Homework 2

Alunos: Lucas Marçal Coutinho, Lucas Mattos Vieira e Loredana Romano Devico.

Códigos de Matrícula: 11911BCC012, 11911BCC015 e 11911BCC001 (respectivamente).

Questões:

P1) Um dos objetivos que orientam o projeto de um sistema de memória é alcançar capacidade de armazenagem, com desempenho aceitável a um custo razoável. Como este objetivo pode ser alcançado? Faça um esboço desta estrutura?

- O tempo de acesso à memória pode ser considerado um dos maiores gargalos da computação e, para compensar esse gap entre processador-memória, foi criada uma estrutura de hierarquia de memória. Separada em níveis, o mais alto é o mais próximo do processador, os registradores. Descendo, temos os níveis de cache e logo em seguida a memória principal, a DRAM. Depois, vem a memória externa, que é um disco rígido fixo e pode ser seguida por discos ópticos e fitas.

Memória na placa, fora da placa e off-line:

REGISTRADORES
MEMÓRIA CACHE
MEMÓRIA PRINCIPAL
DISCO MAGNÉTICO
CDS E DVDS
FITA MAGNÉTICA

P2) O acesso à memória é um enorme gargalo em todos os computadores modernos porque o tempo de ciclo da CPU é muito superior ao tempo de acesso à memória. A Figura1 mostra a performance da memória e da CPU no período 1980-2010. Dessa forma, uma das maneiras de melhorar o desempenho de computadores atuais é evitar acessar a memória principal. O tempo de acesso à memória principal é alto e degrada o desempenho do sistema computacional. A estratégia de usar memória cache tenta suprir essa lacuna existente no desempenho relativo CPU/Memória e assim o uso de memória cache tornou-se um dos principais temas a ser analisado no projeto de um sistema computacional.

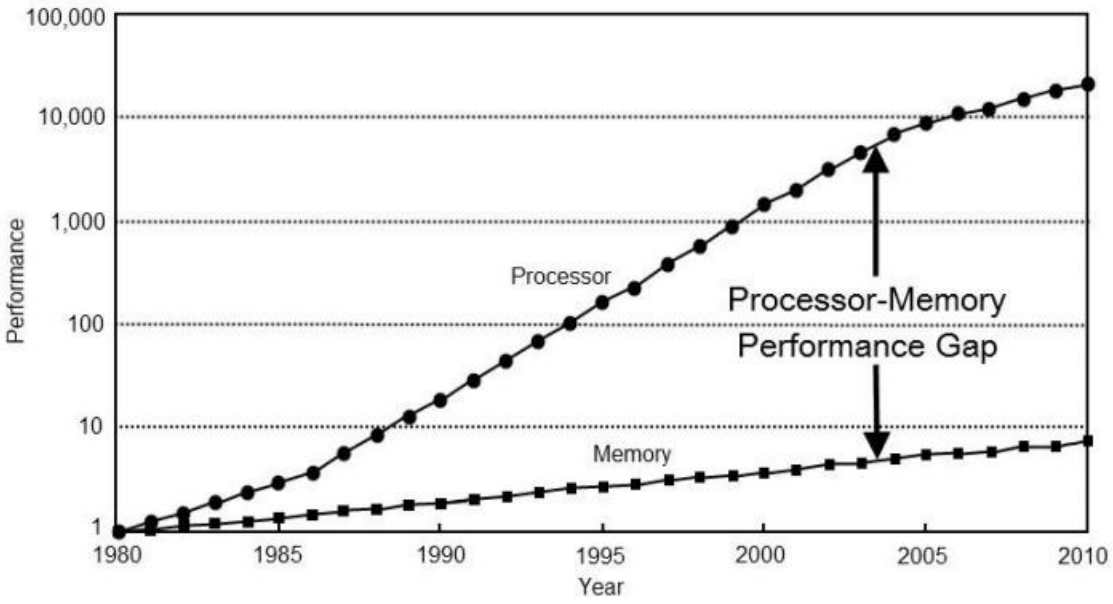


Figura 1-CPU and Memory performance gap.

Considerando tais informações e seus conhecimentos a respeito do projeto de memória cache, responda as questões a seguir:

a) Apresente uma discussão mostrando os *trade-offs* envolvidos nos seguintes aspectos relacionados ao projeto de memória cache: tamanho da cache, tamanho da linha e associatividade. Sugestão: faça uma tabela mostrando como esses aspectos impactam no tempo de acesso, na penalidade por falta e na taxa de acerto. Escreva um parágrafo explicando a tabela.

Aspecto	Descrição	Impacto no Tempo	Penalidade por Falta	Impacto na Taxa de Acerto
Tamanho	Deve ser pequena para que o custo médio por bit seja próximo do custo da memória principal e grande o suficiente para o tempo médio ser próximo do tempo de acesso da memória cache.	O tempo influenciará na penalidade de falha, pois haverá latência até a primeira palavra e o tempo de transferência, além de este aumentar conforme o tamanho do bloco.	Há um aumento da penalidade conforme os blocos ficam maiores; ocorre um stall semelhante ao pipeline.	Com blocos maiores, exploram a localidade espacial e diminuem taxas de erro. Porém, a taxa pode subir se o tamanho do bloco tomar uma fração significativa da cache.
Tamanho da Linha	A princípio, quando se aumenta o tamanho, a taxa de acerto aumenta devido ao princípio da localidade e dados mais úteis são trazidos. Porém, a razão de acerto vai diminuindo e a probabilidade de reuso diminui.	Devido à "previsão" de onde o dado estará, o trabalho de busca da CPU é acelerado. Porém, caso haja muitos blocos com o mesmo código, haverá conflito de posição e reduzindo desempenho.	Com a possível existência de muitos blocos com o mesmo código e com a taxa de acerto reduzindo com o tamanho, há uma penalidade considerável.	A princípio, o aumento do tamanho aumenta também a taxa de acerto; porém, ao aumentar mais, a razão de acerto vai diminuindo.

Associatividade	Permite que cada bloco da memória seja carregado para qualquer linha, usando um rótulo e um campo de palavra, com maior flexibilidade, mas maior complexidade para comparação de rótulos.	Eficiência no momento de alocação, porém, desvantagem ao buscar um elemento. Porém, ainda assim pode ser mais eficiente que o Mapeamento Direto, por precisar ir menos à Memória Principal, que é lenta.	A complexidade do circuito para comparar as tags de todas as linhas da cache em paralelo, com, ao estar cheio, necessária substituição, aumenta a penalidade.	Com os algoritmos de substituição, há um elevado aumento na taxa de acerto.

b) As CPU's modernas apresentam pelo menos dois níveis de cache. Considerando uma CPU com dois níveis de cache, com cache L1 (nível 1) e a cache L2 (nível 2). O tamanho da cache L2 é maior que o tamanho da cache L1. Apresente duas justificativas para a cache L2 ser maior que a cache L1.

- A memória cache é dividida em níveis que dizem respeito à proximidade com o processador; quanto mais próxima, menor seu nível e mais rápido sua execução. Um dos fatores que contribui com a velocidade é o tamanho, o cache L1 então com o menor tamanho, é o mais rápido e com menor espaço. Mas o L1 é muito pequeno para agilizar os múltiplos processos de execução, necessitando de outros níveis, estes mais distantes do processador, e podendo ser maiores e relativamente mais devagar.

Assim, seguindo também uma certa hierarquia, (1) cache L1 é menor porque seu trade-off é ágil, sendo menor em tamanho e mais rápido em execução e porque (2) é o mais próximo do processador e, para "passar a ideia de uma memória rápida", deve ser rápido também.

P3) Qual é o significado dos termos **byte-endereçável** ou **palavra-endereçável**?

- A memória do computador é dividida em células, que são a menor unidade endereçável (ou seja, a menor localização endereçável) e o endereçamento pode ser feito em palavras ou em bytes, dependendo da memória, sendo expressos no sistema binário. Isso significa que cada byte ou palavra tem seu próprio endereço exclusivo e pode ser acessado, variando a quantidade de bits usados para índice, tag e deslocamento.

P4) O que significam ***princípio da localidade da referência*** e ***princípio da inclusão*** em organização hierárquica de memória?

- O princípio da localidade afirma que os dados na vizinhança de uma palavra referenciada têm altas chances de serem referenciados no futuro próximo. Já a Propriedade de Inclusão significa que os itens armazenados no nível mais externo e, durante o processamento, subconjuntos são copiados para o nível inferior de maneira sucessiva. Assim, um dado encontrado em N3 terá cópias nos níveis superiores N4, N5... Nm.

P5) Uma medida dos benefícios de diferentes organizações de cache é a taxa de falta (*miss rate*). Onde taxa de falta é a fração de acessos a cache que resultam em faltas. Denomine e descreva quais são as três categorias associadas as **causas** de falta de cache (*miss cache*):

A Taxa de Falta possui três categorias:

- Faltas Compulsórias: ocorreriam com a primeira referência a um bloco, falhas no acesso, causadas pelo acesso que nunca esteve na cache. É reduzida pela capacidade.

- Faltas por Capacidade: quando um cache não comporta todos os blocos necessários para a execução. É reduzida pelo tamanho do bloco (menor número de blocos pode aumentar conflito, mas o bloco maior também pode aumentar).

- Faltas por Conflito: ocorreriam mesmo com cache não-cheia, mas que não ocorreriam se a cache fosse totalmente associativa. Ocorrem por colisões no mapeamento de blocos na cache, quando diversos blocos competem pelo conjunto. Pode ser reduzida pela associatividade.

P6) Um sistema de ***cache*** tem uma taxa de acerto de 75%, um tempo de acesso de 10ns quando o dado for encontrado no ***cache*** e um tempo de acesso de 100ns se o ***cache*** não contiver o dado. Qual é o tempo de acesso efetivo?

hit-time + (1-hit-rate) * miss-penalty = Tempo de Acesso Efetivo

10ns + (1-0.75)*100 = 10 + 25 = 35ns

Resposta: O tempo efetivo é 35ns.

P7) Por que é tão difícil construir um *cache full associative*?

- O full associative permite que os blocos sejam colocados em qualquer linha, e a tag passa a ser constituída por todos os bits do endereço, menos o índice. Com isso, reduz o número de colisões, mas aumenta o número de bits usados pela tag e o processador tem que procurar em todas as linhas da cache. Além disso, há um consumo alto de bits para o endereço. Com isso, entende-se que o custo em tempo é alto, assim como no processamento.

P8) Para esta tarefa, você deve assumir que a memória principal consiste em 128 palavras (*words*) dividida em blocos de 4 palavras cada (32 blocos). A memória cache consiste em 8 linhas cache (*slots*).

a) Desenhe um diagrama de blocos, similar aquele mostrado em aula, que ilustra esta configuração de memória.

128 palavras, 32 blocos, 4 palavras por bloco → 2^7 → Endereço de tamanho 7

Cache = 8 slots

	Tags	1	2	3	4
0					
1					
2					
3					
4					
5					
6					
7					

RAM = 32 blocos de 4 palavras

	0	1	2	3
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				

b) Assumindo que uma função de mapeamento direto é utilizada

i) Para cada slot no cache, liste os blocos que seriam mapeados para o slot.

0 = 0, 8, 16, 24

1 = 1, 9, 17, 25

2 = 2, 10, 18, 26

3 = 3, 11, 19, 27

4 = 4, 12, 20, 28

5 = 5, 13, 21, 29

6 = 6, 14, 22, 30

7 = 7, 15, 23, 31

ii) Quantos bits são necessários para um endereço? Liste os bits e indique qual parte do endereço que seriam utilizados para (palavra, etiqueta, etc.).

4 palavras → 2 bits (palavra)

8 linhas → 2^3 → 3 bits (slot)

2+3 = 5 → 2 bits para a tag.

TAG	SLOT	WORD
2	3	2

iii) Qual é o endereço para as seguintes palavras: 4, 13, 39 e 89.

palavra	v. bin	tag	slot	word
4	0000100	00	001	00
13	0001101	00	011	01
39	0100111	01	001	11
89	1011001	10	110	01

c) Assumindo que uma função de *mapeamento associativo* é utilizada

i) Para cada slot no cache, liste os blocos que seriam mapeados para este slot (atenção).

- Por ser associativo, qualquer bloco pode ser mapeado para qualquer slot, sendo justamente esse o objetivo do mapeamento associativo. Contudo, caso a memória esteja cheia, é necessário usar alguma política de substituição.

ii) Quantos bits são necessários para um endereço? Liste os bits e indique qual parte do endereço que seriam utilizados para (palavra, etiqueta, etc.).

5 (tag)	2 (word)

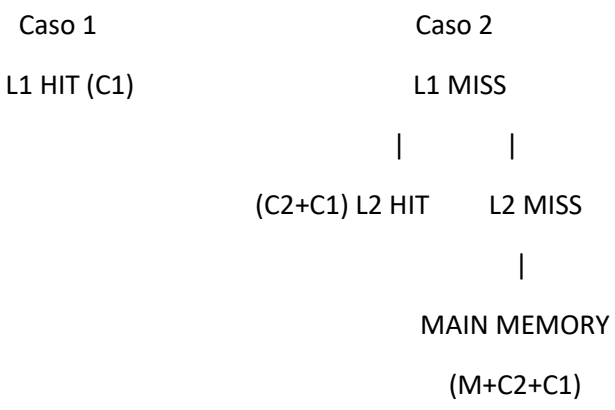
iii) Qual é o endereço para as seguintes palavras: 5, 11, 29 e 99.

palavra	v. bin	tag	word
5	0000101	00001	01
11	0001011	00010	11
29	0011101	00111	01
99	1100011	11000	11

iv) Mostre o estado do cache após a CPU ter requerido as seguintes palavras, se o algoritmo de substituição de blocos *Menos Recentemente Utilizado* (LRU) for empregado: 3, 7, 15,16,17,28,30,56,57,58, 78,79,20,21,23, 98,99,24, 25 e 26.

palavra	bin	tag	word
3	0000011	00000	11
7	0000111	00001	11
15	0001111	00011	11
16	0010000	00100	00
17	0010001	00100	01
28	0011100	00111	00
30	0011110	00111	00
56	0111000	01110	00
57	0111001	01110	01
58	0111010	01110	10
78	1001110	10011	10
79	1001111	10011	11
20	0010100	00101	00
21	0010101	00101	01
23	0010111	00101	11
98	1100010	11000	10
99	1100011	11000	11
24	0011000	00110	00
25	0011001	00110	01
26	0011010	00110	10

P9) Escreva a fórmula do tempo médio de acesso a memória (***Average Memory Access Time - AMAT***) quando há dois níveis de cache L1 e L2.



AMAT = Hit TimeL1 + (Miss RateL1 x (Hit Time L2 + Miss RateL2 x Miss PenaltyL2))

= Hit TimeL1 + (Miss RateL1 x (Hit Time L2 + Miss RateL2 x Hit Time Main Memory))

P10) Por que um cache de segundo ou terceiro nível é **útil**? Que tipo de benefícios proporciona ao desempenho do sistema?

- O cache em níveis foi criado considerando o aumento na velocidade dos dispositivos, especialmente no processador. Pela alta demanda, seria necessário um cache de alta velocidade e capacidade, mas pelo custo é preferível desenvolver níveis que diferem na relação tamanho X velocidade. Assim, o L1 é rápido, porém muito pequeno, necessitando do L2, que é maior porém com velocidade menor que L1. O L3, ainda menos utilizado, vem para auxiliar ao processador acessar cada vez menos a memória principal, lenta e com o maior tamanho, aumentando o clock do processo.

a) Explique como isso afeta o cálculo do tempo médio de acesso à memória (***AMAT***).

- O AMAT é afetado, como visto no exercício nove, porque deve considerar as taxas de acerto e erro de cada nível. O caso mais favorável é o Hit no L1, porém, caso erre e esteja no L2, temos Miss L1 e o Hit no L2. Contudo, se não estiver em nenhum dos dois, deve-se ir à Memória Principal e considerar a taxa M. Para o cálculo, que deve considerar todas as possibilidades, há três cenários possíveis.

b) Quais outras técnicas de aprimoramento de cache podem fornecer uma solução para o mesmo problema?

- Acredito que o Smart Cache, da Intel, desenvolvido para multicore, é uma solução viável por dividir a memória cache real entre os núcleos do processador. Com isso, a taxa de falta diminui, pois nem todos os núcleos precisam de partes iguais do espaço. Isso permite que um núcleo utilize o cache L2 ou L3 se outros estiverem inativos, além da partilha aumentar a rapidez do processo.

P11) Vamos considerar um computador com um cache L1 e uma hierarquia de memória cache L2. Suponha que em 1000 referências de memória haja 80 erros em L1 e 40 em L2.

- 1000 referências, 80 erros em L1 e 40 erros em L2.

a) Quais são as consequentes taxas de falta de cache?

- Taxa de falta:

Miss Rate (L1) → Taxa de Falhas de L1 = 80/1000 = 0.08 = 8% (Global)

Miss Rate (L2) → Taxa de Falhas de L2 = 40/1000 = 0.04 = 4% (Global)

b) Suponha que o tempo de acerto L1 = 1 ciclo de clock; Tempo de acerto L2 = 20 ciclos de clock; Miss Penalty L2 = 200 ciclos de clock; Acessos a memória por instrução = 150%. Qual é o tempo médio de acesso à memória (AMAT)?

- T1 = 1 ciclo, T2 = 20 ciclos, C2 = 200, Acesso = 150%. AMAT?

AMAT = Hit TimeL1 + Miss RateL1 x (Hit Time L2 + Miss RateL2 x Miss Penalty L2)

AMAT = 1 + 0.08 * (20 + 0.50 * 200) = 10.6 ciclos de clock

RESPOSTA: 10.6 ciclos de clock

P12) O tempo médio (esperado) para acessar a memória (AMAT - *Average Memory Access Time*), pode ser calculado pela seguinte formula:

AMAT = hit time + (miss rate × miss penalty)

Lembre-se que a taxa de penalidade (*miss penalty*) é o tempo adicional que é consumido para acessar a memória em um evento de falta de cache (*cache miss*). Por essa razão, uma falta de cache é calculada como (*hit time + miss penalty time*).

Considere um sistema de memórias cache com as seguintes propriedades. Responda, qual é o tempo de acesso médio a esse sistema de memória (AMAT)?

- L1$ hits in 1 cycle (local miss rate 45%)
- L2$ hits in 10 cycles (local miss rate 30%)
- L3$ hits in 50 cycles (global miss rate 10%)
- Memória principal hits in 100 cycles (always hits)

AMAT = 1 + 45% * (10 + (30% * 50 + (10% * 100)))

= 1 + 0.45 * (10 + (0.3 * (50 + 10))) = 1 + 0.45 * (10 + (0.3 * 60))

= 1 + 0.45 * (10 + 18) = 1 + 0.45 * 28

= 1 + 12.6 = 13.6 ciclos.

RESPOSTA: 13.6 ciclos.

P13) Um ***cache*** está sendo projetado para um computador com 232 bytes de memória. O ***cache*** terá 2K slots e usará um bloco de 32 bytes. Calcule, tanto para um ***cache associativo*** quanto para um ***cache com mapeamento direto***, quantos bytes o ***cache*** precisará ter capacidade de armazenar.

- 2^32 bytes → 32 é o tamaho do endereço. Cache com 2k slots → 2028 linhas → 2^11 → 11 slots; 32 bytes → 2^5 → 5 é o índice do byte.

Assim, 2^11+5 = 2^16 = 65536 → 65 kBytes. O tipo de mapeamento utilizado não influencia no tamanho da cache.

P14) Projete e desenhe um diagrama detalhado de uma memória cache com capacidade de 1 Mbytes, mapeamento direto (associatividade unária), 8 palavras por bloco e escrita preguiçosa. O processador emite endereços de 32 bits.

a) Indique como um endereço é interpretado pelo controlador da cache.

b) Repita o exercício, considerando associatividade octa-ária (***8-way set-associative***). Indique o mecanismo de substituição de blocos num mesmo conjunto.

P15) Apresente quais técnicas poderíamos empregar no projeto de organizações hierárquicas de memória para reduzir a penalidade de uma falta (***miss penalty***)?

- Diminuir o Miss Penalty é diminuir a espera da CPU, deixando-a menos ociosa. A melhor maneira é o uso de mutil-níveis de cache (sendo o L1 que permite redução do miss penalty e hit time); há também o aumento da largura de banda entre o cache a memória, que permite o acesso paralelo a mais de uma palavra por bloco. Os bancos de memória de leitura/escrita de múltiplas palavras (chamados Memória Interleaved) também reduzem, pois o endereço é enviado simultaneamente para os diferentes bancos.

Algumas técnicas seriam:

- Blocos de memória buscados durante um miss causam interrupção da execução da CPU, que recomeça quando o bloco é colocado na cache;

- Early restart: A palavra é enviada para a CPU assim que o bloco que contém a palavra for carregado para a cache.

- Critical Word First: Busca a palavra procurada primeiro e envia para a CPU assim que for carregada.

P16) Considere o seguinte código

```
for (i=0; i<20; i++)
        for ( j=0; j< 10; j++)
                a[i] = a[i]*j;
```

a) Dê um exemplo de localidade espacial no código.

- O Princípio da Localidade Espacial diz que há uma probabilidade de acesso maior para dados e instruções em endereços próximos àqueles acessados recentemente.

Um exemplo disso no código seria o acesso em sequência dos elementos do array *a* no loop (que estão armazenados em sequência na memória).

b) Dê um exemplo de localidade temporal no código.

- O Princípio da Localidade Temporal diz que um dado acessado recentemente tem mais chances de ser usado novamente, do que um dado usado há mais tempo.

Um exemplo disso no código seria o uso da variável *i* tanto como contador do loop mais externo quanto seu uso no loop interno para acessar os itens do array *a*.

P17) Considere o programa de multiplicação de matrizes abaixo. As matrizes contêm 1024x1024 elementos, cada elemento um *double* (8 bytes). O programa é executado num único processador, num sistema de memória virtual com páginas de 4 Kbytes. Existe uma cache primária com 64 Kbytes e uma cache secundária com 1 Mbytes. Os blocos de cache primária têm 32 bytes de largura, e os blocos da cache secundária 64 bytes. Descreva o comportamento da hierarquia de caches durante a execução deste programa.

```
for (i = 0; i < 1024; i++) {
        for (j = 0; j < 1024; j++) {
                sum = 0.0;
                for (k = 0; k < 1024; k++)
                        sum += a[i][k] * b[k][j];
                c[i][j] = sum;
        }
}
```

- Devido ao tamanho, a matriz e seus dados serão armazenados, assim como o tamanho dos níveis L1 e L2. Ao rodar o programa, como todas as variáveis se encontram na memória principal, cada valor necessário irá para L2 (por ser um nível mais próximo da memória principal, seu tamanho é maior que L1 (64 bits)) pois pode armazenar vários valores do tipo double.

O cache L1 armazenará os resultados das operações entre os dados em L2, que será o valor a ser apresentado. Desta forma, temos:

a[i][k], b[i][k] e c[i][k] → L2

sum += a[i][k] * b[i][k] → L1

Após cada operação, o valor em L1 passará para L2 e será armazenado em c[i][k], atualizando então na memória principal.

P18) Considere duas organizações de memória cache. A **primeira** é estruturada como um **cache de 32 KB, 2-way set associative, blocos de 32-bytes**. A **segunda** organização é estruturada como um **cache de 32 KB, mapeamento direto, blocos de 32-bytes**. O tamanho do **endereço** de ambas é de **32 bits**. Considere que um multiplexador 2/1 tenha uma latência de 0.6ns, enquanto um comparador de kbits tem uma latência de k/10ns. A latência de um ***cache hit*** (acerto) na organização associativa por conjuntos é definida por **H1**, enquanto a latência do ***cache hit*** da organização com mapeamento direto é definida por **H2**. Determine o valor de **H1**.

P19) Considere o fragmento de código a seguir, armazenado na memória a partir do endereço 0x1000100C. Todas as instruções na máquina alvo possuem 4 bytes de tamanho:

```
loop:
        lw   $r2, 0($r0)
        addi $r3, $r2, 20
        sw   $r3, 0($r1)
        addi $r0, $r0, 4
        addi $r1, $r1, 4
        bnez $r2, loop
```

Este fragmento de código executará em uma máquina com a seguinte organização de memória: **Cache L1** dividido (splited), ambos com **32 KB** e política de mapeamento **2-ways set-associative**. **Cache L2** unificado, com **1 MB** e política de mapeamento **8-ways set-associative**. Nos dois casos o tamanho do **bloco** é de **32 bytes**. Assumir que o **tempo de acerto** (*cache hit*) no **cache L1** consome **4 ciclos**, o tempo de acerto no **cache L2** consome **14 ciclos** e a **penalidade** para transferir um bloco da memória principal (DRAM) para o cache L2 é de **80 ciclos**. Todos os caches implantam a política de escrita ***write-back***.

Inicialmente, os registradores armazenam os seguintes valores:

$r0: 0x00010000.

$r1: 0x00080000.

Começando na posição 0x00010000 todos os valores na memória são diferentes de zero até a posição 0x000100FC. Na posição de memória 0x000100FC está armazenado o valor zero.

a) Determine qual deve ser o Tempo Médio de Acesso a Memória (AMAT) assumindo que um determinado programa (diferente do mostrado acima) realiza em média 2 acessos de dados por instrução e apresenta a seguinte taxa de falha (***miss rate***).

L1 instructions: 10% - L1 data: 5% - L2: 2%

L1 = 4 ciclos, L2 = 14 ciclos, Penalidade DRAM->L2 = 80 ciclos.

AMAT_L1 instructions = 4 + (0.1 * (14 + (0.02 * 80))) = 4 + (0.1 * 15.6) = 5.56 ciclos

AMAT_L1 data = 4 + (0.05 * (14 + (0.02 * 80))) = 4 + (0.05 * 15.6) = 4.78 ciclos

"*assumindo que um determinado programa... realiza em média 2 acessos de dados por instrução*" :

AMAT = (AMAT_L1 instructions + 2 * AMAT_L1 data) = 5.04

b) Determine o número de faltas de cache (**miss cache**), nos módulos *data L1 cache*, *instruction L1 cache* e *L2 cache*, produzidos durante a execução do fragmento de código apresentado acima.

- Como esclarecido no enunciado, o bloco tem tamanho de 32 bytes, permitindo então o armazenamento de 8 instruções (uma palavra tem 4 bytes) por linha do cache. Como apresentado na letra A), o código foi realizado com apenas sete instruções, resultando então em apenas um compulsory miss em cada cache (1 em L1 e 1 em L2) em **L1 - instruction cache**.

Com o armazenamento de 8 instruções por bloco e 64 entradas, o resultado será 8 compulsory misses e 56 hits (em **L1 - data cache**).

P20) *Códigos de Detecção e Correção de Erros*:

a) Aplique a ***codificação Hamming*** para armazenar a seguinte palavra de 8 bits "**10011010**". Utilize a tabela abaixo para apresentar a codificação resultante.

	12	11	10	9	8	7	6	5	4	3	2	1
Posição	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bits	D8	D7	D6	D5		D4	D3	D2		D1		
Check Bits					C8				C4		C2	C1
Código	1	0	0	1	0	1	0	1	1	0	1	1

b) A ***codificação Hamming*** foi utilizada para armazenar o seguinte código de 12 bits "**010101100011**". Verifique se o código está correto, considerando que ele foi criado usando a codificação de paridade par de *Hamming*. Se o código estiver incorreto, apresente o código correto e qual é a palavra de dados original.

	12	11	10	9	8	7	6	5	4	3	2	1
Posição	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bits	D8	D7	D6	D5		D4	D3	D2		D1		
Check Bits					C8				C4		C2	C1
Código	0	1	0	1	0	1	1	0	0	0	1	1

O código está correto.