UNIVERSIDADE DO VALE DO ITAJAÍ
ESCOLA POLITÉCNICA
CIÊNCIAS DA COMPUTAÇÃO

ENZO CIPRIANI TOMASONI
IAN MARCOS

Relatório – Memória Principal e Memória Virtual

ITAJAÍ
2025

1. Introdução

O objetivo do projeto foi desenvolver um programa para lidar com paginação, memória e utilização de algoritmos de substituição. O programa deveria ser capaz de receber uma entrada decimal ou hexadecimal referente a um endereço de memória, interpretar se funciona em 16 ou 32 bits, e formar a posição da memória desejada. Após isso deve ocorrer a busca por esse endereço seguindo sua página e deslocamento, visualizando estruturas como TLB, Page Table, Memória e Disco.

Um importante aspecto desse projeto foi observar e controlar os acertos e erros de página e TLB, além disso foi necessário implementar algum algoritmo responsável por realizar as trocas de endereços da TLB, nesse caso foi utilizado “segunda chance” ou “clock”, que prioriza que entradas utilizadas permaneçam e as demais sejam substituídas.

2. Explicação e Desenvolvimento da Aplicação

Para o desenvolvimento do projeto foi decidido utilizar como linguagem o C++, por apresentar maior conhecimento da dupla realizadora, O código conta com quatro arquivos fundamentais para a aplicação, sendo eles um arquivo de código e três referentes a dados utilizados. Primeiramente existe a main onde o código foi desenvolvido, além disso, temos a “data_memory.txt” que simula a memória do computador e a “backing_store” que simula o armazenamento em disco. Por fim existe “addresses.txt” onde existe uma série de endereços em diferentes formatos, para facilitar as simulações.

Dentro do código existem algumas informações importantes declaradas em seu início, como o tamanho de cada elemento dessa simulação. Existem algumas partes fundamentais para o entendimento do código, como a organização dos bits de um endereço em 16 ou 32 bits, e sua tradução:

Endereços Virtuais

- **16 bits:** Paginação simples, com 12 bits para deslocamento (4 KB por página) e 4 bits para número da página.
- **32 bits:** Paginação hierárquica de 2 níveis, com:
    - 10 bits para o primeiro nível
    - 10 bits para o segundo nível
    - 12 bits de deslocamento

Tradução de Endereço

A tradução segue este fluxo:

1. O programa recebe um endereço virtual (decimal ou hexadecimal).
2. O endereço é dividido entre número da(s) página(s) e deslocamento.
3. O TLB (Translation Lookaside Buffer) é consultado para mapear a página virtual ao quadro físico.

4. Se houver um TLB hit, o endereço físico é imediatamente obtido.
5. Se houver um TLB miss:
    - A tabela de páginas é consultada.
    - Se a entrada estiver inválida (bit de validade = 0), ocorre um **page fault**.
    - A página é carregada da backing store para a memória principal.
    - O TLB é atualizado com a nova entrada.
6. O valor na posição correspondente da memória física é lido e exibido.

3. Algoritmo de Substituição - Second-Chance

A TLB simula uma memória cache de endereços de páginas recentemente acessadas. Quando a TLB está cheia e uma nova entrada precisa ser inserida, usamos o algoritmo **Second-Chance** para escolher qual entrada substituir, esse algoritmo funciona de forma cíclica percorrendo todas entras, observando um bit que indica o uso das entradas. Quando o bit indica que não houve uso ocorre a substituição, caso o algoritmo passe por um bit indicando uso, ele o ignora e muda seu bit para indicar que não houve mais uso.

- Cada entrada possui um **bit de uso (bitUso)**.
- Durante a busca por uma entrada a ser substituída:
    - Se bitUso == false, a entrada é substituída.
    - Se bitUso == true, ele é resetado e a entrada recebe uma "segunda chance"; a busca continua circularmente até encontrar uma entrada com bitUso == false.

Esse algoritmo permite um funcionamento em LRU, onde as entradas usadas recentemente tendem a ser mantidas, enquanto as demais são a principal vítima das trocas.

4. Resultados Obtidos com as Simulações

Durante as simulações esses foram algumas respostas obtidas, dentre elas foi possível ver Page hits quando o dado estava localizado dentro da memory_data, TLB hit quando a página foi usada recentemente e estava ativa na TLB, também foi possível observar o comportamento de entradas em 32 bits e carregamento da backing_store. Por fim, foi possível encontrar valores satisfatórios referentes ao número de páginas, níveis, deslocamento e valores lidos.

Endereço Virtual	Tipo	Página(s)	Deslocamento	Ação	Valor Lido
0x0000	16 bits	0	0	Page hit TLB miss	110
1	16 bits	0	1	TLB hit	166
65535	16 bits	15	4095	Page hit TLB miss	44
0x00010020	32 bits	Nivel 1: 0 Nivel 2: 16	32	Page hit TLB miss Carregado da backing store	203

Após esse teste inicial, foram realizadas simulações mais extensas para observar a estabilidade do programa, onde não foram encontrados erros referentes as saídas e entradas, também foi possível acompanhar melhor a execução do algoritmo de segunda chance, principalmente quando foi alterada o tamanho da TLB para apenas 4 entradas.

5. Códigos Importantes da Implementação

a) Conversão de Endereço:

```
uint32_t converterEndereco(const string &entrada) {
    uint32_t endereco;
    if (entrada.find("0x") == 0 || entrada.find("0X") == 0) {
        stringstream ss;
        ss << hex << entrada;
        ss >> endereco;
    } else {
        endereco = static_cast<uint32_t>(stoul(entrada));
    }
    return endereco;
}
```

Esse trecho foi responsável por realizar a separação de entradas decimais e hexadecimais, através de seu prefixo “0x”.

b) Algoritmo de Segunda Chance:

```
    void inserir(uint32_t paginaVirtual, int quadro) {
        for (auto &entrada : entradas) {
            if (entrada.paginaVirtual == paginaVirtual) {
                entrada.quadro = quadro;
                entrada.bitUso = true;
                return;
            }
        }

        if (entradas.size() < TAMANHO_TLB) {
            entradas.push_back({paginaVirtual, quadro, true});
            return;
        }

        while (true) {
            if (!entradas[ponteiro].bitUso) {
                entradas[ponteiro] = {paginaVirtual, quadro, true};
                ponteiro = (ponteiro + 1) % TAMANHO_TLB;
                break;
            } else {
                entradas[ponteiro].bitUso = false;
                ponteiro = (ponteiro + 1) % TAMANHO_TLB;
            }
        }
    }
};
```

Essa é a função utilizada para realizar inserções e trocas na TLB, utilizando o algoritmo de segunda chance, que busca entradas menos recentes usando seu bit de uso.

c) Tradução com 32 bits (Hierárquica):

```
uint32_t nivel1 = (endereco >> 22) & 0x3FF;
uint32_t nivel2 = (endereco >> 12) & 0x3FF;
uint32_t deslocamento = endereco & 0xFFF;
uint32_t paginaVirtual = (nivel1 << 10) | nivel2;
```

Aqui ocorre a formação de endereços com hierarquia de dois níveis em 32 bits, onde os 10 primeiros bits e os próximos 10, compõem os níveis, que serão utilizados juntos do deslocamento para encontrar um endereço.

6. Análise e Discussão dos Resultados Finais

Os testes demonstraram que:

- A separação entre paginação simples e hierárquica foi eficaz para representar as arquiteturas de 16 e 32 bits.
- A política de LRU na TLB desenvolvida usando o algoritmo de segunda chance foi capaz de selecionar com boa precisão as entradas utilizadas a mais tempo para realizar a sua troca, quando necessário.
- Foi possível ter um bom controle das entradas de dados utilizando um arquivo em .txt. Além de ser fácil de compreender as saídas recebidas de forma individual.

Limitações:

- O bit de controle "Dirty" não teve foco na implementação, ficando sem uma função considerável.
- A interface utilizada usando apenas o terminal foi limitada, dificultando a visualização de algumas métricas referentes ao processo como um todo.

7. Conclusão do projeto

O projeto atinge seu objetivo ao fornecer uma simulação realista da tradução de endereços virtuais com diversos mecanismos utilizados em sistemas operacionais reais, proporcionando compreensão teórica e prática sobre gerenciamento de memória. O desenvolvimento auxiliou na compreensão de ambos os membros do grupo sobre os conceitos e prática envolvendo memoria, paginação e algoritmos de troca.

O projeto está disponível em repositório online no github através do seguinte link: https://github.com/EnzoCTomasoni/T2-SO

8. Referências:

Felipe Viel. Slide Memória Virtual – disciplina de Sistemas Operacionais, Universidade do Vale do Itajaí (Univali), Itajaí, SC, 2025.

GEeksforGeeks. Paging in Operating System. Disponível em: https://www.geeksforgeeks.org/operating-systems-paging-in-operating-system/. Acesso em: 25 mai. 2025.

OSDev. Paging - OSDev Wiki. Disponível em: https://wiki.osdev.org/Paging. Acesso em: 25 mai. 2025.