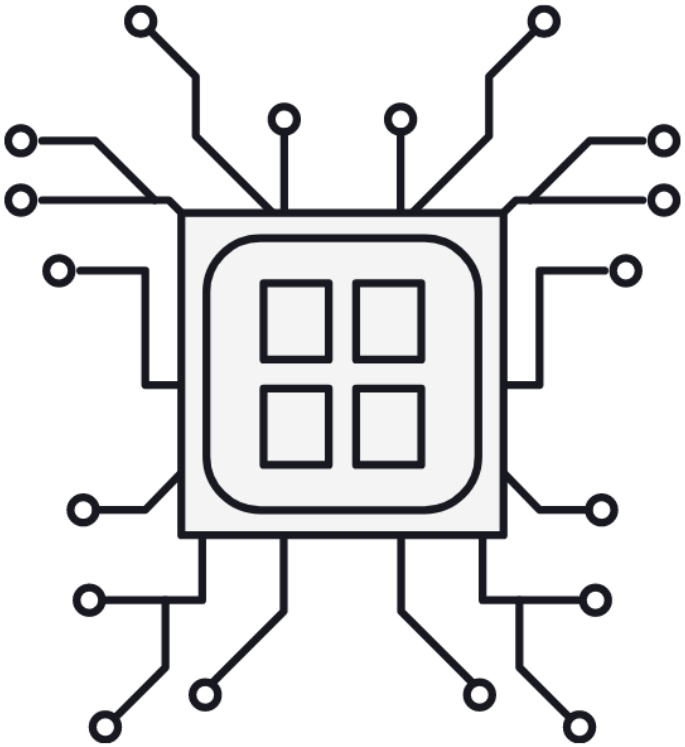
Portfólio

Guilherme Aguiar Silva Milanez

October 14, 2020



Sumário

Contents

1 Motivação

O seguinte trabalho tem como objetivo principal fornecer uma análise acerca dos temas introdutórios da disciplina de Sistemas Operacionais, tai como Processos, Threads, gerenciamento de memória dentre outros. Ao obter tal análise o estudante estará mais familiarizado com conceitos relevantes e que poderão ser, futuramente, facilmente relembrados ao ler esse documento.

2 Introdução

2.1 Histórico

Mostra-se fundamental entender a história e, por conseguinte, a evolução dos sistemas. Pois tais abstrações permite ao entusiasta perceber quais fatores foram definitivos para o acontecimento de grandes marcos evolutivos na área em estudo. Define-se a história dos sistemas em 4 fases, sendo a quarta fase a que estamos inseridos hoje. Cada uma delas será abordada com cuidado nos tópicos a seguir:

Primeira Fase (1945 - 1955)

Na primeira fase da evolução dos computadores, os sistemas operacionais, não eram utilitários existentes, tendo em vista, que as máquinas da época não possuíam concepções de alto nível, como linguagens de programação, unidade central de processamento, múltiplos processos concorrentes dentre outras composições que exigem a integração de um sistema operacional coordenador. Essas máquinas eram baseadas em válvulas, geralmente ocupavam salas inteiras, não possuíam níveis superiores de abstração e o seu principal objetivo era a realização de cálculos matemáticos.

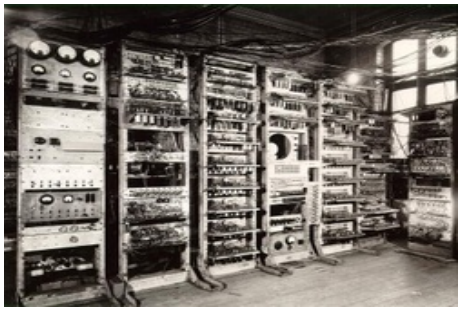


Figura 1: Manchester Mark I (1948)

Segunda Fase (1955 - 1965)

A segunda fase foi marcada, majoritariamente, pela passagem do uso de computadores baseados em válvulas, para computadores baseados em transistores. A eficiência dessa tecnologia, era explícita, não necessitavam de pré-aquecimento para serem utilizados, apresentavam uma melhor eficiência energética, a emissão de calor era reduzida em comparação com as válvulas, além de serem mais rápidos e possuir uma confiabilidade melhor. Foi durante essa fase que surgiram alguns dos conceitos mais importantes para os computadores modernos, como a Unidade Central de Processamento (CPU), unidades de memória, linguagens de programação e métodos de entrada e saída. Cada vez menos, a linguagem de máquina foi sendo usada, dando lugar a linguagem Assembly, definida como uma programação baseada em mnemônicos, que em seguida deu lugar a linguagens de alto nível como Fortran, LISP, ALGOL e COBOL, apresentando um marco para a computação. Com tais mudanças o tamanho das máquinas sofreu uma redução considerável. Além dessas novidades a década de 50 acompanhou o surgimento dos Sistemas Operacionais alguns deles são:
1 - MIT: Tape Director para o UNIVAC 1955 ;
2 - General motors – SO para o IBM 701 `[U+F06C]`1956;
3 - General motors – SO para IBM 704;

Terceira Fase (1965 - 1980)

A terceira geração é denotada pela eclosão dos circuitos integrados, comumente chamados de Microchips esses circuitos eram construídos com Silício. Sua constituição garantia uma grande quantidade de transistores com dimensão bastante reduzida, tal fator contribuiu para o preço dessa tecnologia ser diminuto. Muito se compara o surgimento dos CI com o surgimento da imprensa pela forma de fabricação, esses dispositivos eram fabricados em

grande quantidade e simultaneamente, o que garantiu tamanha explosão no seu uso. Um grande marco para época, foi iniciado pela IBM ao discernir a produção de hardware da produção dos sistemas que futuramente seriam adequados ao maquinário, começando assim a fervorosa industria de software. Os computadores produzidos na década de 60 e 70 foram demarcados pelo uso do Timesharing, tal técnica era incorporada em grandes máquinas que suportavam muitos usuários, em que cada um desses tinha acesso à uma máquina virtual, cujo acesso aos recursos do sistema era igualitário, ou seja os recursos computacionais eram compartilhados. Nesse contexto surge também o MULTICS, que foi utilizado de base para o Unix (1965), que por sua vez, foi um sistema que cooperava com a necessidade de tempo e recurso compartilhado para múltiplos usuários ativos utilizando o sistema.

Quarta fase (1980 - Atualmente)

Com o uso em massa dos circuitos integrados, surge nesse contexto a era dos computadores pessoais ou microcomputadores. Novas arquiteturas de 64bits foram incorporadas aos computadores, que por sua vez passaram a suportar sistemas mais robustos e complexos de alto nível. Alguns dos sistemas que surgiram na época possuem versões atualizadas até os dias de hoje como o MAC OS de 1984 e o Windows em 1984. Um marco divisor nesses sistemas foi a adequação de interfaces gráficas (GUI), algo que é de suma importância para o uso cotidiano de computadores na atualidade. Em meados dos anos 80 houve também o surgimento de redes para computadores pessoais, essas por sua vez executavam sistemas operacionais de redes e sistemas distribuídos, que são utilizados até hoje, com suas devidas atualizações mas com o mesmo propósito.

2.2 Definição

Um sistema operacional pode ser definido, como uma interface complexa que engloba aspectos de baixo nível, como Drivers de dispositivos e recursos computacionais, com aspectos de alto nível como softwares para os usuários. Um sistema operacional pode ser pensado como uma ponte, coordenadora, entre os recursos que o hardware fornece e as exigências que o software impõe.

Figura 2: Esquema Software/SO/Hardware

Um sistema operacional pode ser definido, a partir de seus objetos primordiais, em dois aspectos: Uma Máquina Estendida e um Gerenciador de Recursos.

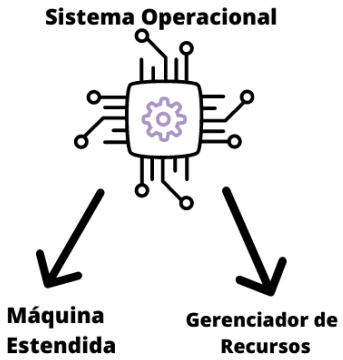


Figure 1: Figura 3: Sistema Operacional abstrações

O sistema operacional na perspectiva de uma máquina estendida, mostra-se como uma ferramente de grande utilidade. Majoritariamente, as arquiteturas dos computadores em nível de linguagem de máquina são complexas e bastante árduas para serem programadas. Até mesmo tarefas "simples" como armazenar, remodelar ou excluir alguma informação de arquivo, nesse nível de abstração, tornam-se mais enigmáticas . O sistema operacional, por sua vez cria uma camada de intermédio de alto nível dedicada ao usuário permitindo que o mesmo não perca tempo para compreender processos e movimentações de informação a nível de máquina a fim de programar ou , até mesmo, desfrutar de alguma informação. A nomenclatura desse modo de visão de um sistema operacional, faz referência com as próprias máquinas virtuais, pois o que é entregue, ao final, para o usuário é uma máquina virtual, mais simples que o sistema em nível de arquitetura, mas mantendo os objetivos; A imagem abaixo ilustra de forma clara a facilidade proporcionada pelo Sistema Operacional ao agir como uma máquina estendida, tando para o usuário comum, quanto para os programadores.

Figura 4: Sistema Operacional como máquina estendida

Por outro lado, o sistema operacional, apresentado como um gerenciador de recursos, tem como cerne manter a ordem dos recursos que o hardware dispõe. Computadores modernos, dispõe de recursos, que eram inimagináveis há alguns anos atrás, como múltiplos cores em CPUs, cada vez mais quantidade memória de acesso randômico, dentre outros aparatos que tornam essas máquinas muito poderosas. Contudo, os softwares no geral, como as aplicações, os utilitários e os sistemas operacionais, também acompanharam essa evolução, cada vez mais esses projetos estão apresentando uma enorme exigência, simultânea, de recursos computacionais havendo, frequentemente, a indispensabilidade de um gestor. Nesse contexto um gerenciador de recursos garante que todas as necessidades, sejam elas de recursos ou de informações, sejam efetuadas, com eficácia, para que todos os processos em atividade possam ser executados com eficiência. Essa distribuição pode ser realizada em função de dois parâmetros:
1 - Tempo: O sistema operacional definirá quando um aplicação acessará qual recurso, ou quando uma aplicação obterá tal, informação, coordenando assim múltiplas solicitações de acordo com a demanda temporal;
2 - Espaço: O sistema operacional dividirá os recursos disponíveis em relação, em competência de ocupação desse recurso, de acordo com a demanda das aplicações, a aplicação que necessita de mais recurso para ser executada, receberá mais recurso.

2.3 Tipos de Sistemas Operacionais

2.3.1 Sistemas Operacionais para Desktop

São sistemas adaptados para a performance em computadores pessoais, desktops, são sistemas diversos, tais como Windows, MacOS, Linux etc. Muito comumente, são utilizados por um único usuário, seu uso é baseado em tarefas do cotidiano, tais como, acesso a internet, carregamento de mídia, editores de texto etc.

2.3.2 Sistemas Operacionais de Grande Porte

Esses sistemas são aptos a suportar um variação singular de entrada e saída composta por milhares de bytes, geralmente esse computadores ocupam salas inteiras e seu uso principal é o processamento de dados em grande escala, sendo utilizados também para hospedarem sites, por possuir a robustez necessária para tal tarefa, essa robustez manifesta-se no seu processamento que é baseado na execução simultânea de múltiplos Jobs, os serviços que os sistemas operacionais dessas maquinas ofertam, são definidos em :
1 - Serviço em Lote;
2 - Processamento de Transação;
3 - Tempo Compartilhado;

2.3.3 Sistemas Operacionais de Servidor

Esses computadores são um pouco mais simples que os de grande porte, por algumas vezes, integrá-los como computadores pessoais. Os sistemas operacionais dessa máquina, é capaz de suportar múltiplos usuários, suas funções mais comuns são: Prover serviços de compartilhamento de dados, impressão,prover acesso a web dentre outros. Sendo que muitas vezes a própria web utiliza desses aparatos, também, para hospedagem de páginas.

2.3.4 Sistemas Operacionais de Tempo Real

Esse tipo sistema possui como parâmetro fundamental, o tempo. São empregados em diversas áreas desde processos fabris, como na produção de automóveis à uso domestico, como sistemas de controle de lavagem de roupas. Podem ser definidos em sistemas críticos, sistemas em que a falha no tempo não é tolerável, e sistemas não críticos em que alguns atrasos são aceitáveis.

2.3.5 Sistemas Operacionais Embarcados

Sistemas operacionais desse tipo são mais complexos, por estarem presentes em dispositivos com diversas restrições computacionais, como baixa capacidade de processamento, pouca disponibilidade de memória, mesmo assim, sua performance deve ser algo a ser bem entregue. Mesmo com restrições, esses sistemas devem coordenar entrada e saída, processamento de um ou mais núcleos, interface para periféricos etc.

2.3.6 Sistemas Operacionais de Cartões Inteligentes

Trata-se de sistemas aptos para operarem os menores dispositivos, que contém apenas um CHIP como CPU. Como os sistemas embarcados, os sistemas de cartões possuem uma severa restrição de consumo de energia e memória; Alguns cartões possuem suporte para pequenas aplicações feitas em Java.

2.4 Revisão de Arquitetura de Computadores

Os computadores, atualmente, são compostos por 4 elementos primordiais, são eles:

1 - Processadores;
2 - Memória;
3 - Dispositivos de Entrada/Saída;
4 - Barramentos;
Cada um desses elementos será explicado com mais detalhes a seguir;

2.4.1 Processadores

O processador é a parte do computador responsável por realizar todas as operações lógicas, cálculos matemáticos e execução de instruções. Comumente esses dispositivos são denominados microchips por integrar uma CPU inteira em um único chip, algo inimaginável alguns anos atrás. As principais funções da Unidade Central de Processamento são:
1 - Busca e execução de instruções;
2 - ordena todos os outros chips existentes na máquina;
A composição da CPU, é enxuta, contendo três elementos básicos; Unidade de Controle, Unidade Lógica e Aritmética e Registradores;
1 - Unidade de Controle: Parte fundamental para o funcionamento do computador; pois possui o controle de todas ações a serem executadas pelo computador, essa unidade garante a execução correta dos programas bem como seus respectivos acessos e manipulações de dados;
2 - Unidade Lógica e Aritmética: Também denominada ALU, é responsável por todas as tarefas relacionadas a processamentos lógicos e aritméticos, necessitados na execução de um programa qualquer;
3 - Registradores: Esses dispositivos permitem que o processador possua uma memória de velocidade considerável, mas com capacidade reduzida, com o objetivo de armazenar, temporariamente, valores intermediários ou informações de comandos; Alguns dos registradores mais importantes são: O contador de programa (PC), que aponta para a próxima instrução a ser executada; O Registrador de instrução (IR) tem como objetivo guardar a instrução que está em execução.

2.4.2 Memória

Todo computador possui em sua composição uma quantidade limitada de memória, portanto o uso e ordenação inteligente desse elemento garantirá a sua eficiência. A memória de um computador é divida nas seguintes categorias que compõem a hierarquia de memória:

1 - Memória Cache: Trata-se de uma memória bastante limitada cujo propósito é o armazenamento rápido de informações dentro do processador para que o mesmo possa realizar as tarefas esperadas.

2 - Memória Principal: É a memória em que são armazenados os programas e informações a serem processadas pela CPU;

3 - Memória Secundária: Nessas memória os dados não são voláteis e podem ser armazenados em grande escala, diferentemente, das memórias primarias e de cache;

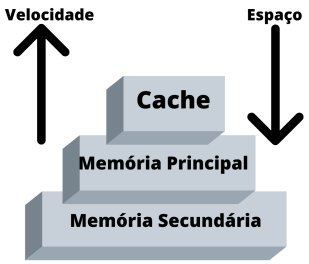


Figure 2: Figura 5: Hierarquia de Memória

Já os tipos de memória, em relação a chips, podem ser divididos em duas categorias, memórias voláteis e não voláteis:
1 - Memória volátil: Comumente denominada RAM, memória de acesso randômico, A CPU utiliza a RAM para guardar e executar os programas vindos do disco, para realizar a leitura e o armazenamento de dados que estão sendo processados;
2 - Memória Não volátil: São memórias cujas informações armazenadas não são perdidas quando o computador é desligado. Os tipos mais comuns são ROM, PROM, EEPROM e EPROM;

2.4.3 Dispositivos de Entrada/Saída

Os sistemas Operacionais também são responsáveis pelo controle dos dispositivos de entrada e saída. Esses dispositivos geralmente são constituídos de duas partes básicas: Um controlador e o próprio dispositivo; O controlador é um conjunto de hardware que recebendo comandos do sistema operacional irá coordenar o dispositivo de entrada e saída, esse por sua vez estabelece uma comunicação restrita com o Driver do dispositivo, programa responsável pela execução de comandos;

2.4.4 Barramentos

Barramentos ou Bus, são implementações físicas, ordenadas, responsáveis pelo tráfego de dados dentro do computador, sendo estabelecido por dois ou mais componentes. Os principais tipos de barramentos são:

1 - Barramento do Processador: Trata-se do barramento que o chipset usa a fim de enviar/receber informações da CPU;

2 - Barramento de Cache: usado nos processadores Pentium Pro e Pentium III, é um barramento dedicado exclusivamente para o acesso ao sistema cache;

3 - Barramento de Memória: Trata-se de um barramento, cuja principal função é conectar o sub-sistema de memória ao chipset e, também, ao processador. 4 - Barramento Local de E/S (Entrada/Saída): Esse barramento é utilizado para realizar a conexão entre periféricos de alto desempenho e memória, chipset e, também, o processador.

5 - Barramento Padrão de E/S: Esse barramento concatena os três barramentos sobre o antigo barramento padrão de E/S, usado para periféricos lentos.

2.5 Conceitos Introdutórios de Sistemas Operacionais

O objetivo dessa unidade é abordar resumidamente os principais conceitos acerca dos sistemas operacionais. Esses conceitos serão melhor aprofundados no decorrer desse documento;

2.5.1 Processos

Um processo ou tarefa, pode ser entendido, de maneira bastante simplificada, como um programa em estado de execução, competindo por recursos de processamento, e interagindo com outros processos; Cada um desses, possui consigo: Um espaço de endereçamento que nada mais é que uma lista de posições na memória, em que o processo pode operar; Um conjunto de registradores, contendo um contador de programas, um ponteiro para a pilha e demais registradores; Além disso os processos possuem todas as informações suficientes para a correta execução de um programa. Os processos constituem-se em uma natureza determinística e sequencial por obedecer as seguintes premissas:

1 - O resultado da execução de um processo é independente da velocidade na qual o processo foi executado;
2 - Se o processo for executado outra vez, utilizando os mesmos dados, ele passará pelo mesmo conjunto ordenado de instruções e ofertará os mesmos resultados;

2.5.2 Deadlocks

Durante a execução de um processo, vários recursos devem ser alocados exclusivamente, para a execução do mesmo; Em um contexto multiprogramado, no qual, vários processos competem por um mesmo recurso, esses processos podem se encontrar em um bloqueio imensurável, pois sempre estarão esperando por um recurso que nunca lhes será proporcionado; Essa situação é conhecida como Deadlock. Uma exemplificação comum dessa situação é a de alocação e, consequente, liberação de recursos;

2.5.3 Gerenciamento de Memória

Em sistemas operacionais mais simples a memória só suporta uma execução de programa por vez, fazendo com que um programa tenha que esperar o termino de outro a fim de poder utilizar desse recurso; Enquanto que em sistemas operacionais mais sofisticados, permitem que múltiplos programas residam na memória principal simultaneamente, havendo a necessidade de uma organização eficaz por parte do sistema. Outra preocupação do sistema operacional em relação à memória é em relação a manipulação do espaço de endereçamento, aspecto inerente aos processos; Em diversas situações o hardware não consegue disponibilizar todo o aparato de memória que um processo necessita. Quando isso ocorre, os sistemas implementam, as chamadas Memórias virtuais, com o intuito de expandir a capacidade desse recurso de maneira abstrata mas conseguindo manter o mesmo nível de entrega desses recursos reais.

2.5.4 Dispositivos de Entrada/Saída

Majoritariamente, os computadores necessitam de informações provenientes dos dispositivos de entrada e saída para realizar algum tipo de operação, cabe ao sistema operacional coordenar, por meio de diversos Drivers de dispositivos e controladores, a correta execução desses dispositivos de entrada e saída.

2.5.5 Arquivos

A parte mais perceptível de um sistema operacional, para o usuário, é um sistema de arquivo. As aplicações utilizam desse aparato, para ler, escrever ou remodelar os mais diversos tipos de arquivos suportados pelo sistema; Com o intuito de manter a vasta quantidade de arquivos organizada dentro da memória, o sistema operacional utiliza de uma ferramente chamada diretório, cuja principal função é o agrupamento organizado de arquivos no sistema. A hierarquia de processos e diretórios é baseada na estrutura de árvore; Todo processo de manipulação, seja criação, remoção ou edição de diretórios e de arquivos é realizado executando-se chamadas de sistemas. Alguns tipos de arquivos especiais são :

1 - Bloco: Utilizados para realizar o acesso à blocos de discos diretamente;

2 - Caracteres: Utilizados para mapear dispositivos que recebem fluxo de caracteres serialmente;

3 - Pipe: Utilizados para manter uma comunicação entre dois processos, semelhante ao processo de leitura e escrita de arquivos;

2.6 Chamadas ao Sistemas

As chamadas ao sistemas, são o aparato que o sistema operacional utiliza para realizar a interface de serviços disponíveis para os processos. Geralmente essas tecnologias são escritas em linguagem de alto nível como C e C++, mas são consideradas rotinas de baixo nível por serem executadas em modo núcleo; Os serviços ofertados por essas chamadas delimitam, majoritariamente, as ações que os sistemas operacionais devem se responsabilizar, alguns desses serviços são: criar e finalizar algum processo, manipular arquivos assim como, gerir entrada e saída e coordenar diretórios; As chamadas ao Sistemas podem ser divididas nas seguintes categorias:

1 - Chamadas ao Sistema Para Gerenciamento de Processos : Sua principal responsabilidade é o gerenciamento de processos; Aluns exemplos são a chamada Fork() que cria um processo filho idêntico ao processo pai e Waitpid() que aguarda um processo filho terminar;

2 - Chamadas ao Sistema para Gerenciamento de arquivos: Usadas para manipular arquivos, algumas dessas chamadas são: Open(), usada para abrir um arquivo para leitura e/ou escrita; Issek() usada para a movimentação de ponteiro na posição do arquivo e Stat() usada para obter a atual situação do arquivo;

3 - Chamadas ao Sistema de Diretório e Arquivo: Essas, por sua vez são usadas quando o Sistema deseja ter algum poder sobre os diretórios e os arquivos; Alguns exemplos são: mkdir() que realiza a criação de um novo diretório, rmdir() que remove um diretório selecionado; Mount() que realiza a montagem de um sistema de arquivos, dentre outras chamadas;
Uma forma de proporcionar ao usuário os serviços ofertados pelo sistema operacional, de forma textual, é por meio do interpretador de comandos. Esse método, usa os comandos para especificar qual chamada de sistema adequada ao pedido do usuário, realizando tal operação de forma implícita.

2.7 Estrutura do Sistema Operacional

2.7.1 Sistemas Monolíticos

Trata-se do tipo de arquitetura de SO mais popular, contudo em sua composição não há uma ordem rígida e organizada, ocasionando em uma miscelânea de procedimentos, frequentemente mal vista. Nessa organização cada procedimento possui parâmetros e cômputos rígidos e bem definidos, a interação entre os demais procedimentos é sem restrição, conforme for sua necessidade. Uma possível organização de estrutura básica de um sistema operacional monolítico é a seguinte:

1 - Um programa Principal cuja responsabilidade é invocar o procedimento do serviço pretendido;

2 - Um conjunto de procedimentos de serviços que tem por função executar as chamadas ao sistema;

3 - Um conjunto de procedimentos Utilitários que ajudam os procedimentos de serviço;

2.7.2 Sistemas de Camadas

Trata-se de uma arquitetura mais sofisticada propondo a organização do sistema em camadas, onde a camada mais baixa realiza a interface com o Hardware e a mais elevada propõem as abstrações adequadas para aplicações de alto nível. Os problemas ocasionados por essa disposição são:

1 - O empilhamento dos softwares em camadas faz com que ocorra atrasos de requisições de alguns softwares por não possuírem o nível de privilégio adequado.

2 - Os serviços do sistema propostos em divisão de camada não é trivial. Tendo em vista que muitas operações são interdependentes e a organização em camadas correlacionadas com níveis de hierarquia seria inviável.

2.7.3 Máquinas Virtuais

Máquinas Virtuais são modelos de software que utilizam da técnica de virtualização para entregar um sistema operacional que independe do hardware, diversas vezes executado sobre outro sistema, atuando como mediador de diversas instâncias de sistemas operacionais executados simultaneamente em um mesmo hardware. A virtualização, por sua vez, trata-se do compartilhamento dos recursos computacionais para vários sistemas, por meio de uma abstração computacional. O núcleo desses sistemas é conhecido como Monitor de máquina Virtual, é responsável por implementar a multiprogramação permitindo múltiplas máquinas virtuais em um mesmo hardware, tendo em análise que tais máquinas não apresentam-se como máquinas estendidas e sim como cópias do hardware fornecendo o mesmo aparato tecnológico oferecido por essa camada.

2.7.4 Exonúcleos

Também conhecidos como sistemas operacionais verticalmente estruturados. Trata-se de um modelo de SO, em que os exonucleos, estruturas pequenas, possuam apenas as responsabilidades de multiplexação e proteção, de recursos, especificamente, memória. Sua principal finalidade é permitir que uma aplicação solicite uma determinada região da memória e que seja satisfeita tal necessidade sem muitas burocracias.

2.7.5 Modelo Cliente-Servidor

A ideia principal dessa estrutura, é retirar o máximo possível de código do nível do núcleo, transferindo-os para as camadas mais superiores, implementando uma grande parte do sistema em processos a nível de usuário. Cria-se duas categorias de processos. Processos clientes realizam requisições quem são atendidas por processos servidores, deixando para o micronucleo apenas o processo do estabelecimento de comunicação entre essas duas categorias de processo em atividade. Por haver uma grande divisão das funcionalidades as operações tornam-se mais eficientes em termos de gerenciabilidade. Além disso, esse modelo permite que os sistemas tenham uma maior adaptabilidade quando incorporados em sistemas distribuídos.

3 Processos e Threads

3.1 Processos

3.1.1 Modelo de Processos

Todo processo, por padrão, possui um registrador denominado Program Counter (PC), esse registrador é majoritariamente utilizado para armazenar informações da próxima instrução a ser executada. Mesmo em execução dentro de CPUs de núcleo único o fato da execução dos processos tratar-se de forma sequencial, transmite a ilusão de estar ocorrendo um paralelismo, essa transição sequencial, denomina-se troca de contextos;

3.1.2 Criação e Termino de processos de Processos

Alguns eventos são responsáveis pela necessidade da criação de um processo. Existem 4 tipos de eventos, principalmente, que demandam a criação de um novo processo, são eles:

1 - Inicialização do Sistema;

2 - Ao se executar uma chamada ao sistema com o proposito de criação de um novo processo decorrente do processo em execução;

3 - Quando há alguma requisição do usuário a fim de criar um novo processo;

4 - Início de algum job em lote, que, nada mais é que um processamento de dados multiprogramados em grande quantidade;

Como todo ciclo sem encerra, após ser criado e realizar,ou não, seu objetivo, um processo deve ser encerrado; Existem algumas situações que desencadeiam o termino de um processo, são elas:

1 - Evasão normal, ocorre de forma voluntária;

2 - Evasão devido a algum erro, ocorre de forma voluntária;

3 - Erro irremediável, ocorre de forma involuntária;

4 - Cancelamento decorrente de outro processo, ocorre de forma involuntária;

3.1.3 Hierarquia de Processos

Processos estão organizados no sistema de forma hierárquica, basicamente pertencendo a duas categorias, Processos pais e processos filhos; Processos pais são responsáveis pela criação de processos filhos, e esses por sua vez, podem criar seus próprios processos, tornando-se processos pais. Nos sistemas de base Unix, a hierarquia é denominada grupo de processos; Já nos sistemas Windows, por exemplo, todos os processos são criados de forma igualitária;

3.1.4 Estados de Processos

Para uma melhor gerência, em panorama geral, de todos os processos no sistema, há uma organização por estados; Existem três estados, nos quais um processo pode se encontrar, são eles:

1 - Estado de execução, no qual o processo está de fato processando informações e executando alguma tarefa;

2 - Pronto, O estado fica em uma espécie de Stand-By, parado.

3 - Bloqueado, O estado encontra-se inerte, incapaz de executar, até que algum elemento externo modifique esse estado;

A figura abaixo exibe os três estados e as possíveis relações de transição entre eles:

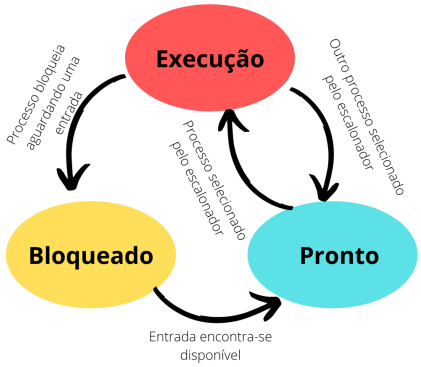


Figura 6: Estados de Processos

3.1.5 Implementação de Processos

A fim de implementar os processos o sistemas mantém uma estrutura de dados denominada, tabela de processo, na qual, cada entrada refere-se a um processo, contendo informações exclusivas sobre o mesmo como : Estado do processo, Informação no contador de Programa, Ponteiro da pilha, informação sobre alocação de memória e os estados dos arquivos relacionados a esse processo. Admitindo-se o modelo de multiprogramação, é pertinente demonstrar os métodos de análise de quando muitos processos competem pelos recursos: O uso da Cpu é dado por 1 - pˆN, onde n é o número de processos e p é a fração de tempo de atraso devido a entrada e saída, relatando o uso em percentual;

3.2 Threads

Threads podem ser definidas como um fluxo de controle, ou linha de execução independente; Processos podem ter diversas threads, caso seja essa situação ele compartilhas seus recursos entre suas threads. Cada thread é identificada pelo código que está operando e por um contexto local thread Local Storage (THS), que compõe o conjunto de registradores e outras informações que são exclusivos de cada thread.

3.2.1 Características

1 - Podem ser consideradas como processos leves, executando de forma concorrente durante a execução do processo proprietário dessas threads;

2 - Threads compartilham alguns recursos como, espaço de endereçamento, arquivos abertos no momento dentre outros;

3 - Cada thread possui parâmetros de informação exclusiva, como ; Contador de Programa, Ponteiro de pilhas e outros registradores;

3.2.2 Motivação

Alguns modelos de aplicação, permitem a ocorrência de múltiplas atividades ao mesmo tempo. Por consequência há o bloqueio das mesmas, periodicamente, o uso de threads sequenciais na decomposição desses modelos torna-os mais simples; Outra motivação para seu uso, é que essas linhas de execução são mais fáceis de criar e destruir que os processos de maneira geral, por serem independentes de associações de recursos; Threads também facilitam o processamento de entrada e saída quando há um grande volume de dados, dando-lhes preferencia e por fim Threads são úteis em sistemas com múltiplos processadores atuando em paralelismo.

3.2.3 Threads de Usuário

São threads que podem ser implementadas em sistemas que não suportam threads, pois esse tipo de thread é executado dentro do processo do usuário, sem haver interferência do sistema, por consequência, é necessário o uso de um sistema supervisor para monitorar os fluxos, cada uma dessas threads pode ter seu próprio algoritmo de escalonamento, tendo em vista sua independência do sistema. Contudo, uma desvantagem desse modelo de implementação de thread, e cada uma, por si só deve ceder recursos de CPU para as demais, sendo necessário realizar uma chamada ao sistema para bloqueio.

3.2.4 Threads de Núcleo

Nesse tipo de thread, o controle é realizado pelo próprio sistema operacional, são utilizadas, majoritariamente, para compor fluxos de execução dentro do SO, portanto não são mais necessários supervisores individuais, tendo em vista que o SO se encarregará disso. As tabelas de threads também estarão no núcleo, bem como as devidas informações, tudo isso, por conseguinte, tende a ser implementado por uso de chamadas ao sistemas, utilizando programação de baixo nível.

3.2.5 Threads Híbridas

Modelo em que threads de núcleo são usadas e então threads de usuários são multiplexadas sobre algumas ou todas as threads de núcleo; Tendo, o núcleo, controle apenas sobre threads de núcleo e essas por sua vez são responsáveis pelas threads de usuário.

3.3 Comunicação Interprocessos

Processos devem manter comunicação uns com os outros. Por meio disso, podem enviar e receber informações relevantes, em que muitas vezes essas são necessárias para o funcionamento de um processo.

3.3.1 Condição de Disputa

Situação em que dois ou mais processos estão realizando leitura de ou escrita de algum dado compartilhado e cujo resultado final é dependente das informações provenientes de quem e quando está executando de forma precisa.

3.3.2 Regiões Críticas

A fim de evitar problemas, quando se é usado métodos de compartilhamento, emprega-se a técnica de exclusão mutua, ou seja, uma forma de assegurar que outros processos sejam impedidos de usar algum recurso compartilhado que já estiver em uso por outro processo. Mesmo assim alguns processos necessitam acessar parte da memória compartilhada, essas regiões que podem ocasionar disputas são denominadas Regiões críticas. Para que dois ou mais processos não caiam em disputas é necessário ir além de delimitar as regiões críticas, tendo em vista que diversas vezes os processos precisam acessar tais regiões. Algumas das ações que podem ser tomadas para evitar condição de disputa, são;

1 - Nunca ocorrer de dois processos estarem simultaneamente em suas regiões críticas;

2 - Nunca deve se dar a precisão da velocidade ou sobre disponibilidade da CPU;

3 - Processos sendo executados fora de suas regiões críticas jamais poderão bloquear outros processos;

4 - A espera para entrar em uma região crítica, por parte de um processo não pode ser eterna;

3.3.3 Exclusão Mútua com Espera Ociosa

Desabilitando Interrupções

Nessa solução a interrupção por parte do relógio não ocorre; Cada processo desabilita todas as interrupções ao adentrar na sua seção crítica e retorna a habilitá-las no instante em que evadi essa região.

Variáveis do Tipo Trava

Nessa solução há uma variável de compartilhamento global que é utilizada como chave sempre que um processo precisa entrar na região crítica, verificando o valor da mesma. Caso essa variável contenha o valor 0 o processo entra na região crítica e altera o valor para 1, pode ocorrer de um ou mais processos entrarem na região antes da permuta de valores da variável.

Alternância Obrigatória

Nesse caso a troca de contexto é ordenada por uma variável denominada Turno. Essa variável é usada em um Loop para realizar tempo de espera, diferentemente do tipo trava que só verificava o valor da chave e toda vez que o processo termina de executar em sua região crítica a variável turno é setada de forma que o outro processo saia do

loop de espera e comece a executar na região crítica.Porém em certos casos essa solução acarreta em um problema sério, que infrige a 3 regras das ações a serem tomadas para evitar condição de corrida, pois um processo fora da sua região crítica bloqueia outro processo. Existem outras soluções como a de Peterson e a TSL que também não são muito eficientes ocasionando inversão de prioridade .

3.3.4 Dormir e Acordar

Tendo em vista problemas como a inversão de prioridade, é implementado, por conseguinte, duas primitivas de comunicação, que realizam bloqueio ao invés de gastar a CPU, quando não podem adentrar a região crítica. Sleep e WakeUp. Sleep indica para o sistema que o mesmo deve ficar suspenso, a chamada WakeUp acorda o processo que é passado como parâmetro.

O PROBLEMA PRODUTOR-CONSUMIDOR

Trata-se de um problema, no qual dois processos compartilham um buffer comum e de tamanho fixo. O produtor insere informações enquanto o consumidor retira essas informações, o problema ocorre quando o produtor deseja inserir um novo item mas o buffer encontra-se cheio, a solução seria colocar o produtor em modo de suspensão para que o consumidor remova um ou mais itens. A imagem abaixo apresenta essa solução:

```
#define N 100
#define TRUE 1
int contador = 0 ;
void produtor(void){
    int item;
    while(TRUE){
        item = produz_item();
        if(contador == N) sleep();
        isere_item(item);
        contador++;
        if(contador == 1) WakeUp(consumidor);
    }
}
void consumidor(void){
    int item;
    while(TRUE){
        if(contador == 0) sleep();
        item = remove_item();
        contador--;
        if(contador == N - 1) WakeUp(produtor);
        consome_item(item);
    }
}
```

Figura 7: Consumidor e produtor

3.3.5 Semáforo

Técnica implementada por Dijkstra, sugeria a utilização de uma variável inteira, denominada semáforo, para tal existem duas operações atômicas DOWN e UP. A operação DOWN checa se o valor é maior que 0, se for o valor será decrementado e continuara, se for 0 o processo será posto para dormir e a operação UP incrementa esse valor. Por serem atômicas, essas operações serão únicas, isso garante que uma vez que o semáforo iniciou sua operação nenhum outro processo poderá acessar o semáforo até que a operação completada ou bloqueada. Os semáforos são usados de duas maneiras:
1 - Mutex: Que é utilizado para garantir exclusão mutua; 2 - Full e Empty : São utilizados para fins de sincronização, definir que certos eventos sequenciais ocorram ou não ocorram; A imagem abaixo exibe a implementação:

```
#define MAX 100
#define TRUE 1
typedef int semaforo;
typedef int Item ;
semaforo mutex, empty, full;
mutex = 1 ;
empty = MAX;
full = 0;
void produtor(void) {
    Item item ;
    while(TRUE) {
        item = produz();
        Down(&empty);
        Down(&mutex);
        insere(item);
        Up(&mutex);
        Up(&full);
    }
}
void consumidor(void){
    Item item ;
    while(TRUE){
        Down(&full);
        Down(&mutex);
        item = remove();
        Up(&mutex);
        Up(&empty);
        item = consome();
    }
}
```

Figura 8: Consumidor e produtor com semáforo

Caso as operações Down do produtor fossem invertidas, dow(mutex) ante de dow(empty) e se o buffer estivesse completamente cheio o produtor seria bloqueado e após dar um down(mutex) no consumidor também estaria bloqueado desencadeando em uma condição de corrida;

3.3.6 Monitores

É uma construção da linguagem de programação, tratando-se de uma coleção de procedimentos e outras estruturas agrupadas em um único módulo, em que apenas um único processo pode estar ativo em dado instante, Sendo os compiladores, os responsáveis por garantir a exclusão mutua.

3.3.7 Troca de Mensagens

Outra técnica de comunicação interprocessos é a troca de mensagens, utiliza de duas operações, Send() e Receive(). Contudo, diferentemente, do semáforo, essas operações são chamadas ao sistema e não apenas construções da linguagem. Não há muita vantagem em sua utilização em uma única máquina, para tal é mais recomendado o uso de semáforo, porém em comunicação de mais de uma máquina, esse método apresenta-se de forma útil, sendo bastante empregada em sistemas de computação paralela.

3.3.8 Barreiras

Essa formulação é empregada sobre grupos de processos, usado quando esses processos trabalham em grupos e são executados por fase.A barreira, por sua vez, é utilizada para delimitar que a próxima fase comece somente quando todos os processos, que estavam em execução na fase atual, estiverem prontos; Assemelha-se a um escolar em dias de excursão, no qual o mesmo só retorna da viagem para a escola quando todos os alunos, que foram para o passeio, encontrarem-se no ônibus.

Figura 9: Metáfora barreira/ônibus
fonte: https://www.gratispng.com/png-yco4da/download.html

3.4 Escalonamento

Trata-se da parte do sistema operacional, responsável por gerir o acesso a CPU em ambiente multiprogramados, enfrentando escolhas de qual processo, encontrando-se no estado pronto, terá acesso a determinado recurso de processamento. Sua implementação precisa ser inteligente e eficaz, tendo em vista, que a troca de contexto é uma ação custosa. O tipo do processo, se é orientado a CPU ou orientado a Entrada e Saída, influenciará a tomada de decisão do escalonador. Existem dois tipos básicos de algoritmos de escalonamento:

1 - Preemptivos: Há um limite de tempo pré-fixado para os processos usarem os recursos do processador, caso não consiga executar todas suas tarefas no tempo determinado máximo, o processo em questão será suspenso;

2 - Não Preemptivos : Não há limite de tempo de uso da CPU, por parte de um processo, ele só para de usufruir do recurso quando o faz voluntariamente ou quando é bloqueado; Diferentes ambientes, requerem diferentes algoritmos de escalonamento, esses ambientes e juntamente sua categoria de escalonador, principais, são:

3.4.1 Em Lote

Como não há espera de usuários, podem ser não preemptivos ou preemptivos com um intervalo de tempo de execução considerável. Tendo como objetivos: Maximizar o número de tarefas por hora, reduzir o tempo entre submissão e termino e mantenimento da CPU em trabalho; Os algoritmos principais para esse tipo de sistema são:

1.1 PRIMEIRO A CHEGAR PRIMEIRO A SER SERVIDO: Os processo são organizados em uma fila de prontos;

1.2 JOB MAIS CURTO PRIMEIRO; Não preemptivo, é necessário saber o tempo restante de execução para se escolher o melhor;

1.3 PRÓXIMO DE MENOR TEMPO RESTANTE; Trata-se da implementação preemptiva do algoritmo de JOB MAIS CURTO PRIMEIRO;

1.4 ESCALONAMENTO EM TRÊS NÍVEIS: Compostos por três escalonadores: De admissão, de memória e de CPU;

3.4.2 Interativo

Preempção é precisa, a fim de evitar congestionamento de processos ao usar a CPU, esse por sua vez tem como objetivos principais: Respostas eficientes à requisições interativas e a satisfação do usuário; Os algoritmos principais para esse tipo de sistema são:

2.1 ESCALONAMENTO POR CHAVEAMENTO CIRCULAR: Algoritmo preemptivo em que cada processo tem seu próprio quantum de tempo, geralmente de 20 a 50ms;

2.2 ESCALONAMENTO POR PRIORIDADES: A cada processo é atribuído um grau de prioridade e o processo de mais alta prioridade será executado. A cada tique do relógio a prioridade do que está executando é reduzida para evitar ocupação total;

2.3 ESCALONAMENTO POR FILAS MÚLTIPLAS: Os processos são remanejados de acordo com a necessidade de tempo, podendo cair na fila e, portanto, tendo mais tempo de acesso a CPU;

2.4 PRÓXIMO PROCESSO MAIS CURTO: Trata-se da implementação preemptiva do algoritmo de JOB MAIS CURTO PRIMEIRO;

2.5 ESCALONAMENTO GARANTIDO: Tendo n usuários, para cada processo é alocado 1/n tempo de processamento;

2.6 ESCALONAMENTO POR LOTERIA: Os processos são agraciados por bilhetes e o prêmio é um bom quantum de execução, processos podem transacionar os bilhetes entre si conforme a necessidade;

2.7 ESCALONAMENTO POR FRAÇÃO JUSTA: O usuário é um peso, ou seja, escalona-se de uma forma que todos os usuários tenham acesso à CPU;

3.4.3 Tempo Real

Por os processos já serem feitos com demanda de tempo, a preempção é supérflua, os objetivos do escalonamento para esses sistemas são, basicamente, tentar evitar dados perdidos e evitar a queda de qualidade em sistemas multimídia; Dado M eventos periódicos, o evento i ocorre no período P e requer C segundos de uso, logo a carga poderá ser trata, somente se o somatório de C/P for menor ou igual a 1;

Os principais objetivos desses algoritmos baseiam-se em três partes fundamentais, Justiça, isto é, cada processo deve obter uma porção justa de processamento;Aplicação de política, verificando-se constantemente, se a política remanejada está sendo de fato efetivada e por fim Equilíbrio, Com o intuito de manter todas as partes do sistemas fora do ócio.

3.4.4 Escalonamento de Threads

Threads de usuário, por não ter controle por parte da CPU, em ambientes múltiprogramados, por não haver interrupção de relógio, a thread define seu quantum ; Threads de núcleo, cabe ao escalonador escolher qual thread executar.

3.5 Jantar dos Filósofos

Modelagem de problema para processos que competem por recursos limitados. Tem se cinco filósofos sentados ao redor de uma mesa circular. Cada filósofo tem um prato de espaguete. Precisa-se de dois garfos para comer entre cada par de pratos está um garfo. Ou o filósofo está pensando ou está dormindo; Tendo como ideia central a prevenção de Deadlocks, dependendo de como a implementação está sendo realizada, o problema pode gerar uma condição de corrida, tendo a analogia de processos com os filósofos. A imagem abaixo apresenta uma solução :

```c
#define N       5
#define ESQUERDA (i+N-1)%N
#define DIREITA  (i+1)%N
#define PENSANDO 0
#define FAMINTO  1
#define COMENDO  2
#define TRUE 1
typedef int Semaforo;
// uma estado para cada filosofo
int estado[N];
Semaforo Mutex = 1;  /*vetor de semaforo para cada estado */ Semaforo S[N];
void tenta(int i){
    if(estado[i] == FAMINTO && estado[ESQUERDA] != COMENDO && estado[DIREITA] != COMENDO){ estado[i] = COMENDO; Up(&S[i]); }
}
void P_Garfo(int i ){
    // operacao atomica do mutex , bloqueia os outro para acessar a região critica
    Down(&Mutex);
    estado[i] = FAMINTO;
    // tentativa de alocar recurso (pegar garfo)
    tenta(i);
    Up(&Mutex); Down(&S[i]);
}
void D_Garfo(int i){
     Down(&Mutex);
     estado[i] = PENSANDO;
     tenta(ESQUERDA); tenta(DIREITA);  // ve quem vai ser o proximo a comer
     Up(&Mutex);
}
void filosofo(int i ){
    while(TRUE){
        // estado de espera
        pensar();
        //aloca recurso
        P_Garfo(i);
        // usa recurso
        comer();
        // devolve recurso
        D_Garfo(i);
    }
}
```

Figura 10: Solução Jantar dos Filosofos

3.6 Problema dos Leitores e Escritores

Problema em que é necessário gerenciar quando leitores e escritores podem escrever num Buffer.Vários processos podem realizar leitura pois não há modificação no Buffer, mas apenas um processo pro vez pode realizar a escrita por proporcionar modificação na base de dados; A imagem abaixo permite a visualização de uma possível solução:

```
#define TRUE 1
typedef int Semaforo;
typedef int Leitor;
Semaforo mutex = 1 ;
Semaforo bd = 1 ;
Leitor Pleitor = 0;
void leitor(){
    while(TRUE) {
        Down(&mutex);
        // pode aumentar o numero de processos leitores
        Pleitor++;
        if(Pleitor == 1 ) Down(&bd);
        Up(&mutex);
        Ler_BD(); // acesso ao buffer
        Down(&mutex);
        Pleitor --;
        if(Pleitor==0) Up(&bd);
        Up(&mutex);
        Usar_dados_lidos();
    }
}
void escritor(void){
    while (TRUE){
     dados_para_escrever();
     Down(&bd);
     Escrever_BD(); // altera dados do buffer
     Up(&bd);

    }

}
```

Figura 11: Solução Leitor/Escritor

4 Gerenciamento de Memória

4.1 Conceitos Básicos de Gerenciamento de Memória

A memória é um recurso limitado, o ideal é que ela seja disposta, na máquina, em um modelo de hierarquia em que os mais altos níveis são mais rápidos e os mais baixos possuem maior capacidade;

4.2 Multiprogramação sem Abstração

Os programas utilizam bastante a memória, programas sem abstrações são simplesmente alocados sem nenhum precedente ou regra, causando problemas; As possíveis abstrações para esse tipo de problema dividem-se em:

4.2.1 Realocação Estática

No momento da alocação de memória para instruções haverá a modificação real dos endereços para que tudo seja reajustado; trata-se de um processo um pouco rudimentar;

4.2.2 Espaço de Endereçamento

Trata-se de um grupo de endereços com a finalidade de identificar a memória; O que esse método propõe é a criação de uma memória abstrata para abrigar os processos. Não sendo necessário a realocação estática.

4.2.3 Registrador Base e Limite

Para cada um dos processos que precisaram usar a memória, existe um par de registradores, Base e Limite, que determinam a região de memória usada pelo processo em questão, o registrador base será o endereço inicial do programa e o registrador limite é o tamanho do processo;

4.2.4 Swapping

Em sistemas de compartilhamento de tempo a memória principal pode não conseguir conciliar todos os processos. Ideia básica seria a de utilizar espaço em disco como extensão da memória RAM, e nesse, alocar os processos enquanto estiverem bloqueados, realocando-os para a memória assim que são desbloqueados. Para se realizar a troca deve-se usar os métodos de realocação estática ou registrador Base-Limite; O heap do processo pode crescer de forma inesperada, para tal situação, muda-se o processo de local ou retira outro processo, para que isso ocorra de maneira segura é necessário haver um espaço adjacente livre;

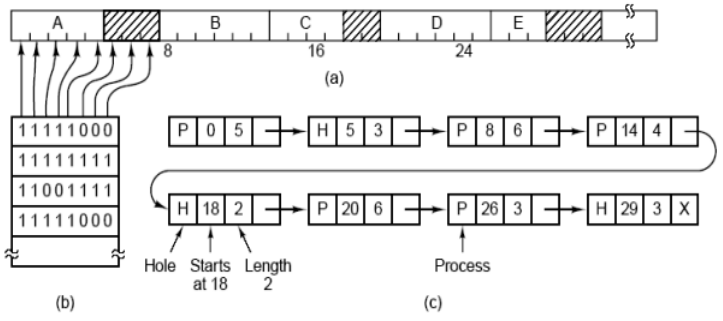
4.2.5 Gerenciamento de Memória Livre

A realização desse processo é de muita utilidade pois, é necessário saber a quantidade do recurso que poderá ser usada por outro processo; A ideia principal consiste em: seccionar a memória principal em unidades de alocação de n bytes e representar a ocupação de de lotes de unidades que poderão estar livres ou ocupados, usando um mapa de bits ou então uma lista encadeada;

1 - MAPA DE BITS: Armazenamento enxuto e simples, mas buscar por determinados tamanhos de lote livre, acarreta na análise de várias palavras; Algo de relevância nesse método é o tamanho da unidade de alocação. Essa se relaciona com o mapa da seguinte forma: Quanto menor a unidade de alocação, maior será o mapa de bits e quanto maior a unidade de alocação maior será a probabilidade de haver desperdícios;

2 - LISTA ENCADEADA: A lista é composta por várias células ligadas, cada célula contém o endereço inicial e o tamanho de uma partição estando ela ocupada ou livre; O uso de listas duplamente encadeadas, facilita o processo de mesclar espaço livre com segmentos não utilizados anteriormente;

A figura abaixo representa as duas técnicas citadas:



Representação dos espaços de memória com mapa de bits e lista ligada. (a) 5 segmentos alocados a processos e três livres. (b) mapa de bits. (c) lista ligada.

4.2.6 Gerenciamento de Memória com Listas

Existem dois possíveis casos principais: Quando processo é do tipo swapped out, isto é, retirar o processo da memória principal, a lacuna correspondente tende a ser relacionada com espaços vizinhos livres, Já no caso de o processo é swapped in, está voltando para a memória principal, percorre-se a lista buscando um espaço livre suficientemente grande;

ALOGRITMOS PARA DEFINIÇÃO DESSES SEGMENTOS:

1 - FIRST FIT:O primeiro espaço encontrado é alocado;

2 - NEXT FIT: FirstFit + Armazenamento do ultimo segmento que coube na requisição, a próxima busca começa por esse endereço;

3 - BEST FIT: Percorre toda a lista e aloca o menor segmento possível, fazendo isso pode restar fragmentos reduzidos para alocação de outros processos;

4 - WORST FIT: Corre toda a lista e aloca a maior região possível, não havendo forma de reposição de espaços que sobram;

4.3 Memória Virtual

Muitas vezes os processos carecem de mais memória do que o hardware pode oferecer para que os mesmos possam trabalhar, para isso, utiliza-se a memória virtual utilizando-se parte do disco como extensão da memória RAM. Quando esses novos segmentos são gerados, o seus identificadores são denominado endereços virtuais e cabe a CPU enviar os endereços à MMU, que é a unidade de gerenciamento de memória, para que então, possam ser configurados em endereços físicos. Em máquinas que não utilizam memória virtual, os endereços físicos são os mesmos dos endereços virtuais. O processo de conversão dos endereços é ilustrado abaixo:

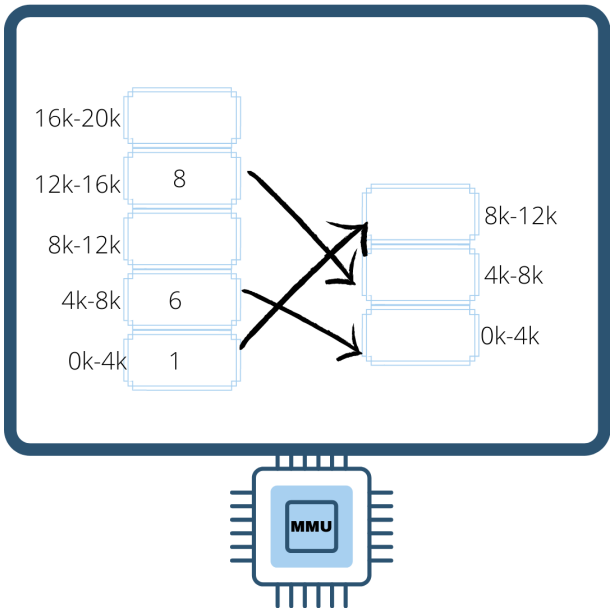


Figura 13: MMU

Os processos de formulação dos endereços podem ser entendidos da seguinte forma : Utiliza-se a técnica de paginação, na qual, os espaço de endereçamento de cada processo e a memória física são segmentados em partições de mesmo tamanho, essas partições são denominadas páginas e quadro de páginas. Tendo se um tamanho da página sendo de 4KB pode variar de 512 Bytes a 64 KB; Um espaço de endereçamento virtual de 64K é mapeado em 32KB de memória RAM; O número da página é usada como índice para a tabela de páginas, estrutura usada para armazenar as páginas; Cada entrada na tabela contém, entre outros:

- Bit presença: se página está em memória ;
- Número da moldura ;
- Referenciação;
- Verificação de Modificação ;

• Proteção para Leitura/ Escrita ;

O ENDEREÇO FÍSICO será composto por 13 bits menos significativos (direita) + 3 bits de tabela de página; Já o ENDEREÇO VIRTUAL será composto por 4 bits mais significativos para encontrar em qual parte da tabela será encontrado. Por exemplo se os 4 bits mais significativos forem: 0011 a pagina estará no campo 3 da tabela como mostra a figura abaixo retirada do livro[1]:

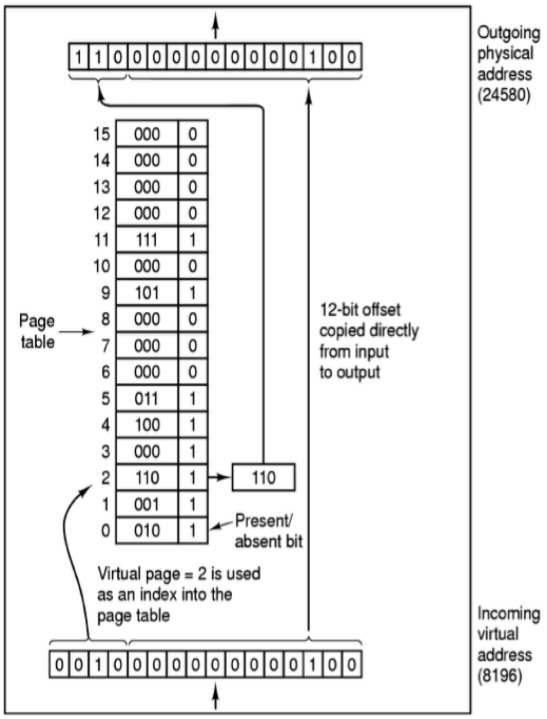


Figura 13: MMU

4.3.1 Acelerando a Paginação

A paginação precisa ser um processo eficaz e rápido, se o espaço for relativamente grande, a tabela também será! Sabe-se, que todas as tabelas de páginas são mantidas em memória, o que pode causar certos gargalos, para tanto, é usado um conjunto de registradores que possuem o intuito de armazenar as páginas usadas com mais frequência, esse conjunto recebe o nome de TLB, ou tabela de tradução de endereço.

4.3.2 Tabela de Páginas Multiníveis

Trata-se de uma solução para tabelas com memórias grandes; A ideia principal por trás dessa técnica é tornar a Tabela paginada, mantendo, apenas os segmentos cujo mapeamento esteja sendo usado. Geralmente usa-se dois níveis:
• Nível 1 : Tabela Raiz;
• Nível 2 : Tabela de Páginas;
Exemplo: Endereço de 32 bits dividido em pt1 = 10 btis, pt2 = 10bit , Deslocamento = 12bits; Cada entrada na Tabela nível 2 representa 4k(tamanho da página) ; Cada entrada na Tabela nível1 representa 1kX 4k = 4Mb;

4.3.3 Tabelas de Páginas Invertidas

Para sistemas com arquiteturas de 64 bits, o espaço de endereçamento é de 264 bytes, Independente de o tamanho de página ser de 4KB, isso leva a geração de tabelas muito grandes em nível de milhões de entradas na tabela de páginas, algo inviável. Portanto , em alguns sistemas usa-se uma Tabela invertida: Nessa existe apenas uma entrada por moldura de página na memória real, ao invés de uma entrada por página do espaço de endereçamento virtual; Um dos problemas desse método é que a tradução do endereço virtual para o real pode ser vagarosa; pesquisando na tabela inteira ao invés de ir pulando por faltas; A solução para esse problema é a TLB tornando eficiente o sistema de busca, ou utilizando um tabela Hash.

4.4 Algoritmo de Substituição de Páginas

A situação da falta de página leva à necessidade de se escolher qual página deve ser removida, para isso alguns algoritmos são empregados:

4.4.1 Algoritmo Ótimo de Substituição de Página

• Algoritmo Ótimo, mas impossível de ser implementado;
• Prever quais paginas provocarão falhas para retirá-las;

4.4.2 Algoritmo de Substituição de Página Não Usada Recentemente (NUR)

• Trabalha-se com os bits de Referenciação e Modificação;
• Há classificação das páginas em : Classe 0: Não referenciada, não modificada; Classe 1: Não referenciada, modificada; Classe 2 :Referenciada, modificada; Classe 3: Referenciada, Não modificada :
• Remove-se a página de classe mais baixa ;

4.4.3 Algoritmo de Substituição de Página Primeiro a Entrar Primeiro a Sair (FIFO)

• Mantém uma lista encadeada de todas as páginas ;
• Mais antiga na cabeça da lista e a mais nova na cauda ;
• Página na cabeça da lista é retirada, a nova é inserida na cauda ;
• Página duradouras podem ser usadas frequentemente, sendo uma desvantagem;

4.4.4 Algoritmo de Substituição de Página Segunda Chance

• Modificação FIFO;
• Verificação do bit de referenciação R ;
• Se R = 0, a página será substituída imediatamente;
• Se R = 1,bit R será permutado para 0; A página será alocada para o final da lista de páginas; Tempo de carregamento é setado como no início;
• Busca continua;

4.4.5 Algoritmo de Substituição de Página de Relógio

• Baseado no algoritmo de segunda chance, diferindo-se apenas na implementação;
• É implementada um ponteiro circular para o bit R;
• Se R = 0, a página será substituída imediatamente;
• Se R = 1,bit R será permutado para 0; A página será alocada para o final da lista de páginas; Move o ponteiro do relógio ;
• Busca continua;

4.4.6 Algoritmo de Substituição de Página Menos Recentemente Usada

• Quando ocorre falta de página, descarta-se a página que não é usada há mais tempo;
• Lista Encadeada de todas as páginas na memória;
• Página mais recentemente usada no início da lista e a pagina que foi usada há mais tempo no fim da lista ;
• Lista deve ser atualizada a cada referência da memória;
IMPLEMENTAÇÃO EFICIENTE EM HARDWARE • Outra implementação seria, manter um contador, físico, em cada entrada da tabela;
• Ao acessar uma página, atualiza-se este contador na entrada da tabela correspondente;
• Página com menor contador é selecionada;
• De tempos em tempos esse contador é zerado;
IMPLEMENTAÇÃO EFICIENTE EM SOFTWARE • Contadores relacionados a uma pagina, setados em 0;
•A cada trava do relógio, O SO percorre todas as páginas na memória, o bit R é alocado;
• Contadores nunca serão zerados;
• Sem modificação a pagina menos frequentemente usada seria a selecionada o que nem sempre é uma boa opção;
• Com modificação: Contadores deslocados 1 bit a direita a cada bit R analisado;
• Normalmente, basta haver um contador com 8 bits e períodos de 20 ms entre os clocks. Se uma página não foi acessada a 160 ms, provavelmente não é mais relevante;
A figura abaixo ilustra esse algoritmo de solução baseado em software:

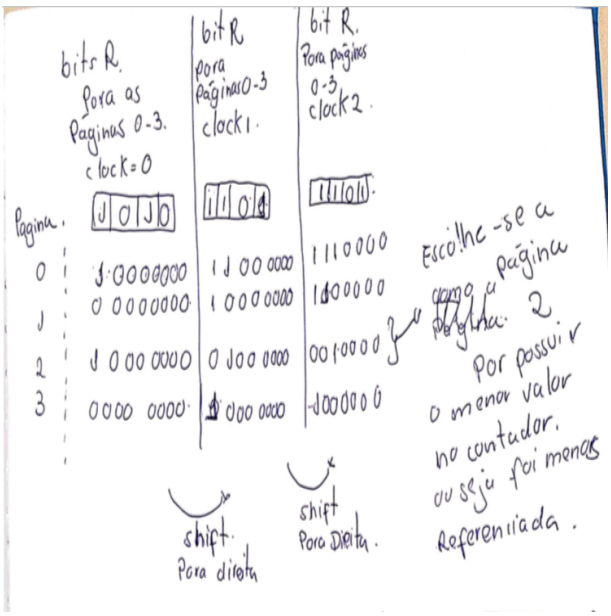


Figura 14: LRU em Software

4.4.7 Algoritmo de Substituição de Página de Conjunto de Trabalho

• Paginação por Demanda: Estratégia em que há paginação apenas quando a falta de página é detectada, ou seja sempre que há necessidade da mesma e não se tem disponível, passando a gerar poucas faltas de páginas;
• Localidade de referência, durante sua execução, o processo só vai referenciar uma fração relativamente reduzida de suas páginas ;
• Conjunto de Trabalho refere-se ao conjunto de páginas referenciadas pelas ultimas K referencias da memória, pelo principio da localidade e referencia o conjunto tende a ser limitado;
• Ultrapaginação, Trata-se da situação em que a memória é insuficiente para o conjunto de trabalho, causando inúmeras faltas de páginas, ocorrendo perda de desempenho;
• Pré-Paginação, carrega-se paginas, antes da execução na suposição de que elas serão utilizadas;
• O objetivo do algoritmo é encontrar uma página, que não esteja presente no conjunto de trabalho e removê-la, da memória.
• Por ser custoso manter todas as informações do conjunto de trabalho, é realizada uma aproximação, usando o tempo em que cada página foi acessada;
• O tempo é atualizado de acordo com o bit R da página ;
• Se o Bit R = 0, ou seja não foi referenciada no momento, verifica-se a idade da página, isto é, Tempo Virtual - Última Referência;
• Se a idade for maior que uma constante definida pelo conjunto de trabalho, implica-se que essa página não pertence ao conjunto de trabalho e poderá ser removida ;
• Toda a tabela é usada na busca por informações; Para melhorar mescla-se com o algoritmo do relógio, implementando uma lista circular de tempo que periodicamente atualiza-se em cada página que tenha o bit R=1;
• O algoritmo torna-se eficiente e simples, e bastante utilizado;

4.5 Questões de Memória Virtual

4.5.1 Política de Alocação

Política Local: O sistema aloca uma fração da memória pra cada processo. Já uma Política global, trata-se de quando o sistema aloca molduras de paginas aos processos em execução; Uma política global, funciona melhor pois em frações de memória, sistemas com grande necessidade de memória ocorrerá ultrapaginação. Com a política global, é necessário iniciar com um número de molduras relativamente proporcional ao seu tamanho. Tentando atualizar essa quantidade, de acordo com a quantidade de falta de páginas.

4.5.2 Controle de Carga

Sempre que o conjunto de trabalho, superam a capacidade da memória, ocorre ultrapaginação, o algoritmo PFF, identifica que todos necessitam de mais memória e nenhum tem para dispor. Resolve-se isso levando processos para o disco, sempre analisando quais deixar na memória.

4.5.3 Tamanho de Página

Sistema Operacional pode escolher o tamanho da página independentemente da implementação física. Algumas vantagens para páginas pequenas: Melhor aproveitamento da memória física; Mais programas usando a memória; Diminuir desperdícios na ultima página de cada segmento; Já sua principal desvantagem encontra-se na hora da transferência em que por ser somente uma página por vez acaba necessitando de mais trabalho.
Algumas vantagens para páginas grandes: Tabelas de páginas reduzidas; Transferência de uma página por vez da memória por disco e vice-versa; Espaços distintos para endereçamento e dados;

4.5.4 Páginas Compartilhadas

Processos executando o mesmo programa poderiam compartilhar as páginas, o processo ficarão mais simples se o espaço de endereçamento estiverem separados, até mesmo página de dados podem ser compartilhadas, mas apenas para leitura; Problemas de compartilhamento encontram-se quado processos terminam e é necessário saber se existia compartilhamento de página, havendo a necessidade de implementação de uma estrutura de dados. Podem ocorrer muitas faltas de páginas, devido ao controle de página para evitar ultrapaginação da outra. Bibliotecas compartilhadas, nada mais são que códigos compilados para serem compartilhados entre diversos programas diferentes durante sua execução. Eles são distribuídos como arquivos ".so" em "/usr/lib/" no caso do linux, para sistemas Windows temos as DLLs .

4.5.5 Política de Limpeza

Deamons de Paginação: Responsáveis por garantir um estoque de molduras de páginas disponíveis. Um relógio com dois ponteiros, pode ser uma implementação, onde o primeiro limpa os deamons de paginação, que por conseguinte limpa as páginas sujas. Facilitando a alteração de quando essa página será levada para o disco.

4.5.6 Backup de Instrução

É de extrema relevância manter armazenado as informações do contador de programa, no inicio de cada instrução. Outros registradores, que por efeitos colaterais foram alterados devem ser salvos, para uma possível retomada e correta execução.

4.5.7 Retenção de Páginas

Quando se tranca na memória, as páginas que contém buffer de E/S de forma que fica inviável retirá-las de lá. É importante para que certos processos de copia ou obtenção de informação sejam feitos de forma eficiente e não ocorra duplo trabalho ou situações semelhantes.

4.5.8 Memória Secundária

Existência da memória secundária, chamada Swap para Unix, ou uma alocação de partição para o swap. A área de troca pode ter um espaço para todos as páginas em sequência, a partir do endereço virtual e inicial; Possibilidade de se ter um mapa de disco;

4.5.9 Política VS Mecanismo

Escalonador de páginas encontrado no espaço do usuário trata as falta de página e implementa algum algoritmo de substituição, ficando a cargo do Núcleo o tratamento da MMU. Logo o mecanismo encontra-se no manipulador de falta de página que faz parte do núcleo e a política por conta de um escalonador de página externo que executa no espaço de usuário, para tal é necessário que esse paginador tenha acesso a informações relevantes para análise das páginas;

4.6 Segmentação

O método da Paginação resolve o problema da fragmentação externa em memória unidimensionais. Contudo em muitos problemas a memória possui mais de 1 dimensão e um outro problema de peso e que o espaço de endereçamento (lógico) de um processo é encontrado de forma linear. Para tanto, o método da segmentação permite realizar a visualização de um espaço de endereçamento como um conjunto de segmentos de espaços de endereçamento independentes. Segmentos diferentes, podem ter tamanhos diferentes e esses podem ser alterados durante a execução. O compartilhamento de rotinas é facilitado.

4.6.1 Segmentação com Paginação em MULTICS e em Pentiun

Cada programa no MULTICS tem uma tabela de segmentos, com um descritor para cada segmento. Potencialmente poderia existir mais de quatro milhões de entrada na tabela, essa por sua vez forma um segmento e que também é paginado.O descritor informa se o segmento se encontra ou não na memória e que sua respectiva tabela estará na memória. Se estiver na memória, seu descritor contém um ponteiro de 18bits para sua tabela. Pelo fato do endereçamento ser de 24bits e as páginas são alinhadas em 64 bytes, apenas 18bits são necessários no descritor para armazenar o endereço da tabela. Cada segmento é um espaço de endereçamento comum, podendo também passar pela paginação. O tamanho da página usualmente é de 1024 palavras. O endereço consiste de duas partes : O segmento e o endereço dentro do segmento, que por sua vez é seccionado em número de páginas e uma palavra interna da página; O endereço possui 34 bits sendo normalmente, 18 bits para número de segmentos, 6 bits para número de páginas e 10 bits de deslocamento; A figura abaixo retirada do livro [1] ilustra a memória virtual do MULTICS:

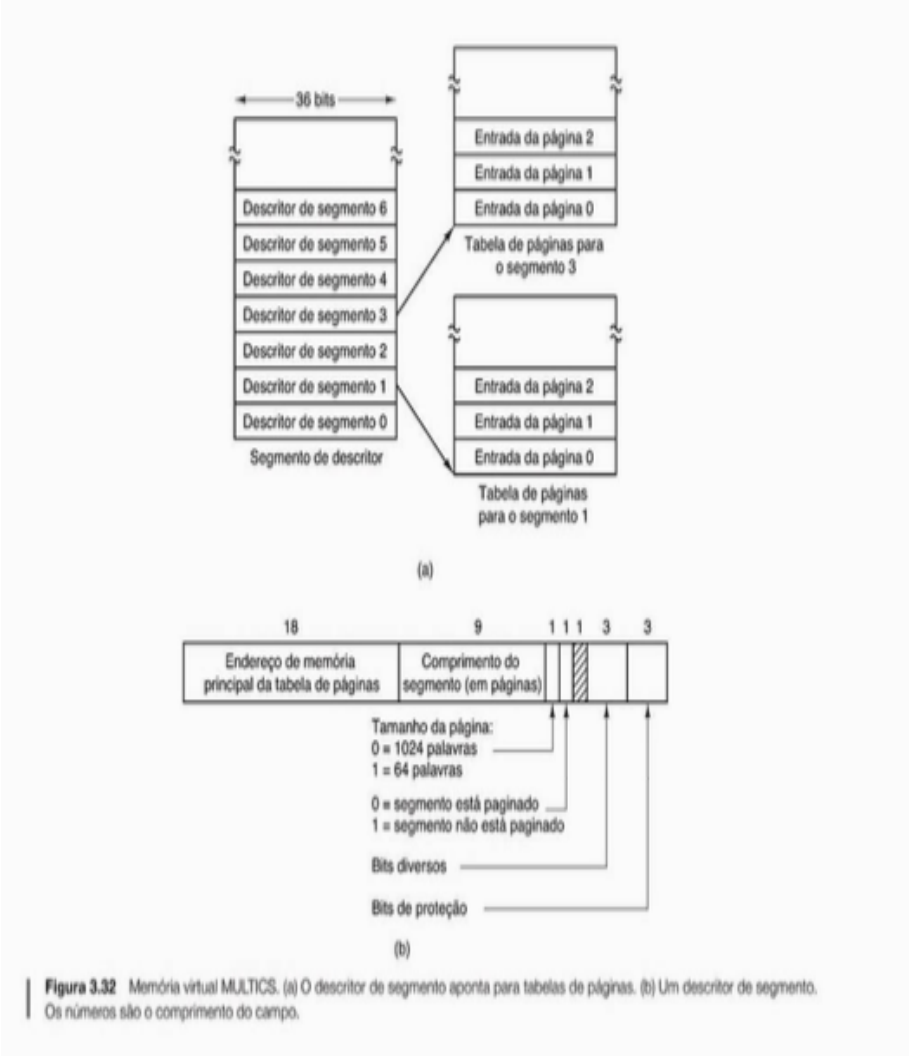


Figura 14: Multics

Enquanto o MULTICS possui 256k de segmentos independentes, cada um com até 64k de palavras de cerca de 36K, No Pentium há 16k de segmentos em independência, cada um com até 1 bilhão de palavras contendo cada uma 32 bits, uma mudança notável. Poucos programas necessitam de mais segmentos, então o que importa é o tamanho do mesmo e não a quantidade. A parte principal da sua memória é composta por duas tabelas de descritores locais (TDLs) e a tabela de descritores globais (TDG).Um programa nesse sistema possui sua própria TDl mas compartilha de uma única TDG com os demais. Na TDL são encontradas informações como: Dados, Código, Pilha segmentos locais etc. Já TDG descreve os segmentos dos sistemas. O acesso a memória se dá pelo par seletor e deslocamento, a figura abaixo representa um seletor e a seguida o descritor de segmentos no pentium

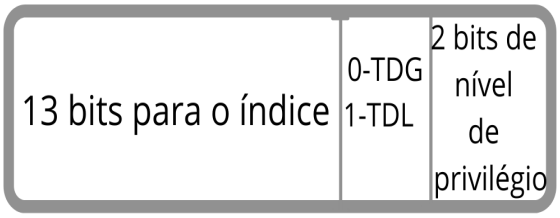


Figura 15: Seletor no Pentium

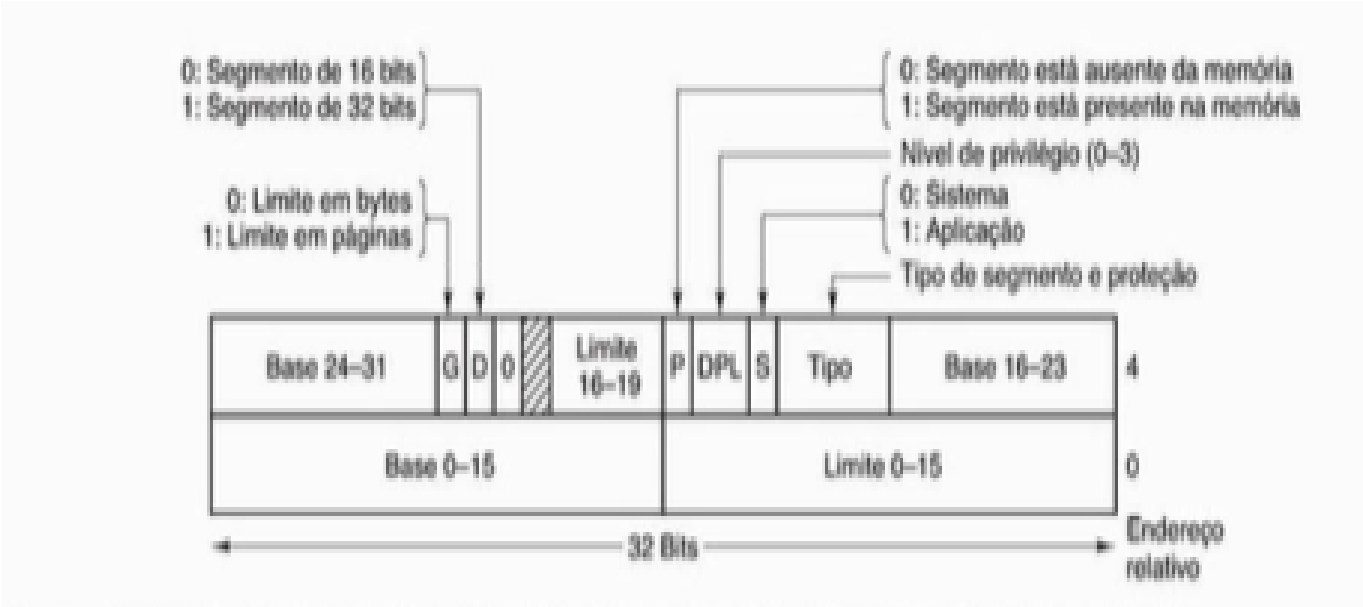


Figura 16: Descritor de segmento no Pentium [1]

References

[1] Andrew S Tanenbaum and Nery Machado Filho. *Sistemas operacionais modernos*, volume 3. Prentice-Hall, 1995.

Swart, Jacobus W. "Evolução de microeletrônica a micro-sistemas." CCS e FEEC-Unicamp (2000).

Carro, Luigi, and Flávio Rech Wagner. "Sistemas computacionais embarcados." Jornadas de atualização em informática. Campinas: UNICAMP (2003).

Maziero, Carlos A. "Sistemas operacionais: conceitos e mecanismos." Livro aberto. Acessível em: http://wiki. inf. ufpr. br/maziero/lib/exe/fetch. php (2014): 27.

https://edisciplinas.usp.br/pluginfile.php/3404408/mod$_resource/content/1/Aula07_Historia_SistemasOperacionais.pdf$

WILLRICH, Roberto. "INE5602 Introdução à Informática." Florianópolis: UFSC, Depto. de Informática e de Estatística, Curso de Graduação em Sistemas de Informação (2000).

FAINA, Luis F., Eleri CARDOZO, and Mauricio MAGALHÃES. "Introdução aos Sistemas Operacionais." Dep. de Eng de Computação e Automação Industrial. Faculdade de Engenharia Elétrica e Computação, UNICAMP (1992).

https://ycpcs.github.io/cs420-fall2017/lectures/lecture3$_services_and_system_calls.pdf$

Lotti, Luciane Politi, and Alysson Bolognesi Prado. "Sistemas virtualizados–uma visão geral." (2010).