (Conjunto [bits pra representar o total de conjuntos/linhas) 3 (Palavra [bits para representar total de palavras])

– Aproveitamento efetivo: 100% (só dados na cache, controle fica nas MAs)

– Número de linhas da cache: Tamanho da linha? Cada linha tem bloco de 8 palavras de 16 bits = 8 * 16 = 128 bits / 8 = 16 bytes

Quantas linhas cabem na cache?

Cache tem 256 Kbytes = 256 * 1024 = 262144 bytes / 16 = 16384 linhas

– Tamanho das memórias associativas: Quantas?

Uma para cada conjunto, ou seja 16

-Tamanho de cada uma?

Cada linha da MA tem tamanho do Tag = 21 bits

O número de linhas de cada MA nessa técnica é igual ao número de linhas do conjunto que ela endereça. Como a cache tem 16384 linhas e são 16 MAs, cada MA endereça 16384 / 16 = 1024 linhas.

Uma MA tem então 1024 (linhas) * 21 (tag) = 21504 bits / 8 = 2688 bytes / 1024 = 2.625 Kbytes

TOTALMENTE ASSOC: 25 TAG 3 PALAVRA

Número de linhas da cache:

Tamanho da linha?

Cada linha tem bloco de 8 palavras de 16 bits = 8 * 16 = 128 bits / 8 = 16 bytes

Quantas linhas cabem na cache?

Cache tem 256 Kbytes = 256 * 1024 = 262144 bytes / 16 = 16384 linhas

– Tamanho das memórias associativas:

Quantas?

Uma única memória associativa

Tamanho da MA?

Cada linha da MA tem tamanho do Tag = 25 bits

O número de linhas da MA nessa técnica é igual ao número de linhas da cache que ela endereça. Como a cache tem 16384 a MA endereça 16384 linhas

Uma MA tem então 16384 (linhas) * 25 (tag) = 409600 bits / 8 = 51200 bytes / 1024 = 50 Kbytes

MAPEAMENTO DIRETO:

11 tag + 14 indice + 3 palavra

com 14 bits para linha (podendo endereçar 16384 linhas da cache)

– Divisão de bits do endereço:

Tamanho da linha = 1+11 (Tag)+128 = 140 bits / 8 = 17.5 bytes

Quantas linhas cabem na cache?

Cache tem 256 Kbytes = 256 * 1024 = 262144 bytes / 17.5 = 14979 linhas

Cache tem 14979 linhas e Tag = 28 (endereço) – 3 (palavra) – 14 (linha) = 11

– Divisão de bits do endereço:

11 (tag) 14 (linha) 3 (Palavra)

– Aproveitamento efetivo:

Dados em cada linha: um bloco de 8 palavras de 16 bits = 128 bits

Tamanho total da linha: 1 (validade) + 11 (Tag) + 128 (bloco) = 140 bits

Percentual de aproveitamento: 140 -> 100%

128 -> ?% (uso para dados)

Aproveitamento efetivo = 128 * 100 / 140 = 91.42%

– Tamanho das memórias associativas:

Não utiliza MAs nessa técnica

https://www.inf.pucrs.br/~flash/orgarq/aulas/memoria

25. Considerando uma cache unificada de 32KB que apresenta 60 faltas para cada 1000 instruções em um processador com clock de 2GHz. Considere que existe 1,5 acessos a memória para cada instrução, que um acerto utiliza 1 ciclo de clock e a penalidade de falta é de 100 ciclos para o acesso a memória principal.

(a) Calcule o tempo médio de acesso à memória e desempenho do processador neste sistema.

Taxa_de_faltas = (60/1000)/1,5=0,04

Período_clock = 1/(2.10^9)=0,5ns

TMAM = ($ciclo_{acerto}$ + taxa_falta * $Ciclo_{Erro}$) * $Periodo_{Clock}$

TMAM=[1+(0,04*100)] * 0,5ns = 2,5ns

Tempo_processador = [CPI.I+1,5I*(0,04)*100]*0,5ns = [CPI.I+6.I]*0,5ns = I(0,5CPI+3)ns

(b) Calcule o ganho no tempo médio de acesso a memória, e o desempenho do processador, com a inserção de uma cache de segundo nível, que apresenta 30 faltas a cada 1000 instruções e tempo de acerto de 10 ciclos. Considere que a penalidade de falta da cache de segundo nível também é de 100 ciclos para o acesso a memória principal.

Taxa = taxaL2 / taxaL1

Taxa_falta_local_L2 = 30/60 = 0,5

TMAM={1+0,04*[10+0,5*(100)]}*0,5ns = 1,7ns

Ganho_TMAM= 2,5/1,7 = 1,47

Tempo_processador = [CPI.I + 1,5I*{ 0,04*[10 + 0,5*(100)]}] * 0,5ns = [CPI.I+3,6.I] * 0,5ns = I(0,5CPI+1,8)ns

26. Considerando uma cache unificada de 32KB que apresenta 80 faltas para cada 900 instruções. Considere que existe 1,7 acessos a memória para cada instrução, que um acerto utiliza 1 ciclo de clock e a penalidade de falta é de 120 ciclos para o acesso a memória principal e que o tempo médio de acesso à memória é de 3,7ns.

(a) Calcule o período de clock.

Taxa_de_faltas = (80/900)/1,7=0,05

$TMAM = (ciclo_{acerto} + taxa\_falta * Ciclo_{Erro}) * Periodo_{Clock}$

3,5 = (1 + 0,05 * 120) * P

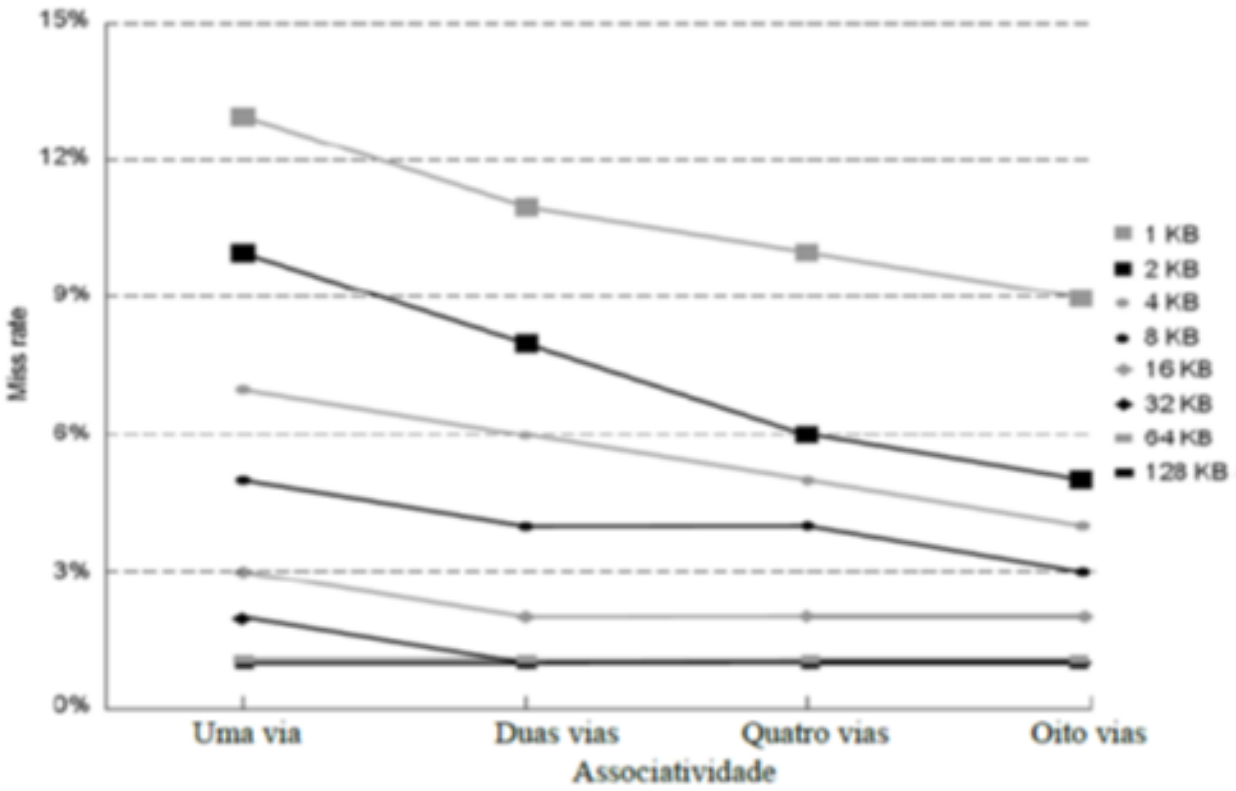
P = 3,5 / (1+0,5*120) = 0,5ns

b)Qual é o tempo total de processamento assumindo que o processador executou 10000 instruções e levou em média 1,4 ciclos por instrução?

Tempo_processador = [CPI.I+1,7I*(0,05)*120]*0,5ns = [CPI.I+10,2.I]*0,5ns = I(0,5CPI+5,1)ns = I(0,5 * 1,4 + 5,1) = 5,8ns *10000 = 58000ns

27. O gráfico abaixo apresenta os resultados médios de taxa de miss para diferentes tamanhos de cache, utilizando diferentes níveis de associatividade.

(a) Explique o comportamento das curvas do gráfico abaixo:



Quando a associatividade é de uma via, temos uma cache de mapeamento direto – neste caso, o conteúdo de cada posição da memória principal só pode ser mapeado para uma única posição da cache; como consequência, várias (ou muitas) posições da memória serão mapeadas para a mesma posição da cache, o que favorece o surgimento de faltas por conflito de dados, produzindo, assim, taxas de faltas mais elevadas. Porém, à medida que o grau de associatividade aumenta, cada conjunto da cache possui duas ou mais posições, de modo que cada posição da memória principal é mapeada para um conjunto da cache; como cada conjunto possui mais de uma posição, existe mais de uma possibilidade de posição de mapeamento de determinada posição de memória para o respectivo conjunto da cache, o que faz reduzir o índice de faltas por conflitos de dados e, portanto, provoca a redução da taxa de faltas. Portanto, quanto maior for o grau de associatividade da cache, mais posições haverão por conjunto da cache para o mapeamento de posições da memória principal, diminuindo, assim, a taxa de faltas.

Com relação ao tamanho da cache, quanto maior for o tamanho da cache, mais posições haverá na mesma para o mapeamento de conteúdos de posições da memória principal; com isso, a possibilidade de faltas na cache por conflitos de dados diminui, pois será mais rara a disputa por um bloco da memória cache para mapear uma posição da memória principal. Além disso, o índice de

faltas por capacidade também diminui à medida que o tamanho da cache aumenta. Logo, o aumento do tamanho da cache faz com que a taxa de faltas, em geral, diminua.

(b) Explique porque as curvas para as caches de 64K e 128K apresentam um comportamento diferente das demais curvas.

Quanto maior for o tamanho da cache, menor será a taxa de faltas devidas a capacidade da cache e a conflitos de dados. Por isso, para as caches de 64 e 128 KB do gráfico, o aumento do grau de associatividade (que também contribui para a diminuição da taxa de faltas) faz diferença para até duas vias; com um número maior de vias, o benefício do aumento da associatividade já não se destaca tanto em face das vantagens supracitadas referentes ao aumento do tamanho da cache; por conta disso, quanto maior for o tamanho da cache, menos o aumento do grau de associatividade pesa para a diminuição da taxa de faltas.

28 - Explique o possível motivo do comportamento das curvas de instruções e dados terem comportamentos distintos entre a Figura 1 e 2:

As curvas diferem possivelmente pelos diferentes comportamentos dos programas. Através delas podemos deduzir que GCC_1 é um programa possivelmente sequencial que utiliza dados distintos, pois, além de possuir um miss rate alto, os dados ainda conseguem ter um grande aproveitamento das vantagens provenientes da associatividade.

Já a aplicação JPEG_1 deve possivelmente acessar os mesmos dados e instruções muitas vezes devido a loops, fazendo assim com que o miss rate seja baixo e que a associatividade tenha muito impacto. Ainda pode-se inferir que JPEG_1 utiliza uma grande quantia de dados para as mesmas instruções, já que ao contrário de GCC_1 o miss rate de dados é sempre maior do que a de instruções.

Esta diferença em ambos programas justificaria o comportamento dessemelhante das curvas.

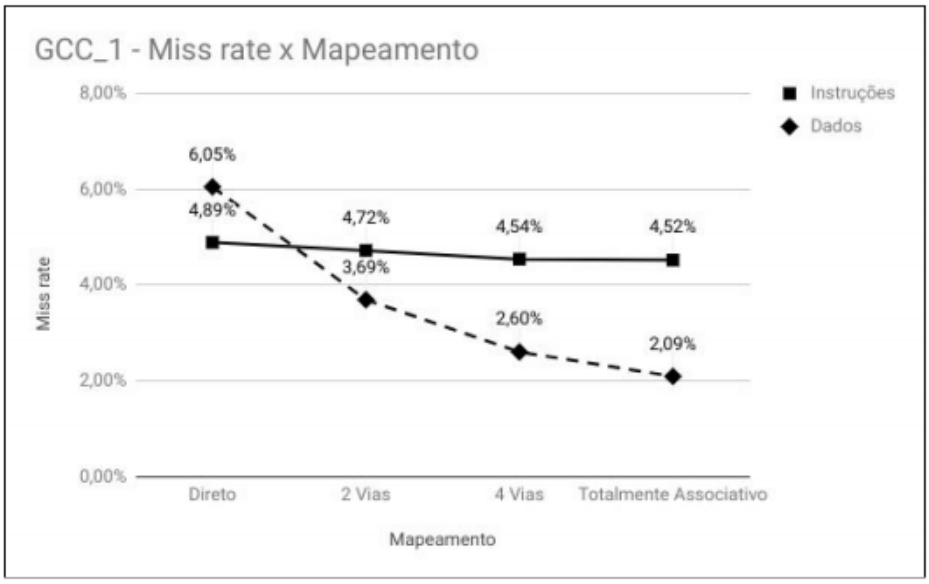


Figura 1. Miss rate para os mapeamentos direto, 2 vias, 4 vias e totalmente associativo com o benchmark GCC_1.

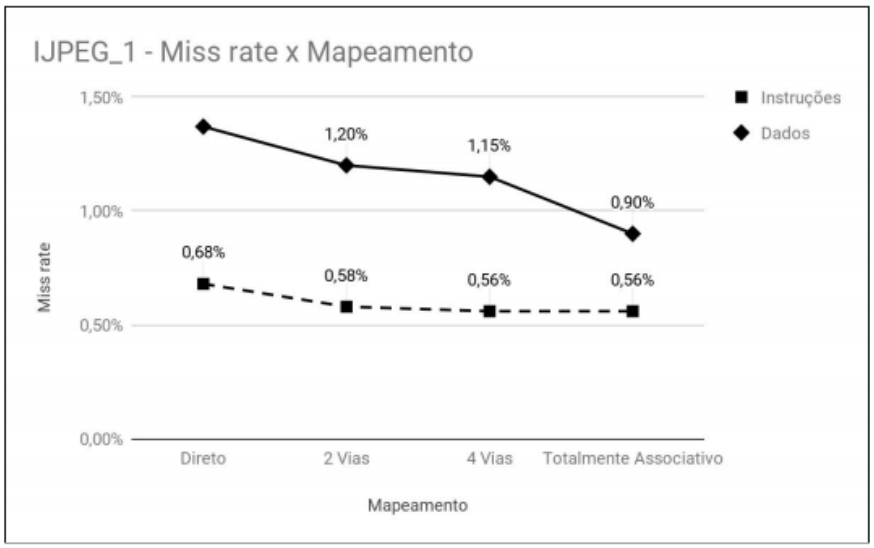


Figura 2. Miss rate para os mapeamentos direto, 2 vias, 4 vias e totalmente associativo com o benchmark IJPEG_1.

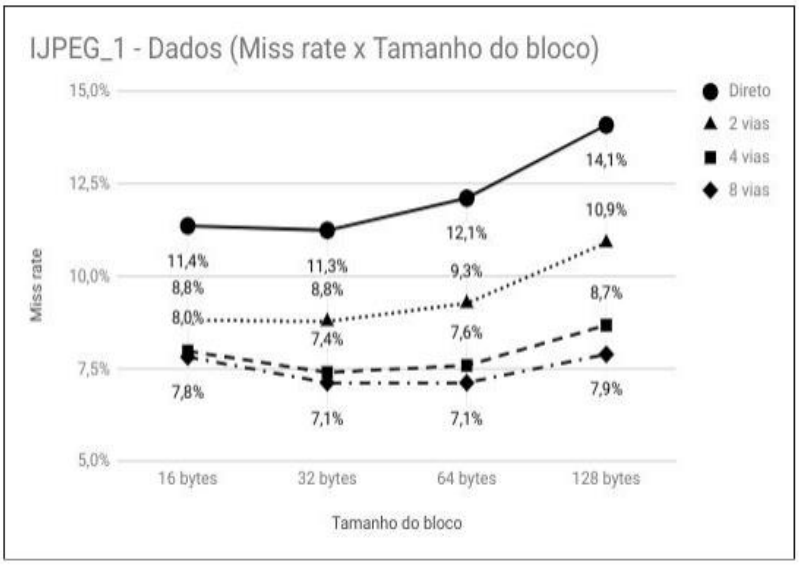
29- Dados os gráficos a seguir, responda:

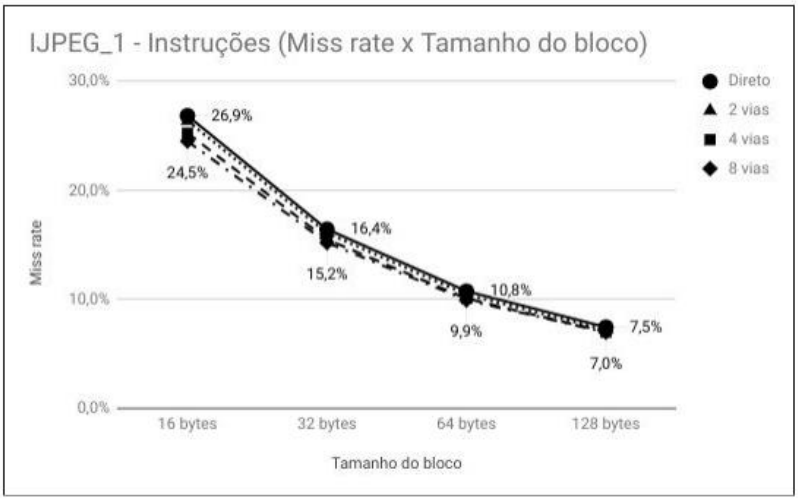
a. **Porquê aumentar o tamanho da cache acabou aumentando a taxa de miss de dados?**

Quando se aumenta o tamanho do bloco se busca mais palavras ao mesmo tempo, contudo, os blocos acabam sendo alterados antes de todas suas palavras serem consultadas. Causando assim um maior número de misses.

a. **Porque o mesmo comportamento não ocorreu para os acessos de instruções?**

Porque o código é possivelmente sequencial, aproveitando bastante a propriedade da localidade espacial da cache.

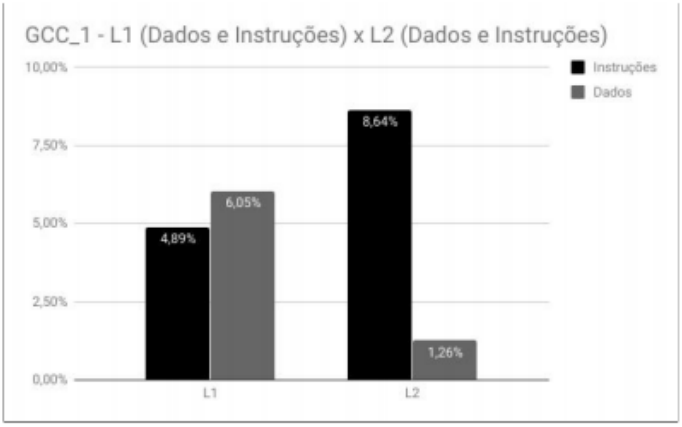


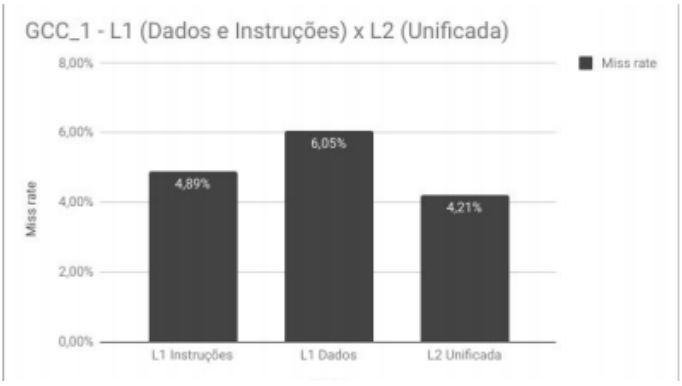


30- Analisando ambas figuras abaixo, explique na sua opinião se utilizar uma cache L2 unificada é mais eficaz do que ter uma para cada tipo de acesso. Explique também o porque a taxa de miss da unificada não é uma média entre as taxas da L2 de dados e instruções.

Caches separadas são mais complexas de serem projetadas e são melhor aproveitadas em sistemas superescalares, o que não é o caso neste experimento. Desta forma, pode-se afirmar que seria mais eficaz utilizar-se uma cache unificada para o sistema sendo utilizado, pois a utilização de caches separadas acabou causando uma alta taxa de miss para instruções, fazendo com que o ganho em utilizá-la não seja expressivo o suficientes para justificar o custo extra.

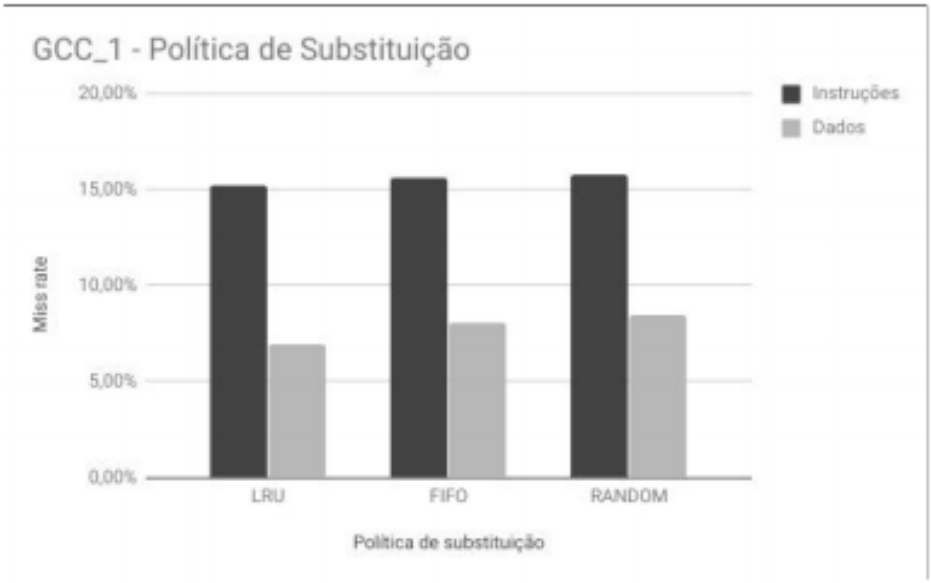
b) Uma cache unificada por lidar com ambos tipos de informações tende a balancear naturalmente a demanda do processador enquanto também consegue explorar ambos os tipos de localidade. Estas características causadas pela junção de ambas informações fazem com que as interações dentro da memória sejam distintas e aumentem o hit rate, causando assim uma assimilaridade entre os hits e misses de ambas técnicas.

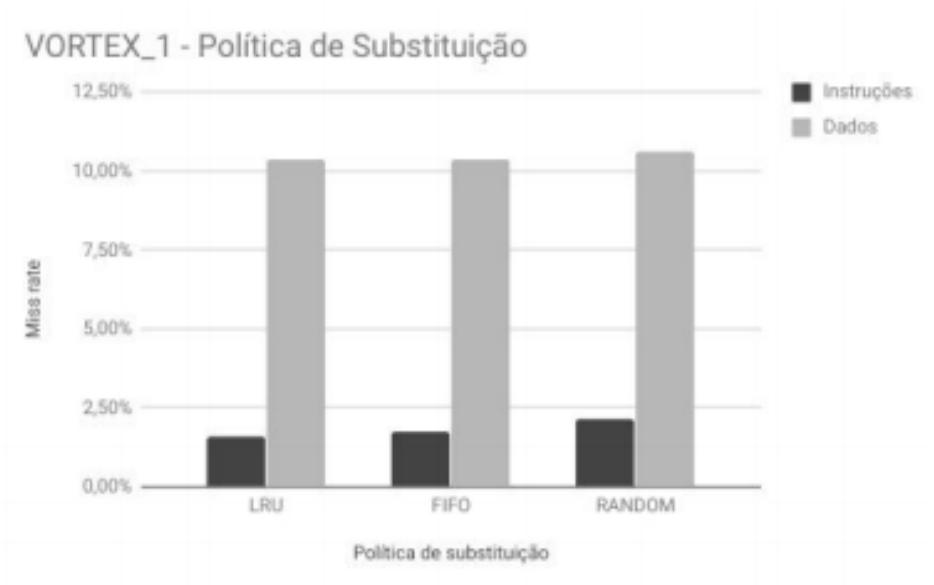




31- Explique os possíveis motivos para as políticas de substituição causarem maior miss de instruções para a aplicação GCC_1 e de dados para a Vortex_1. Ainda, justifique o porquê pode ser desejável utilizar a política FIFO ou RANDOM sendo que a LRU provê melhores resultados em ambos cenários.

Possivelmente a aplicação GCC_1 possui muitas instruções distintas para os uma pequena quantidade de dados enquanto Vortex_1 possui loops que utilizam muitos dados diferentes. Pois desta forma enquanto GCC_1 trocaria muitas instruções na memória (devido aos misses) Vortex_1 teria que trocar muitos dados.
Apesar dos ganhos da LRU serem maiores, o ganho é muito pequeno para justificar o custo extra implicado pela técnica. Sendo assim, LRU seria desejável apenas em projetos onde ter um baixo miss rate seja crucial. Desta forma poderia ser desejável utilizar as políticas FIFO ou até mesmo RANDOM devido suas baixas complexidades e ganhos similares.




VORTEX_1 - Política de Substituição
12,50%
10,00%
7,50%
5,00%
2,50%
0,00%
Miss rate
LRU
FIFO
RANDOM
Política de substituição
Instruções
Dados