

# BCC266 - Trabalho Prático 2: Hierarquia de Memória

**Disciplina:** Organização de Computadores - UFOP

**Alunos:** Arthur Norberto da Silveira, Rafael Prado Mol e Silva, Rosalino Kenedy Serra

## 1. Introdução e Objetivos

Este trabalho prático consiste na implementação de um simulador de hierarquia de memória cache com três níveis (L1, L2, L3) interligados a uma memória principal (RAM) e uma CPU.

O objetivo principal é analisar o comportamento e o desempenho de diferentes configurações de cache (tamanhos variados) sob diferentes cargas de trabalho (probabilidades de repetição de instruções), observando o impacto das políticas de gerenciamento de memória.

### Principais Características Implementadas:

- **Arquitetura:** CPU ↔ MMU ↔ L1 ↔ L2 ↔ L3 ↔ RAM.
- **Unidade de Transferência:** Blocos de 4 palavras (inteiros).
- **Mapeamento:** Associativo (simulado através de busca sequencial na cache).
- **Política de Substituição: LRU** (Least Recently Used) - substitui o bloco menos recentemente acessado.
- **Política de Escrita: Write-Back** (Escrita Retardada) - a RAM só é atualizada quando um bloco "sujo" (modificado) é expulso da cache.

## 2. Como Compilar e Rodar

O projeto utiliza um `Makefile` para automatizar a compilação.

### Compilação Básica

Para compilar todo o projeto (Simulador + Gerador de Instruções):

```
make
```

### Execução Manual

Gere o arquivo de instruções (certifique-se de configurar a probabilidade desejada no código do gerador antes):

```
./gerador_instrucoes
```

Execute o simulador:

```
make run
```

### Execução Automatizada (Bateria de Testes)

Para gerar a tabela completa de resultados (M1 a M5) automaticamente:

```
./rodar_testes.sh
```

### Comandos Úteis do Makefile

- `make` : Compila o projeto.
- `make run` : Compila (se necessário) e roda o simulador.
- `make clean` : Remove arquivos objeto (.o) e executáveis para uma compilação limpa.

## 3. Resultados Experimentais

Os testes foram realizados utilizando 5 configurações de máquinas (M1 a M5) e 3 cenários de repetição de instruções (50%, 75% e 90%), conforme especificado no enunciado.

### Cenário 1: 50% de Repetição

**Característica:** Baixa localidade temporal. Carga de trabalho mais dispersa.

Máquina	L1	L2	L3	Taxa C1 %	Taxa C2 %	Taxa C3 %	Taxa RAM %	Tempo (u.h.)
M1	8	16	32	22,84%	60,31%	2,62%	14,23%	2.643,253
M2	32	64	128	86,27%	0,64%	0,82%	12,27%	2.175,604
M3	16	64	256	83,50%	2,45%	3,31%	10,74%	1.953,790
M4	8	32	128	22,84%	63,44%	1,36%	12,36%	2.301,613
M5	16	32	64	83,50%	1,19%	1,22%	14,09%	2.501,980

### Cenário 2: 75% de Repetição

**Característica:** Média localidade temporal.

Máquina	L1	L2	L3	Taxa C1 %	Taxa C2 %	Taxa C3 %	Taxa RAM %	Tempo (u.h.)
M1	8	16	32	80,90%	10,82%	0,39%	7,89%	987,819
M2	32	64	128	92,21%	0,25%	0,48%	7,06%	876,426
M3	16	64	256	91,97%	0,35%	1,32%	6,36%	801,726
M4	8	32	128	80,90%	11,26%	0,70%	7,14%	901,059
M5	16	32	64	91,97%	0,16%	0,29%	7,58%	937,896

### Cenário 3: 90% de Repetição

**Característica:** Alta localidade temporal. O processador acessa repetidamente os mesmos endereços.

Máquina	L1	L2	L3	Taxa C1 %	Taxa C2 %	Taxa C3 %	Taxa RAM %	Tempo (u.h.)
M1	8	16	32	93,09%	4,17%	0,12%	2,62%	333,783
M2	32	64	128	97,39%	0,07%	0,12%	2,41%	304,353
M3	16	64	256	97,30%	0,17%	0,20%	2,33%	296,352
M4	8	32	128	93,09%	4,29%	0,20%	2,43%	311,823
M5	16	32	64	97,30%	0,07%	0,09%	2,53%	318,942

**Nota:** O tempo de execução é calculado em unidades hipotéticas onde L1=1, L2=10, L3=100, RAM=1000.

## 4. Análise e Conclusão

Com base nos dados obtidos, podemos tirar as seguintes conclusões sobre o sistema implementado:

### Impacto da Localidade Temporal

- Fica evidente que o aumento da probabilidade de repetição (de 50% para 90%) melhora drasticamente o desempenho da Cache L1.
- Na máquina M1 (a menor configuração), a taxa de acerto da L1 saltou de 22,8% (cenário 50%) para 93,0% (cenário 90%).
- Isso demonstra que caches, mesmo pequenos, são extremamente eficientes para programas que exibem forte localidade temporal (loops frequentes).

### Tamanho da Cache vs. Carga de Trabalho

- **No Cenário de 50%:** a diferença entre ter uma cache L1 pequena (M1/M4 com 8 linhas) e uma média (M2/M5 com 16/32 linhas) é brutal. As máquinas M1 e M4 sofreram com muitos misses, sobrecarregando a L2 e a RAM.
- **No Cenário de 90%:** essa diferença diminui. Mesmo a M1 conseguiu segurar a maioria das instruções na L1. Isso indica que aumentar o tamanho da cache traz retornos decrescentes quando o programa é muito repetitivo.

### Eficiência do LRU

A política LRU mostrou-se eficaz em manter os blocos mais úteis próximos à CPU. Nos cenários de alta repetição, os dados raramente precisavam ser buscados na RAM (apenas ~2,5% dos acessos), provando que o algoritmo identificou corretamente os dados "quentes".

### Observação sobre o Gerador de Instruções

Notou-se que o tempo total e o número absoluto de acessos diminuem conforme a probabilidade aumenta. Isso ocorre devido à lógica do gerador: instruções repetidas dentro de um loop (N\_FOR) consumem iterações do gerador mais rapidamente, resultando em menos instruções únicas e totais geradas para o mesmo tamanho de entrada.

### Conclusão Final

Em suma, a implementação da hierarquia de memória com MMU e políticas Write-Back/LRU foi bem-sucedida, validando os conceitos teóricos de que a hierarquia de memória é essencial para reduzir o gargalo de desempenho entre a CPU e a Memória Principal.