Universidade do Vale do Itajaí

Curso: Bacharelado em Ciência da Computação

Professor: Felipe Viel

Disciplina: Sistemas Operacionais

Avaliação – Escalonamento e Memória

Acadêmicos: Alexandre Machado de Azevedo,

Matheus Passold Carelli,

Vinícius dos Santos Moreira.

Itajaí, Outubro de 2023

Projeto

Suponha que um sistema tenha um endereço virtual de tamanho entre 16 bits à 32 bits com deslocamento na página de 256 b à 4 Kb. Escreva um programa que receba um endereço virtual (em decimal) na linha de comando ou leitura do arquivo addresses.txt faça com que ele produza o número da página e o deslocamento do endereço fornecido, sendo que essa posição indica qual a posição que será lido do arquivo data_memory.txt. Você irá encontrar esses arquivos no github da disciplina, mais especificamente na pasta Memory (link repositório).

Por exemplo, seu programa seria executado da seguinte forma:
./virtual_memory_translate.exe 19986
ou
./virtual_memory_translate.exe addresses.txt

Seu programa produzirá:

- O endereço 19986 contém:
  - o número da página = 4
  - o deslocamento = 3602
  - o Valor lido: 50 (exemplo)

No caso, o número em binário é 0100 1110 0001 0010, sendo que 0100 diz respeito à página e 1110 0001 0010 diz respeito ao deslocamento na página. Para manipular os números em nível de bit, é recomendado usar os operadores bitwise (bit-a-bit) da linguagem escolhida.

No caso o exemplo apresentado é para 16 bits. No caso de 32 bits, haveriam mais 16 bits a esquerda (mais significativo) referentes ao número de páginas.
Escrever este programa exigirá o uso do tipo de dados apropriado para armazenar 16 à 32 bits (short ou int). É recomendado que você também use tipos de dados sem sinal. Além disso, para endereços de 32 bits deve ser possível usar paginação hierárquica de 2 níveis mantendo 4 Kb, com cada nível tendo 10 bits de tamanho.

Resolução:

O código foi implementado em JavaScript, sendo criado diversas funções com o intuito de ser possível que seja retornado o número de páginas, o deslocamento e o valor correspondentes ao endereço de memória passado na execução do programa. Foram criados para transformar números decimais em binários e ser possível identificar o local de memória correspondente ao número decimal informado. Foi criado também uma função para a leitura de arquivos txt com valores de endereço de memória, além de funções para transformação e leitura dos binários a partir do número decimal para ser possível obter o valor da página, do deslocamento e do valor.

Resultados:

Neste projeto podemos observar os diversos valores resultantes dos cálculos dos endereços para realizar a sua paginação e observar os valores das páginas, do deslocamento e do valor correspondente ao endereço lógico passado. O código do projeto contém duas funções de paginação, a paginação normal e a paginação hierárquica. A paginação normal executa tanto valores de 16 à 32 bits, já a hierárquica executa somente endereços de 32 bits, já que trata-se de uma paginação que disponibiliza de dois níveis que seguem o padrão requisitado no enunciado do projeto.

Para demonstrar os resultados de cada paginação, abaixo estão 3 figuras que mostram a execução para cada tipo de entrada e paginação. Os parâmetros passados na Figura 1 e na Figura 2 exemplificam a dinamicidade do código, permitindo executar sem passar um parâmetro de deslocamento/tamanho de página, o qual irá utilizar o valor padrão de 12 bits/4096 ou passando um valor que pode ir de 8 a 12 bits(256/512/1024/2048/4096) conforme requisitado pelo enunciado do trabalho.

Na Figura 1 está sendo referenciado todos os possíveis casos para a execução da paginação dos endereços, no caso utilizando um endereço de 16 bits que é o número 19986. Ao executar o arquivo, foi realizado com sucesso o cálculo do endereço desejado, sendo retornado o endereço lógico informado(19986), o número de páginas(4) e o deslocamento(3602) calculados e o valor(65) correspondente.

```
xandeturf@Speed:~/sistemas_operacionais/trabalhos/m2/javascript$ node script.js endereco=19986
[
  {
    "enderecoSolicitado": "19986",
    "numero_paginas": 4,
    "numero_deslocamento": 3602,
    "numero": "65"
  }
]
xandeturf@Speed:~/sistemas_operacionais/trabalhos/m2/javascript$ node script.js endereco=19986 deslocamento=256
[
  {
    "enderecoSolicitado": "19986",
    "numero_paginas": 78,
    "numero_deslocamento": 18,
    "numero": "65"
  }
]
xandeturf@Speed:~/sistemas_operacionais/trabalhos/m2/javascript$ node script.js endereco=19986 deslocamento=512
[
  {
    "enderecoSolicitado": "19986",
    "numero_paginas": 39,
    "numero_deslocamento": 18,
    "numero": "65"
  }
]
xandeturf@Speed:~/sistemas_operacionais/trabalhos/m2/javascript$ node script.js endereco=19986 deslocamento=1024
[
  {
    "enderecoSolicitado": "19986",
    "numero_paginas": 19,
    "numero_deslocamento": 530,
    "numero": "65"
  }
]
xandeturf@Speed:~/sistemas_operacionais/trabalhos/m2/javascript$ node script.js endereco=19986 deslocamento=2048
[
  {
    "enderecoSolicitado": "19986",
    "numero_paginas": 9,
    "numero_deslocamento": 1554,
    "numero": "65"
  }
]
xandeturf@Speed:~/sistemas_operacionais/trabalhos/m2/javascript$ node script.js endereco=19986 deslocamento=4096
[
  {
    "enderecoSolicitado": "19986",
    "numero_paginas": 4,
    "numero_deslocamento": 3602,
    "numero": "65"
  }
]
```

Figura 1: resultados 16bits.

Para utilizar endereços de 32 bits o cálculo para obter o endereço correspondente pode ser feito de duas maneiras. A primeira maneira basta somente passar o endereço lógico do mesmo jeito que foi inserido para calcular o endereço de 16 bits, já a segunda maneira é com a paginação hierárquica deve ser passado um parâmetro “hierarquia=true” para indicar que o script deve executar a segunda função de localização.

Na Figura 2 podemos observar um escopo do parâmetro praticamente idêntico ao primeiro caso e com isso podemos observar que se trata da mesma função de paginação, mas agora com 32 bits. Nela é fornecido o endereço lógico 429986 para ser calculado e com isso o resultado podemos observar na Figura 2 abaixo, sendo bem maior a quantidade de páginas em relação às da Figura 1, e o motivo é bem simples.

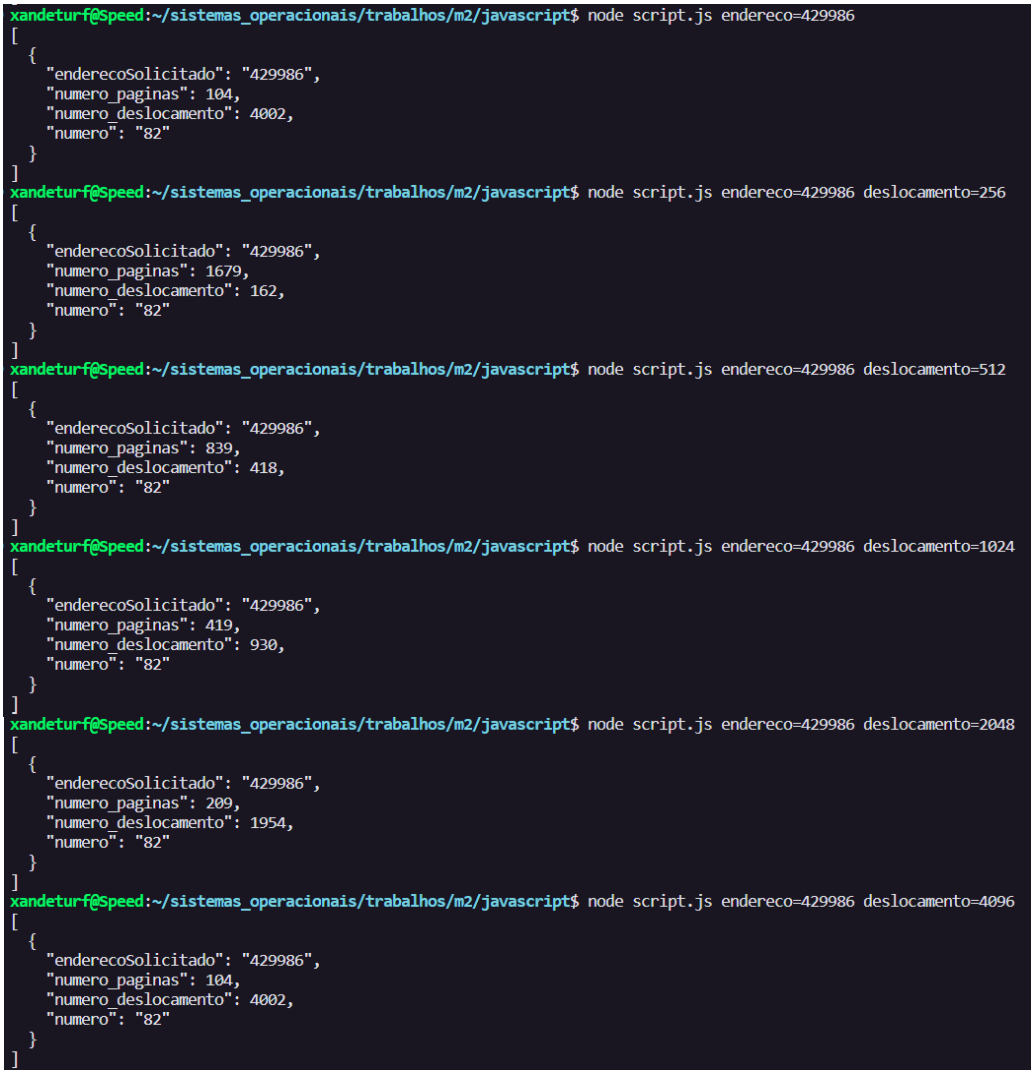

Figura 2: resultados 32bits.

O último caso, e não menos importante, corresponde à paginação hierárquica. Na Figura 3 podemos observar que a diferença já começa na chamada do script.js, o qual recebe um endereço de 32 bits e um parâmetro de ativação da paginação hierárquica “hierarquia=true”. Com isso, durante a execução vai ser direcionado para ser utilizado a função de paginação correspondente e o cálculo do endereço é executado. Neste tipo de paginação, os 32 bits são divididos em 10 bits para cada nível, 1 e 2, e 12 bits para o deslocamento na página, o qual reflete também o tamanho da página, assim cumprindo os parâmetros que foram requisitados no enunciado do projeto para a paginação hierárquica.

Na Figura 3 é possível observar a quantidade de páginas para cada nível e o deslocamento interno, além da confirmação de ativação da paginação, o endereço passado e o valor correspondente.

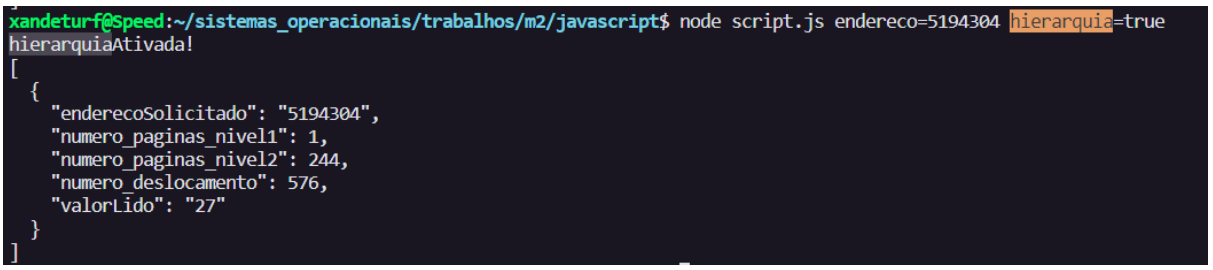

Figura 3: resultado 32bits com hierarquia.

Códigos:

Essa função(Figura 4) se trata da conversão do endereço que foi passado por parâmetro na execução do código que está na forma decimal, para sua representação em binário. Sendo assim possível de realizar os cálculos necessários para obter as páginas, o deslocamento e o valor necessário.

```
function decimalToBinary(decimal) {
    let binary = "";
    while (decimal > 0) {
        let remainder = decimal % 2;
        binary = remainder + binary;
        decimal = Math.floor(decimal / 2);
    }
    return binary;
}
```

Figura 4: função que converte decimal para binário.

A função localizarEndereco(Figura 5) é para encontrar um endereço específico dentro de um espaço de endereçamento de memória(no nosso caso o arquivo data_memory.txt), com base em diferentes tamanhos de página. Ela recebe como entrada o enderecoSolicitado, que é o endereço que se deseja localizar, o tamanhoDaPagina, que define o tamanho de cada página de memória, e a variável eh32bits, que indica se o espaço de endereçamento é de 32 bits ou 16 bits.

Começa convertendo o enderecoSolicitado de decimal para binário e preenchendo-o com zeros à esquerda, dependendo da configuração de 32 bits ou 16 bits. Em seguida, ela itera por números decimais, construindo binários para representar o número de páginas e o deslocamento dentro de uma página de memória. Ela compara o endereço binário completo construído com o enderecoSolicitado. Se houver uma correspondência, a função retorna um objeto com informações sobre o endereço encontrado, caso contrário, ela continua a iterar.

```
function localizarEndereco(enderecoSolicitado, tamanhoDaPagina, eh32bits) {
    let enderecoBinario = decimalToBinary(enderecoSolicitado).padStart(eh32bits ? 32 : 16, 0);

    let numero_paginas = 0;
    let numero_deslocamento = 0;
    let tratamentoBinarioDeslocamento = Math.log2(tamanhoDaPagina);

    for (const numero of numeros_decimais) {
        var binario = '';
        if (eh32bits) {
            binario = decimalToBinary(numero_paginas).padStart(32 - tratamentoBinarioDeslocamento, 0);
        } else {
            binario = decimalToBinary(numero_paginas).padStart(16 - tratamentoBinarioDeslocamento, 0);
        }
        binario += decimalToBinary(numero_deslocamento).padStart(tratamentoBinarioDeslocamento, 0);

        if (enderecoBinario == binario) {
            return {
                enderecoSolicitado,
                numero_paginas,
                numero_deslocamento,
                numero
            };
        }
        numero_deslocamento++;
        if (numero_deslocamento >= tamanhoDaPagina) {
            numero_paginas++;
            numero_deslocamento = 0;
        }
    }
    return false;
}
```

Figura 5: função que localiza o endereço.

A função localizarEnderecoHierarquia(Figura 6) tem como objetivo encontrar um endereço específico dentro de uma hierarquia de páginas de memória. Ela recebe como entrada o enderecoSolicitado, que é o endereço que se deseja localizar, o tamanhoDaPagina, que define o tamanho de cada página de memória (mantido como 4KB) conforme solicitado.

Ela começa convertendo o enderecoSolicitado de decimal para binário e preenchendo-o com zeros à esquerda para ter um comprimento de 32 bits. Em seguida, ela itera por números decimais, construindo binários para representar o nível 1, o nível 2 da hierarquia e o deslocamento dentro de uma página de memória. Ela compara o endereço binário completo construído com o enderecoSolicitado. Se houver uma correspondência, a função retorna um objeto com informações sobre o endereço encontrado, caso contrário, ela continua a iterar.

```
function localizarEnderecoHierarquia(enderecoSolicitado, tamanhoDaPagina, eh32bits) {
    console.log('hierarquiaAtivada!');

    tamanhoDaPagina = 4096; // mantem o tamanho de 4Kb conforme solicitado

    let enderecoBinario = decimalToBinary(enderecoSolicitado).padStart(32, 0);
    let limitMaximoHierarquia = 1024;
    let numero_paginas_nivel1 = 0; // limit maximo de 1024 = 10 bits
    let numero_paginas_nivel2 = 0; // limit maximo de 1024 = 10 bits
    let numero_deslocamento = 0;

    let bitsParaDeslocamento = Math.log2(tamanhoDaPagina); // 12
    let bitsParaNiveis = 10; // conforme solicitado

    for (const numero of numeros_decimais) {
        let binarioNivel1 = decimalToBinary(numero_paginas_nivel1).padStart(bitsParaNiveis, 0);
        let binarioNivel2 = decimalToBinary(numero_paginas_nivel2).padStart(bitsParaNiveis, 0);
        let binarioDeslocamento = decimalToBinary(numero_deslocamento).padStart(bitsParaDeslocamento, 0);

        let binarioCompleto = binarioNivel1 + binarioNivel2 + binarioDeslocamento;

        if (enderecoBinario == binarioCompleto) {
            return {
                enderecoSolicitado,
                numero_paginas_nivel1,
                numero_paginas_nivel2,
                numero_deslocamento,
                valorLido: numero
            };
        }

        numero_deslocamento++;

        if (numero_deslocamento >= tamanhoDaPagina) { // 4096
            numero_deslocamento = 0;
            numero_paginas_nivel2++;
            if (numero_paginas_nivel2 >= limitMaximoHierarquia) { // 1024
                numero_paginas_nivel2 = 0;
                numero_paginas_nivel1++;
            }
        }
    }
    return false;
}
```

Figura 6: função localizarEnderecoHierarquia.

Resultados finais:

Pode se observar que o código escrito teve resultados esperados para a resolução do problema. Foi possível visualizar a página, o deslocamento e o valor do endereço corretos, utilizando diversos parâmetros que deveriam ser possíveis de serem alterados. Dentro os parâmetros possíveis, o tamanho da página poderia ser de 256b a 4Kb e em todas as execuções os valores foram calculados de maneira correta. Além disso, o valor do endereço de 16 bits a 32 bits também era possível de ser modificado e ainda sim o código foi executado corretamente. As conversões de decimal para binário e vice-versa funcionaram corretamente e permitiram a precisão necessária para a pesquisa. O objetivo principal do projeto foi alcançado.