PCS 3446 - Sistemas Operacionais

Prof. João José Neto

AULA 06

Administração de Memória (5)

Memória Segmentada simples e com paginação

Segmento

- **Segmentação** é outra forma de dividir um programa para ser alocado na memória em áreas não contíguas.
- Enquanto a paginação fraciona o programa em áreas de tamanho uniforme, a segmentação divide-o em **segmentos**, áreas de tamanhos variados, geralmente refletindo a sua estrutura lógica.
- Ao contrário da paginação, que exige a divisão da memória em blocos do mesmo tamanho das páginas, a segmentação aloca na memória **partições de tamanhos variados**, iguais aos dos segmentos.

Espaço bidimensional de endereçamento

- Um programa pode assim ser representado na forma de uma **árvore de segmentos**.
- **Cada segmento** possui internamente seu próprio **espaço linear de endereçamento**.
- Dessa forma, para endereçar qualquer ponto do programa é preciso **indicar o segmento**, e depois, **o endereço local, interno** ao segmento.
- Nas instruções de referência à memória, os bits do campo de endereçamento devem estar agrupados formando essas duas componentes:

número do segmento	endereço interno ao segmento

Segmentação simples

- É possível implementar segmentação **com ou sem paginação**, simplesmente como mais uma forma de gerenciar o uso da **memória física**, e/ou aplicando-a como técnica de **virtualização**.
- A **segmentação física** opera assim:
  - Divide-se o programa em **segmentos**
  - Aloca-se uma **partição** para cada segmento
  - Monta-se uma **tabela de mapeamento** de segmentos, indicando a posição de cada um na sua partição física
  - Trata-se cada **segmento** como se fosse uma **partição relocável**, usando como base seu endereço de alocação
  - A **cada referência** feita à memória, determina-se a **base** correspondente, a ser usada na **relocação** dinâmica.

Procedimentos puros

- Procedimentos puros são aqueles cujo **código permanece invariáve**l durante toda a sua execução.
- A **alocação segmentada** de memória facilita a implementação de procedimentos puros, bastando para isso alocar sua parte **invariável** separadamente daquelas que contenham **dados modificáveis**.
- **Cada instância** de um procedimento puro deve ter **sua própria área de dados**.
- Dessa maneira, **a parte não modificável do programa** pode ser **compartilhada** em tempo de execução.

Efeitos colaterais

- É o nome que se dá às **alterações do conteúdo** de áreas de memória pela execução de **comandos imperativos**.
- Efeitos colaterais impedem aos programas que operem como procedimentos puros, ou seja, fazem com que se **tornem variáveis, dinâmicos, não-reentrantes**.

Segmentos não-modificáveis

- Segmentos que implementam **procedimentos puros podem ser reinstanciados**, simultaneamente **reutilizados**, e arbitrariamente **compartilhados**.
- Em ambientes com memória segmentada, **assegura-se que as partes estáticas possam ser reinstanciadas** de forma automática, assegurando assim robustez e **integridade dinâmica** aos programas.

Procedimentos reentrantes

- Procedimentos **reentrantes** são obtidos trivialmente empregando-se segmentos que implementem códigos de **procedimento puro**, que podem ser ativados inúmeras vezes sem alteração.
- **Múltiplas instâncias simultâneas** compartilham o código mas possuem áreas de dados separadas.

Procedimentos recursivos

- A **recursão** pode ser facilmente implementada com **procedimentos puros**, garantindo sua **transparência referencial** (substituindo-se qualquer expressão pelo seu valor, o comportamento do procedimento não varia).
- Procedimentos **recursivos** costumam ser implementados dividindo-se o programa em **segmentos para código e segmentos para dados**.
- Segmentos referentes a **áreas de dados e parâmetros** são **reinstanciados** a cada chamada recursiva, enquanto os **segmentos de código** vão sendo sucessivamente **reutilizados** sem a necessidade de reinstanciação, por serem estáticos.

Alocação dinâmica de variáveis locais

- Programas **estruturados em blocos** costumam efetuar **alocações dinâmicas** de suas variáveis locais, no momento em que os **blocos** em que estas foram declaradas são ativados pela primeira vez.
- A alocação segmentada dá apoio a essa forma de gerenciamento das partes variáveis do programa, permitindo que as **áreas necessárias de memória** sejam **alocadas à medida da necessidade**, em tempo de execução.

Segment map table

- Essa tabela, como o nome indica, implementa o **mapeamento do conjunto de segmentos** nos respectivos endereços de alocação, na memória física.
- Assim, indexando-se essa tabela com o **número de qualquer segmento** do programa, obtém-se como resultado o **endereço inicial de alocação** desse segmento na memória física.
- Usado como **base de relocação**, esse endereço permite resolver dinamicamente os endereçamentos feitos a qualquer posição de memória interna ao segmento.

Segment name table

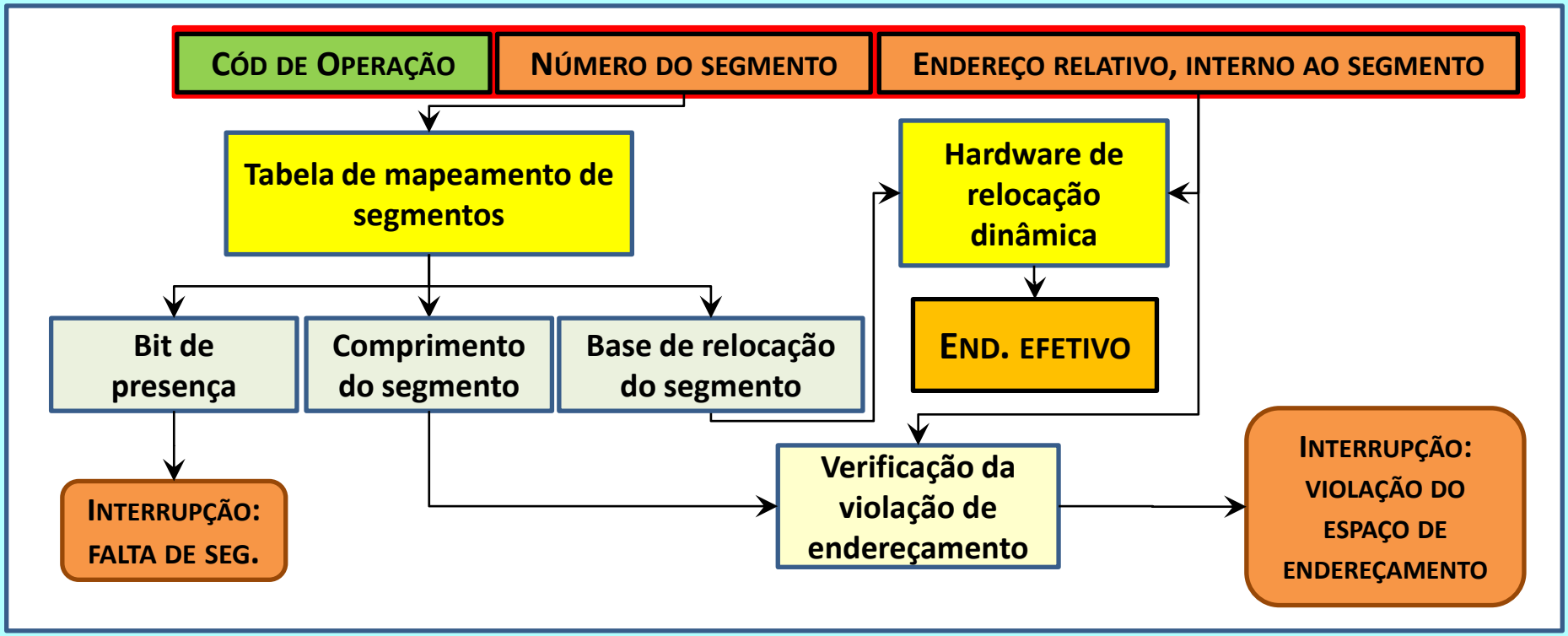
- A **tabela de nomes de segmentos** deve acompanhar o código do programa para que, no momento da sua carga, possa ser construída a tabela de mapeamento de segmentos.
- Naturalmente, para que o endereçamento de áreas internas ao segmento possa ser efetuado corretamente, todas as **referências simbólicas** aos nomes dos segmentos devem ter sido previamente **resolvidas**.

Mapeamento de endereços em sistemas com memória segmentada

- Em sistemas com memória segmentada, os programas referenciam **endereços lógicos bidimensionais:** (segmento, endereço interno).
- Cada **segmento**, de acordo com a tabela de mapeamento de segmentos, está alocado em alguma **partição da memória**, cujo **endereço inicial consta na tabela**.
- Esta, indexada pelo **número do segmento**, fornece o **endereço inicial da partição**, a ser usada como **base de relocação** para a determinação do endereço absoluto correspondente à posição (relativa) referenciada de memória.
- Quando se implementa **memória virtual segmentada**, caso o segmento referenciado **não esteja presente** na memória, o hardware deverá gerar um **pedido de interrupção**, solicitando a **carga do segmento** na memória.

Hardware de mapeamento de segmentos

- Instruções de referência à memória têm a estrutura abaixo.
- Ao executar a instrução, o número do segmento indexa a tabela de mapeamento, recuperando a base de relocação.
- Esta é somada ao endereço interno, obtendo-se o endereço efetivo.
- A ausência do segmento e endereços incorretos provocam interrupções.



Active Segment Table

- Ao longo da execução, as instruções do programa vão efetuando **referências aos seus diversos segmentos**.
- Em ambientes em que a segmentação é usada para **virtualizar a memória**, somente segmentos que já foram referenciados poderão estar em atividade em cada instante.
- Na **tabela de segmentos ativos** uma lista deve ser mantida e dinamicamente atualizada, indicando todos os segmentos do programa que estão em atividade no momento, ao longo de toda a execução do programa.

Proteção de segmentos

- Sendo partes lógicas completas do programa, os **segmentos podem ser devidamente classificados**, de acordo com o papel que seu conteúdo desempenha na execução do programa.
- Podem ser adicionados ***tags*** a cada segmento, com a finalidade de informar sua função na lógica do programa, permitindo assim que o hardware execute **operações dinâmicas de proteção**, evitando consequências no caso da ocorrência de alguma tentativa de violação de direitos de acesso.

Bit de falta de ligação

- Referências a **segmentos ainda não usados** podem ser mantidas na forma de **referências simbólicas**.
- Na primeira vez em que são referenciados, é necessário que o **ligador** (***linker***) promova as devidas conexões entre o segmento referenciado e o programa que referencia tal segmento.
- Em sistemas que oferecem este recurso, um **bit de falta de ligação** pode ser usado para indicar, na tabela de mapeamento de segmentos, se o segmento em questão já foi ligado ao programa ou ainda não.

Interrupção de falta de ligação

- Quando todos **os segmentos** do programa já tiverem sido carregados **na memória**, os correspondentes mapeamentos podem ser imediatamente efetuados, e **nenhuma interrupção** acontecerá.
- Caso contrário, ao ocorrer uma referência a um **segmento ainda não ligado**, a tabela de mapeamento informará que a ligação deste segmento ainda não se realizou, e portanto **um serviço do sistema deve ser acionado** para que tal segmento seja devidamente ligado ao programa.

Ligação dinâmica dos segmentos

- Quando um **segmento nunca antes referenciado** é acionado pela primeira vez, o seu **bit de falta de ligação,** ainda ligado, dispara uma **interrupção** no hardware.
- Isto sinaliza ao sistema operacional a **necessidade,** na ocasião, **da ligação dinâmica** daquele segmento.
- O atendimento dessa interrupção acionará o ***linker***, o qual promoverá a **ligação dinâmica** do segmento.
- É este o mecanismo que aciona as **DLL's** (***dynamic library link***), ou seja, bibliotecas de ligação dinâmica, em que a ligação só é feita se a execução da função de biblioteca for realmente requisitada pelo programa.

Bit de presença do segmento

- Quando o **segmento ainda não foi carregado** fisicamente na memória, um bit da tabela de mapeamento de segmentos (**bit de presença**) é mantido desligado, informando que seu endereço físico ainda não foi preenchido pelo sistema.
- Toda vez que um segmento nestas condições é **referenciado**, ocorre uma interrupção, cujo tratamento promove sua **cópia para a memória física,** de modo que possa ser executado.
- Nesse tratamento, o seu **bit de presença é ligado**, indicando ao hardware que o correspondente conteúdo da tabela de mapeamento de segmentos já aponta corretamente o endereço físico no qual o segmento está carregado.

Interrupção de falta de segmento

- Se for referenciado um segmento cujo **bit de presença esteja desligado**, isso provoca uma **interrupção** (de **falta de segmento**), informando ao sistema operacional sobre a necessidade de que uma cópia do segmento seja feita, do disco para a memória.
- Ao tratar essa interrupção, o sistema operacional põe em execução o **carregador** (***loader***), que aloca ao segmento em questão uma partição livre na memória principal, para aí carregar o segmento faltante referenciado.
- Nessa ocasião, no campo de endereços da tabela de mapeamento deverá ser registrado o endereço inicial da partição alocada ao segmento, a ser usado como **base de relocação** no momento da execução, para que seja feita a resolução dos seus endereços (relativos) referenciados.

Memória Virtual Segmentada

- O uso das **interrupções de falta de ligação** e **de falta de segmento** propiciam **virtualizar** a memória segmentada.
- Seu mecanismo de funcionamento é **similar ao da paginação virtualizada**, embora envolva mais tabelas:
  - **Carrega-se**, ao ser acionado o programa, seu **primeiro segmento** apenas, e sinaliza-se que todos os demais segmentos do programa estão ainda ausentes e não ligados.
  - A cada **interrupção de segmento não ligado** promove-se a **ligação dinâmica** correspondente a tal segmento.
  - A cada **referência a um segmento ausente**, a interrupção associada provoca a **carga dinâmica** do segmento na memória.
  - Em cada um desses tratamentos de interrupção, **atualizam-se** adequadamente **as tabelas de mapeamento** de segmentos, de segmentos **ativos**, de **nomes** de segmentos, **e promove-se a reexecução** da instrução que provocou a interrupção.

Segmentos compartilhados

- Como foi mencionado, os segmentos que implementam **procedimentos puros** podem ser **compartilhados** sem problemas, uma vez que a execução desses segmentos não altera o seu código.
- O sistema operacional deve manter uma tabela informativa de **quais segmentos estão instanciados mais de uma vez**, para evitar que sejam removidos da memória antes que todas as suas instâncias tenham completado as respectivas atividades.

Segmentos de ligação

- A **ligação dinâmica** traz uma pequena **dificuldade à implementação de procedimentos puros**: torna necessária a **substituição das referências simbólicas** aos segmentos (antes da ligação) **por referências numéricas correspondentes** (depois da ligação), e isso acarreta modificar o conteúdo do segmento.
- Para contornar este problema, **segmentos de ligação**, separados do código, são usados para guardar essas informações, e são referenciados pelo código.
- A alteração dos segmentos de ligação **concentra todas essas modificações em um único segmento**, evitando que o próprio código tenha de ser modificado.

Segmentos executáveis

- São os **segmentos de código**, propriamente ditos.
- Em ambientes em que são implementados **procedimentos puros**, os segmentos executáveis são constituídos por **código estático**.
- Esses segmentos são exclusivamente utilizados para **execução**, não podendo ser lidos nem modificados por programas operando em modo usuário.
- A **proteção de acesso** é implementada de forma trivial pelo próprio hardware, que gera um pedido de **interrupção por violação do espaço de endereçamento** caso haja alguma tentativa de acesso indevido.

Memória virtual segmentada com paginação

- Em sistemas com memória segmentada, cada **segmento** pode ser **tratado como uma sequência de páginas**, a exemplo do que foi feito na alocação de memória particionada.
- Isso permite empregar diversos métodos já estudados em paginação, incluindo a virtualização.
- Neste cenário, **segmentos e páginas não presentes na memória física são requisitados dinamicamente**.
- Com isso, é possível unir as vantagens dos dois métodos de implementação de **memória virtual**, através do uso de um **esquema híbrido**.
- Em consequência, essa técnica também **neutraliza** alguns dos **pontos de desvantagem** que a paginação e a segmentação apresentam, um em relação ao outro.

Hardware necessário

- Descreve-se a seguir uma implementação desse esquema misto de administração de memória, tal como foi realizada já nas suas origens, **no sistema IBM/370**.
- Outros computadores que empregam esta técnica o fazem segundo esquemas diversos, todos conceitualmente bastante parecidos.
- Tratando-se de meros detalhes de implementação, tais variantes não serão estudadas neste contexto.

Formato dos Endereços Lógicos

- O sistema IBM/370 utiliza **páginas de 2K ou 4K bytes**, e **segmentos** com tamanho de **64K ou 1024K bytes** no máximo.
- Por exemplo, um caso com páginas de 4K e segmentos de 64K (máximo) teria **endereços com o seguinte formato** (1 palavra = 32 bits):
  - [0-7] campo **não utilizado** (8 bits)
  - [8-15] número do **segmento** (8 bits)
  - [16-19] número da **página** (4 bits)
  - [20-31] **endereço relativo, interno** à página (12 bits)

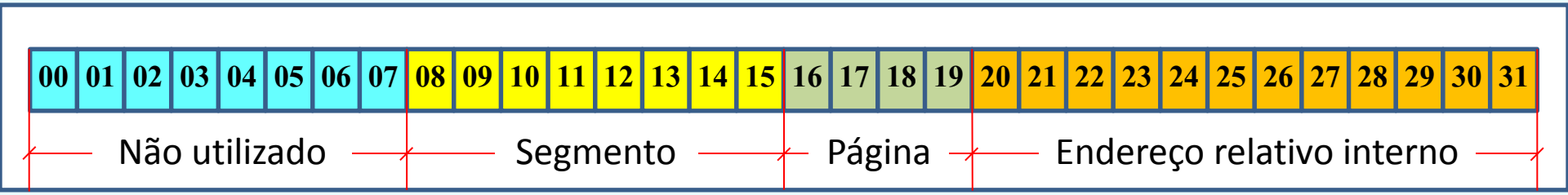


Tabela de Jobs (indexada pelo número do job)

- **Cada linha** desta tabela contém as informações relativas ao **correspondente job**, ou seja, o registrador da tabela de segmentos (*segment map table* – SMT) de um job, formado de dois campos, contendo:
  - O **comprimento da SMT** do job correspondente (mod 16)
  - **Ponteiro para a SMT** do job, ou seja, endereço dessa SMT na memória do computador (mod 64)

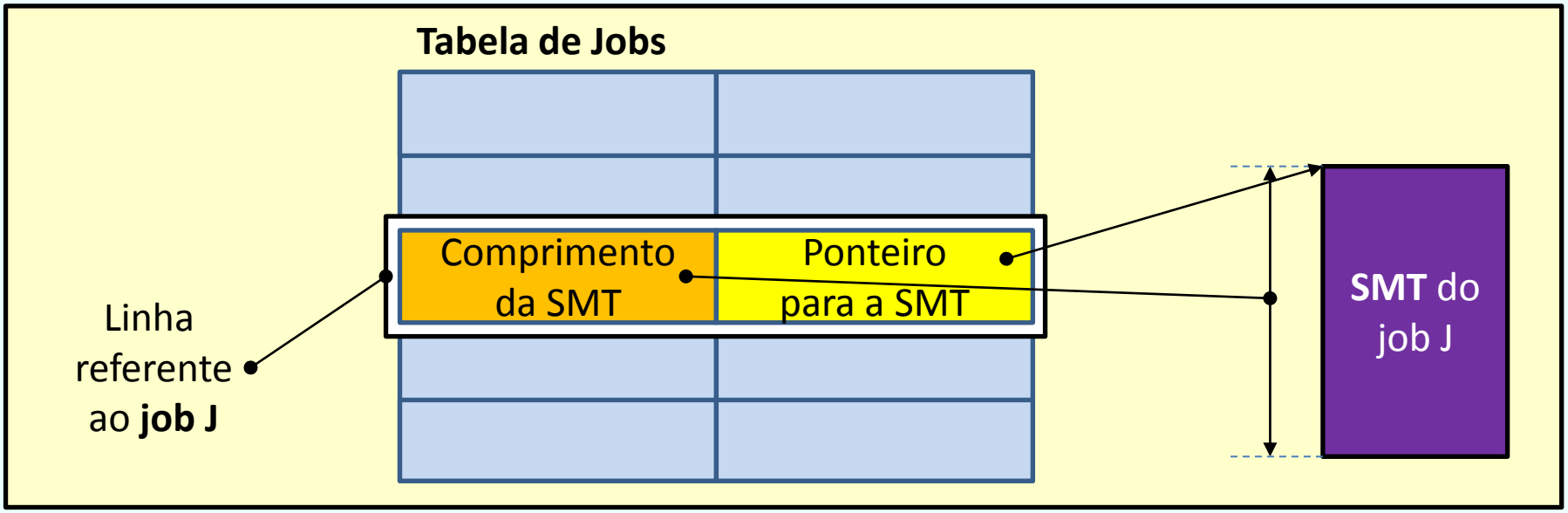
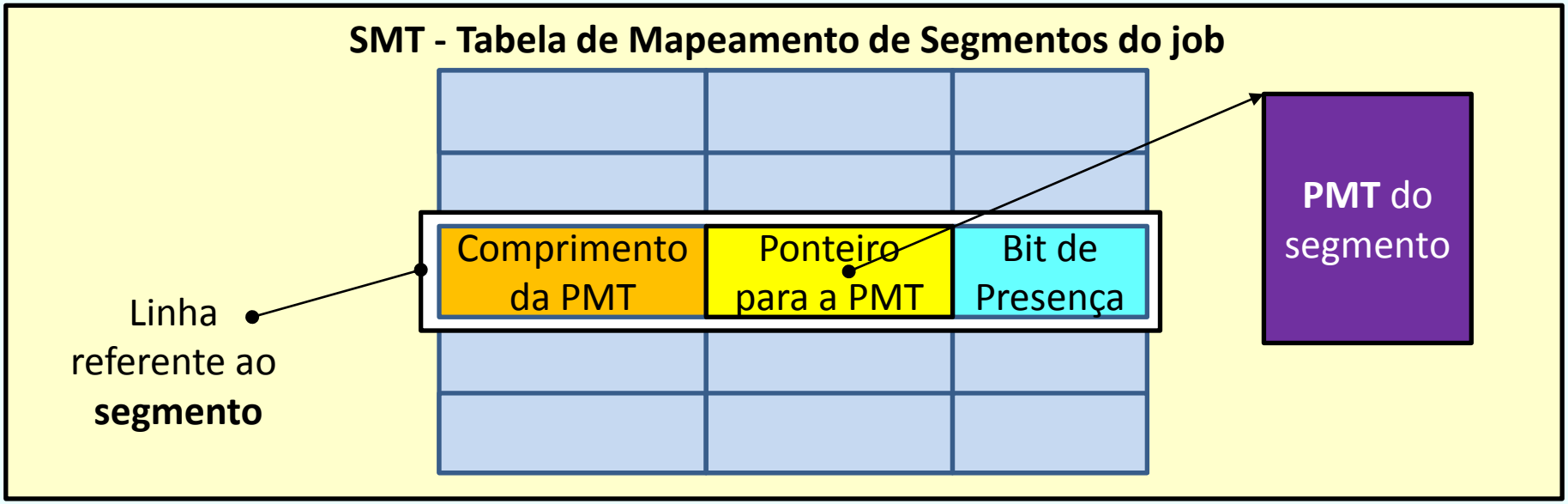


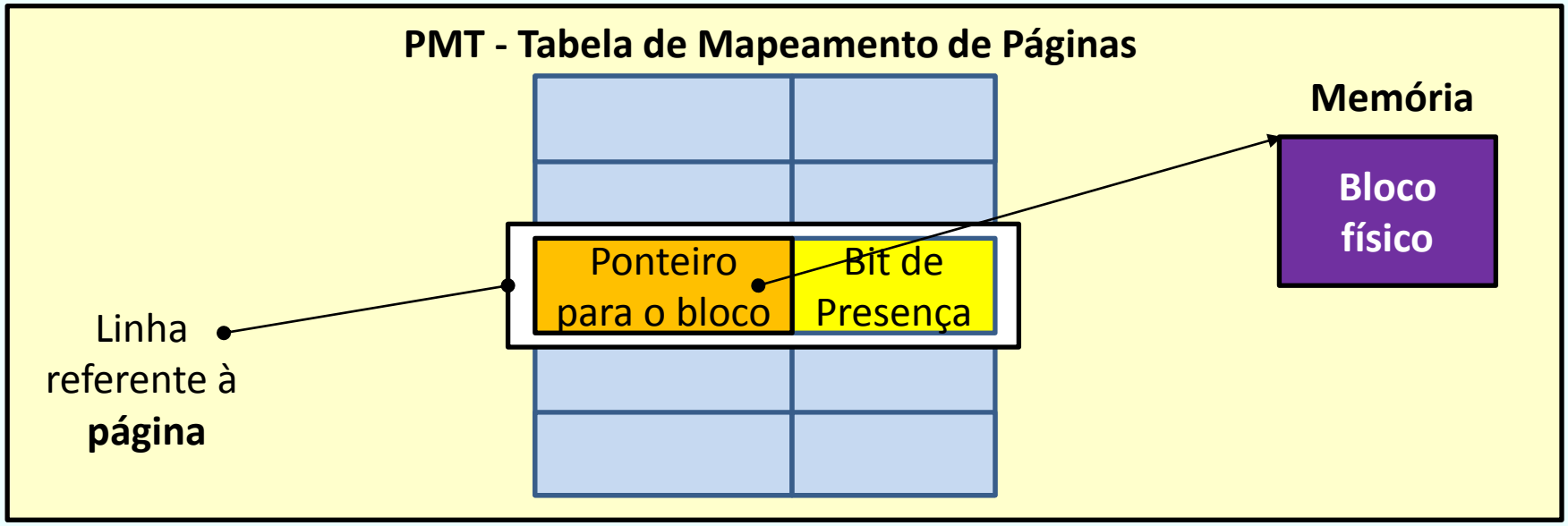
Tabela de Mapeamento de Segmentos (SMT) (indexada pelo número do segmento)

- Cada linha desta tabela refere-se a um segmento do programa.
- **Cada elemento desta tabela** descreve tal segmento através de três informações associadas a tal segmento:
  - **Comprimento da PMT** (*page map table*) daquele segmento
  - **Ponteiro para a PMT** correspondente ao segmento, ou seja, o endereço de tal PMT na memória do computador
  - **Bit de presença**, referente ao segmento em questão



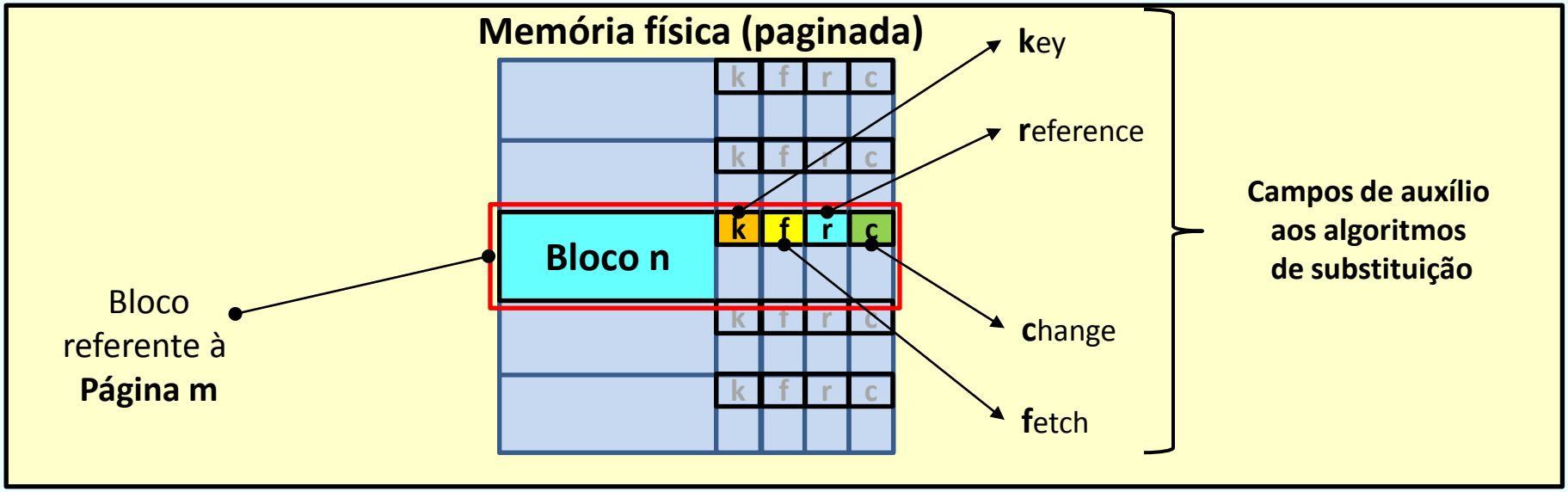
(PMT) Tabela de Mapeamento de Páginas (indexada pelo número da página)

- **Cada linha** desta tabela refere-se a **uma página do segmento**.
- **Cada elemento** desta tabela descreve tal página através de duas informações associadas a tal página:
  - Ponteiro para a posição da memória física que aloja a página, ou seja, o **número do bloco da memória física** no qual tal página foi alocada
  - **Bit de presença**, relativo à página em questão



Memória Física (indexada pelo número do bloco)

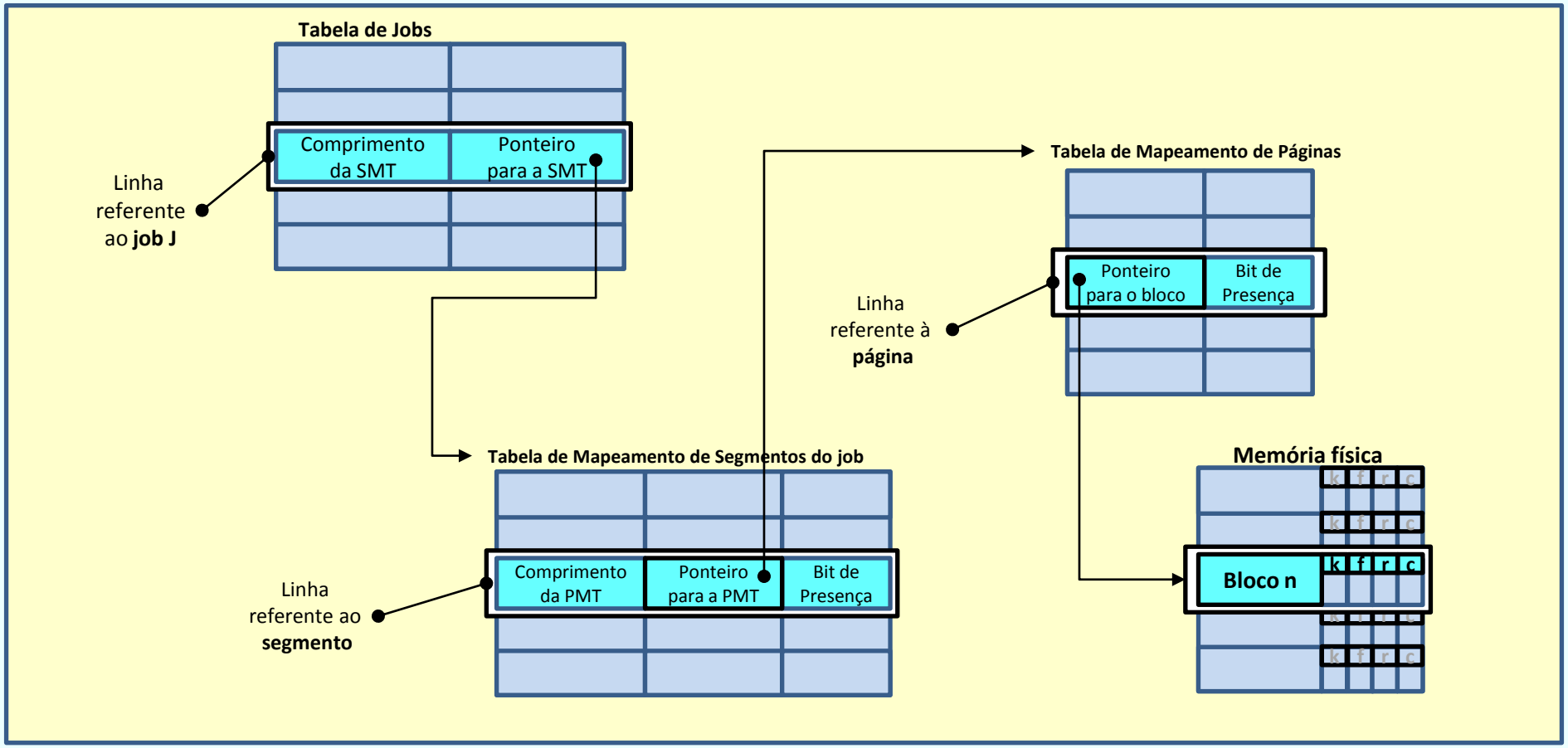
- Cada elemento da memória física é um **bloco do tamanho de uma página**, ao qual estão associados quatro campos, dinamicamente manipulados pelo sistema operacional, cujos conteúdos são informações para proteção e para simplificar o algoritmo de substituição:
  - **k (key)** – chave associada à página: deve ser **igual ao campo lock**
  - **f (fetch)** – bit que indica que o bloco foi **acessado recentemente**
  - **r (reference)** – indica que o bloco foi referenciado recentemente
  - **c (change)** – indica que o bloco foi modificado recentemente



- O operando contido no **campo de endereço** das instruções de referência à memória (portanto, um **endereço lógico**) reflete as características lógicas do espaço de endereçamento do programa
- Portanto, no momento da execução, deve ser **mapeado** adequadamente, para que a partir dele seja calculado o correspondente **endereço físico** na memória.
- Para isto, o computador deve dispor de um sofisticado **hardware de mapeamento**, que consta de várias **tabelas de apontadores**, bem como de diversas **interrupções especiais**, relacionadas com os mecanismos implementados de **paginação, segmentação, proteção e ligação dinâmicas**.

Mapeamento (simplificado) da Memória Segmentada com Paginação (no IBM/370)

- Juntando as partes, obtém-se o esquema simplificado seguinte:



Relembrando

- Observe-se que **todas as outras técnicas** já estudadas de alocação de memória estão compreendidas nesse esquema, como **casos particulares**.
- Para **evitar a degradação no desempenho** da máquina, o sistema IBM/370 utiliza uma espécie de cache em hardware chamado ***Translation Lookaside Buffer***
- Este é implementado como uma ***memória associativa*** de rascunho, exclusivamente utilizada para o trabalho de **resolução dos endereços** (incluindo mapeamento).
- Essa prática **reduz em até 90% o *overhead*** causado pela necessidade de efetuar a tradução dos endereços.

Software necessário

- O software utilizado neste esquema de gerenciamento de memória **pode utilizar todos os algoritmos já mencionados** até aqui, tanto para **segmentação** como para **paginação**
- Esses algoritmos devem sofrer as devidas **adaptações** para tornar mais eficiente a programação do sistema operacional
- Isso se faz, por exemplo, **eliminando tarefas duplicadas**, preservando assim no programa final apenas um procedimento para cada tarefa.

Interrupções

- Três rotinas de tratamento de interrupção devem existir:
  - **Falta de ligação**, que ocorre quando um segmento é referenciado pela primeira vez
  - **Falta de segmento**, que ocorre toda vez que é feita uma referência a um endereço pertencente ao espaço de endereçamento de um segmento que esteja ausente da memória
  - **Falta de página**, que ocorre sempre que for referenciado um endereço interno a uma página ausente da memória
- **Todas** essas três rotinas **podem** ser ativadas (uma de cada vez), durante a execução de **uma única instrução** de referência à memória, bastando para isso que seja feita, por exemplo, uma referência a um segmento até então não carregado na memória.

Falta de Ligação

- A rotina de tratamento de interrupções de **falta de ligação** associa um **número exclusivo** ao nome simbólico do segmento referenciado, **desliga o bit de presença** correspondente na tabela SMT (*Segment Map Table*) para indicar que aquele segmento não está presente na memória, e acerta o ponteiro utilizado como ligação (*link*) para apontar o endereço do **ponto de entrada do segmento** na SMT.

Falta de Segmento

- O tratamento desta interrupção cria um elemento novo na AST (*Active Segment Table*) caso o segmento já não estiver sendo utilizado por outro processo. Acerta a PMT (*Page Map Table*), indicando que **as páginas** associadas ao segmento em questão **não estão ainda presentes** na memória, e atualiza o *File Map Table* na *Active Segment Table*. Atualiza a SMT para **apontar adequadamente para a PMT do segmento**.

Falta de Página

- (Este procedimento **substitui o loader** tradicional não paginado.)
- O tratamento desta interrupção **busca um bloco livre de memória**, efetuando, se isso for necessário, **remoções de páginas** presentes, e utilizando para tanto as **políticas disponíveis de substituição**. No bloco vazio assim obtido, é então **carregada a página solicitada**, atualizando correspondentemente a PMT.

Vantagens da segmentação paginada

- Este método acumula todas as vantagens já vistas para a segmentação e para a paginação:
  - **Dispensa a contiguidade da alocação** de memória
  - **Dispensa a alocação integral** do programa na memória
  - **Evita a fragmentação**
  - **Permite virtualização da memória**
  - **É totalmente transparente para o usuário**
  - **Compatibiliza** computadores que utilizam memórias de diferentes tamanhos
  - Permite importantes **proteções e compartilhamentos**
  - Permite o uso de **ligações dinâmicas**

Desvantagens

- Exige um **hardware adicional** oneroso (interrupções, tabelas de mapeamento, supervisão permanente do hardware)
- ***Overhead*** no processador para a **resolução de endereços** lógicos
- Dificuldade adicional na confecção do sistema operacional, o que leva a **sistemas bastante complexos e extensos**
- No nível dos segmentos, **o problema da fragmentação interna persiste**, causando um desperdício médio de meia página para cada segmento (**pior que no caso da paginação** simples, que era de meia página para cada programa)
- **Dois níveis de *thrashing*** podem ocorrer agora:
  - *thrashing* **de segmentos**
  - *thrashing* **de páginas**

Usos da técnica

- Sistemas operacionais modernos adotam essa técnica e se mostram **muito bem sucedidos**, pois operam satisfatoriamente, e isso prova que suas desvantagens podem ser superadas com bastante êxito.
- Os sistemas **MULTICS e IBM OS/VS-2** são dois exemplos históricos de antigos sistemas operacionais pioneiros, que apresentaram um sucesso muito grande com o emprego deste método de alocação.
- Este é o método de gerenciamento de memória que se mostra **o mais geral e flexível**, e que tem proporcionado na prática os **melhores resultados no desempenho resultante** do sistema computacional.

Tabela comparativa das técnicas de administração de memória

- Para finalizar o estudo da administração de memória em sistemas operacionais, apresenta-se uma **tabela comparativa** compreendendo alguns dos atributos mais relevantes:
  - Como é referenciado o **Espaço de Endereçamento**
  - Capacidade de implementar **Multi-programação**
  - Possibilidade de efetuar **Relocação Dinâmica**
  - Disponibilidade de mecanismo de **Ligação Dinâmica**
  - Capacidade de realizar a **Virtualização de memória**
  - Capacidade de processar **programas maiores que** o espaço existente de **memória física**

Efeito / Alocação	Endereçamento	Multi-programação	Relocação dinâmica	Ligação dinâmica	Virtual	Programas > memória
Contígua	Linear contíguo	não	não	não	não	não
Contígua +SO	Linear contíguo	dois	não	não	não	não
Particionada	Linear contíguo	sim	não	não	não	não
Particionada relocável	Linear contíguo	sim	sim	não	não	não
Paginada simples	Linear esparso	sim	Mapeamento de páginas	não	não	não
Paginada virtual	Linear esparso	sim	Mapeamento de páginas	não	sim	sim
Segmentada simples	Bidimensional contíguo	sim	Mapeamento de segmentos	sim	não	não
Segmentada virtual	Bidimensional contíguo	sim	Mapeamento de segmentos	sim	sim	sim
Segmentada e paginada física	Bidimensional esparso	sim	Dois mapeamentos	sim	não	não
Segmentada e paginada virtual	Bidimensional esparso	sim	Dois mapeamentos	sim	sim	sim
Swapping	Linear contíguo	intercalados	não	não	não	não
Overlay	Linear contíguo	não	não	não	não	sim