Memória Virtual

MAC 344 - Arquitetura de Computadores
Prof. Siang Wun Song

Baseado em W. Stallings

Computer Organization and Architecture

Apoio do sistema operacional por hardware

- O sistema operacional é o software que controla a execução de programas em um processador e gerencia os recursos.
- Para obter eficiência e velocidade, funções do sistema operacional como escalonamento de processos (*scheduling*) e gerenciamento de memória contam com o apoio de hardware.
- Esse apoio consiste em registradores e *buffers* especiais, e circuitaria para realizar tarefas de gerenciamento de recursos.
- Veremos agora o gerenciamento de memória e o apoio de hardware. Em especial veremos o uso de memória virtual que tem os benefícios:
  - Um processo pode executar sem ter todas as instruções e dados dentro da memória principal.
  - O espaço de memória disponível ao programa pode exceder o tamanho da memória principal.

Serviços de um sistema operacional

De forma resumida, um sistema operacional (S.O.) típico fornece os seguintes serviços. Detalhes são estudados na disciplina Sistemas Operacionais.

- Criação do programa: editores, depuradores, etc. Não fazem parte propriamente do S.O. mas são acessíveis através do S.O.
- Execução do programa: carrega o programa (instruções e dados) na memória, inicializa e prepara dispositivos de entrada e saída, arquivos e demais recursos.
- Acesso a dispositivos de entrada e saída: livra do programador detalhes e conhecimento de instruções específicas, sinais de controle, etc.
- Acesso controlado a arquivos: S.O. se precoupa com os detalhes de acesso a disco, fita, etc., além de prover mecanismos de proteção sobre direito de acesso.
- Acesso a recursos do sistema: em caso de uso compartilhado, o S.O. controla o acesso a recursos compartilhados e resolve conflitos.
- Deteção e resposta a erros: S.O cuida de erros de hardware (como erros de memória e erros de dispositivos) e erros de software (como *overflow* em aritmética, acesso proibido de posições de memória) e responde de acordo (eventualmente abortando o programa).
- Contabilidade: coleta estatísticas de vários recursos, monitora desempenho.

S.O. é parte do sistema sendo controlado

- O S.O. controla os recursos de um computador.
- Em geral um controlador de um sistema é externo ao sistema.

  Exemplo: um semáforo que controla o fluxo de automóveis.

Source: Wikimedia Commons

S.O. é parte do sistema sendo controlado

- No caso do S.O., ele é um software como qualquer outro e também usa os mesmos recursos como os demais.

Exemplo: Um carro oficial controlando o tráfego. O carro que controla também tem que usar a estrada junto com os demais veículos.

Source: Wikimedia Commons

S.O. é parte do sistema sendo controlado

Outro exemplo: automóveis oficiais do Detran controlam as rodovias para melhor fluir o tráfego. Para isso, os carros oficiais também precisam usar a rodovia, e.g. bloqueando-a para poder prover socorro em caso de acidente, ou para mudar a sinalização na rodovia numa operação descida (da Rodovia Imigrantes). Para gerenciar o bom uso da rodovia, carros oficiais precisam usar a mesma rodovia, numa aparente contradição.

S.O. é parte do sistema sendo controlado

- O S.O., como qualquer outro programa, é executado pelo processador.
- Uma parte do S.O., chamada *kernel* ou *núcleo*, reside na memória do computador, juntamente com os demais programas e dados de usuários.
- O S.O. frequentemente passa o controle do processador para executar programas de usuários e, quando necessário, precisa retomar o controle do processador.

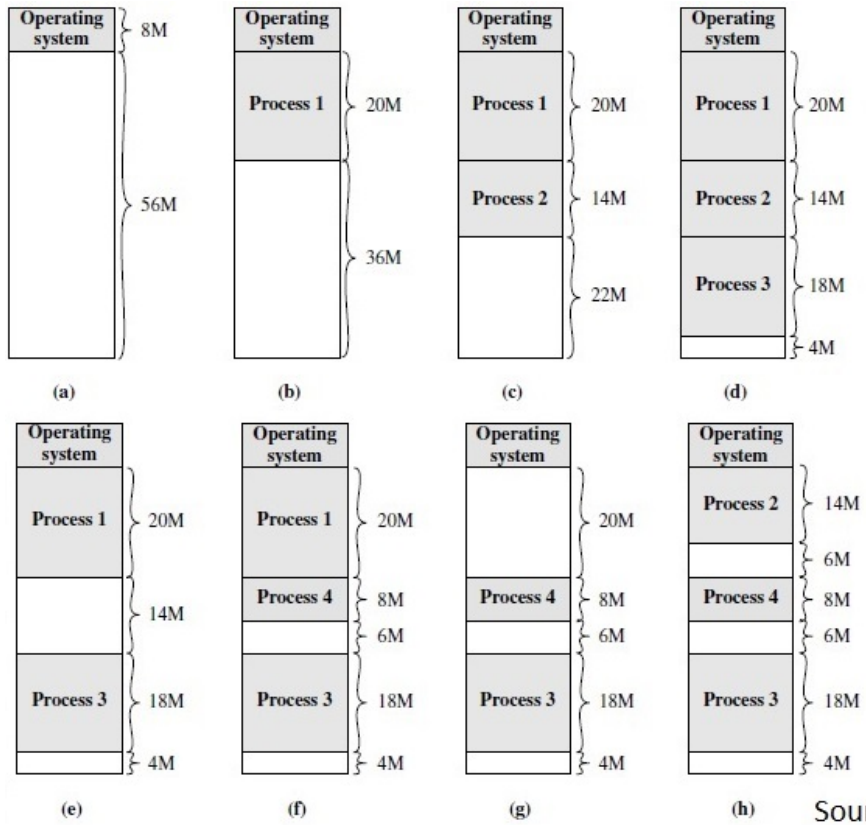
  (Lembre-se do exemplo da rodovia usada ora por usuários ora bloqueada por carros oficiais para determinadas necessidades.)
- Num sistema de mono-programação, a memória é dividida em duas partes: uma para o S.O. (monitor) e outra para o programa sendo executado.
- Num sistema de multi-programação, a parte da memória de usuário é subdividida para acomodar múltiplos processos.

Swapping

- Uma função do S.O. é o gerenciamento de memória
- Uma outra função é o escalonamento de processos. O S.O. mantém e administra três tipos de filas.
  - Novos processos que acabam de entrar no sistema aguardam numa fila de novos processos ou fila de longo prazo, tipicamente implementada em disco.
  - Processos prontos para entrar ou voltar a execução são mantidos em uma fila de processos prontos para execução ou fila de curto prazo.
  - Processos que tiveram que largar o processador para aguardar entrada e saída (E/S) são colocados em um fila de E/S.
  - Um processo na fila de longo prazo pode ser trazido do disco para a memória para execução. Ao necessitar de uma operação de E/S pode ser transferido para a fila de E/S. Terminada a operação pode ser movido para a fila de curto prazo e, eventualmente pode ser movido para a memória para execução. Tal movimento de um processo para dentro e fora da memória recebe o nome de *swapping*.

Particionamento, fragmentação e compactação

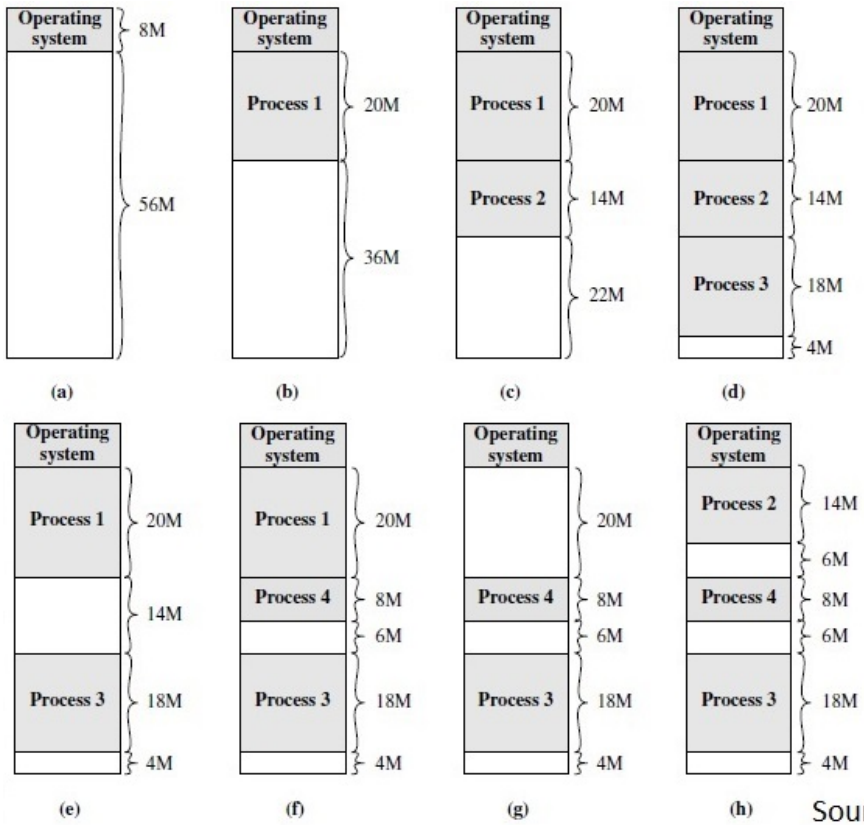
- A memória é particionada em tamanhos fixos (não necessariamente iguais).
- Quando um processo é trazido para dentro da memória, ele é colocado na menor partição que é suficiente para acomodá-lo.
- Como o tempo, a memória pode ficar fragmentada ao deixar pequenos buracos não contíguos de memória livre.
- Para resolver este problema, usa-se compactação, um processo demorado, em que o S.O. junta todos os processos em um mesmo bloco contíguo, liberando assim espaço livre.



Source W. Stallings

Endereço lógico e endereço físico

- Fica óbvio que, através de *swapping*, um processo não necessariamente vai ser carregado sempre na mesma posição da memória.
- Mais ainda, se a compactação é feita, um processo pode ser deslocado quando já está na memória.
- Num processo há instruções e dados. Instruções podem conter endereços de dados da memória ou endereço de instruções usado em desvios.
- Quando um processo é recarregado na memória, esses endereços podem mudar. Para resolver isso, usam-se endereço lógico e endereço físico.



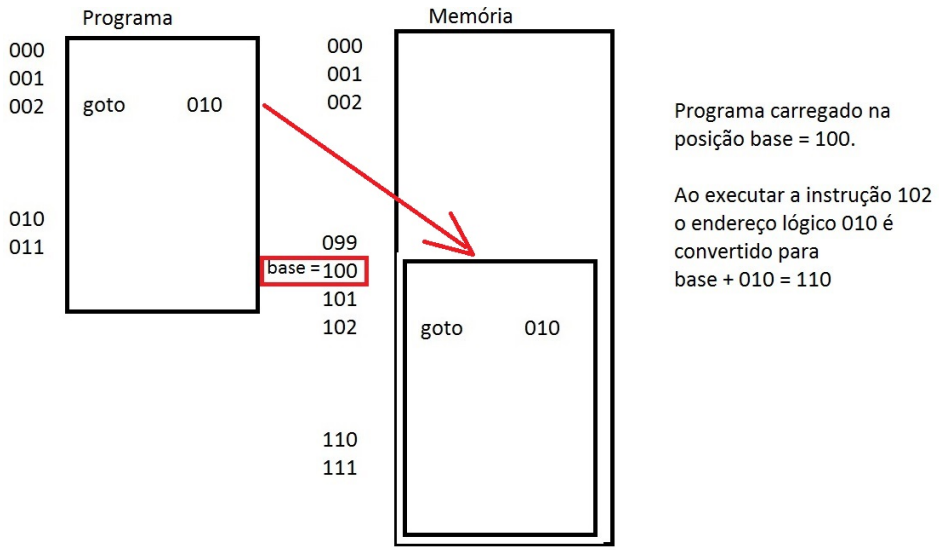
Source W. Stallings

Endereço lógico e endereço físico

- Endereço lógico: Expresso como uma posição relativa ao início do programa. Endereços em um programa contêm somente endereços lógicos.
- Endereço físico: É a localização real na memória.
- Quando um processo é carregado na memória com seu início na posição *base*, então a posição *base* é somada a cada endereço lógico para obter o endereço físico.

  Endereço físico = endereço lógico + base.

- Esse processo é feito por hardware.

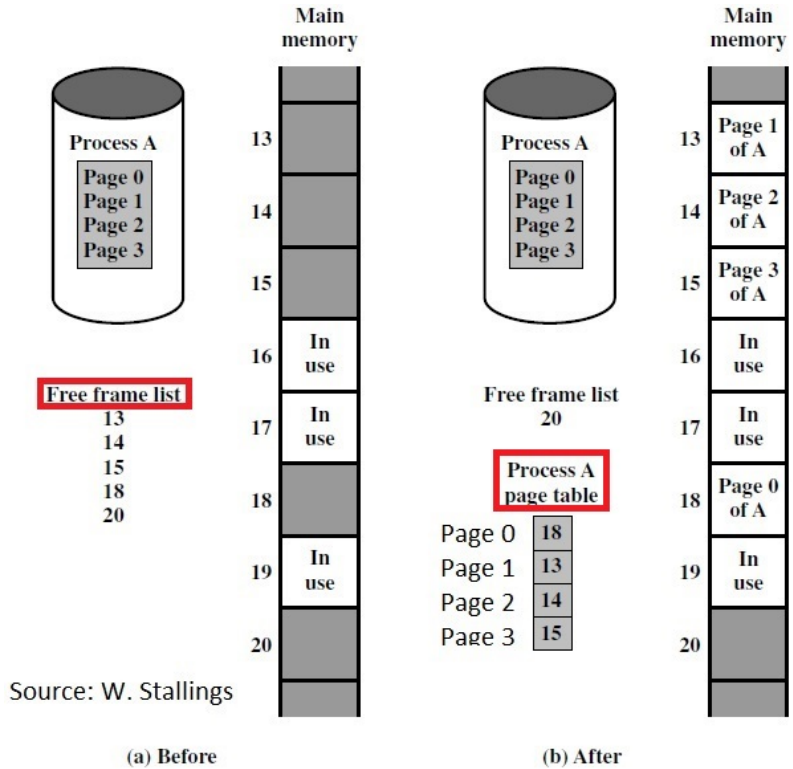


Memória virtual

- Com o aparecimento de programas cada vezes maiores em tamanho e a percepção de que não é necessário carregar todo o programa na memória, aparece o conceito de memória virtual.
  - Um processo pode executar sem ter todas as instruções e dados dentro da memória principal.
  - O espaço de memória disponível ao programa pode exceder o tamanho da memória principal.
- Veremos duas maneiras de implementação de memória virtual.
  - Paginação.
  - Segmentação.

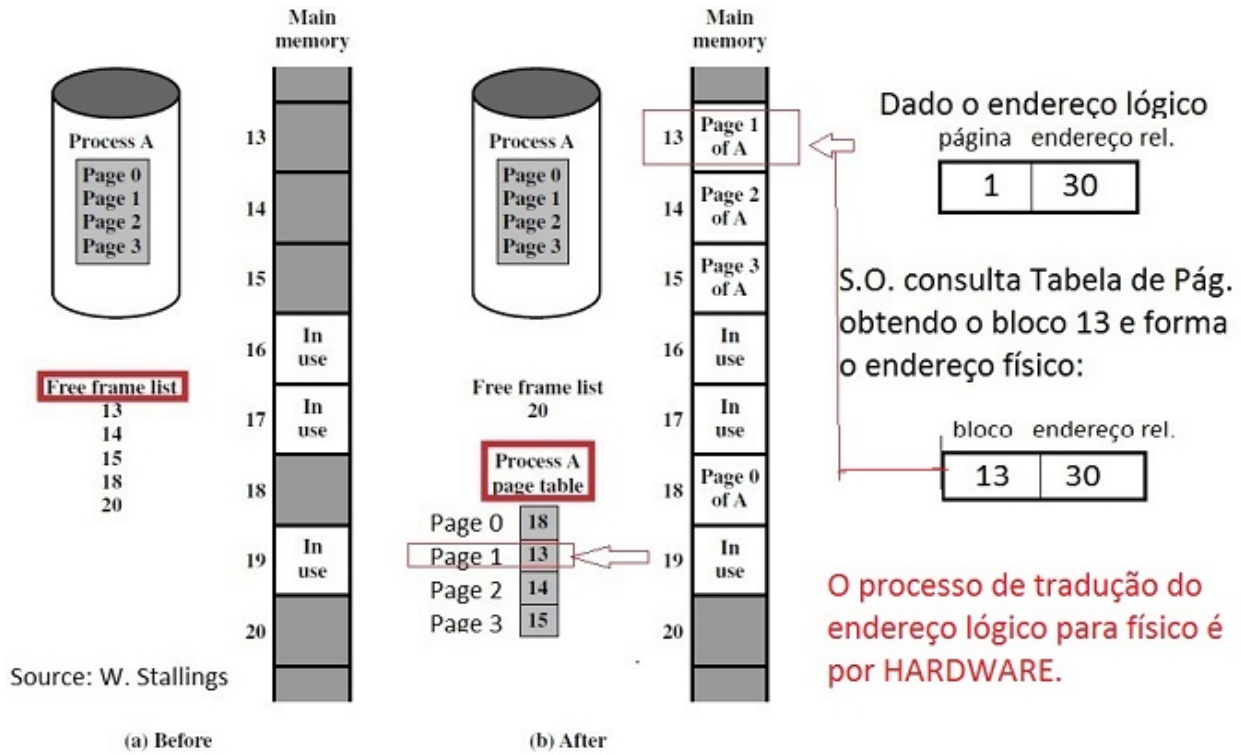
Paginação

- A memória é particionada em blocos ou quadros *frames* de tamanho fixo. Inicialmente todos os blocos estão livres.
- A chamada Tabela de Blocos Livres registra quais blocos estão livres.
- Cada processo é dividido em páginas de igual tamanho que o bloco. Assim, uma página de um processo pode ser carregado em um bloco de memória.
- A chamada Tabela de Páginas (existe uma tabela para cada processo) registra, para dada página, em que bloco está carregado.
- As tabelas são mantidas pelo S.O. Para carrega uma nova página na memória, o S.O. consulta a tablea de blocos livres e aloca a página no bloco selecionado.



Paginação

- O processo utiliza o endereço lógico, que é transformado pelo S.O. para um endereço físico, como se segue.
- Endereço lógico: número da página e o endereço relativo (ou deslocamento) dentro da página.
- Através da Tabela de Páginas, para uma dada página, o S.O. localiza o número do bloco em que ela está carregada.
- Com esse bloco e o endereço relativo, o S.O. forma o endereço físico.
- Endereço físico: número do bloco e o endereço relativo (ou deslocamento) dentro do bloco.



Memória virtual com paginação sob demanda

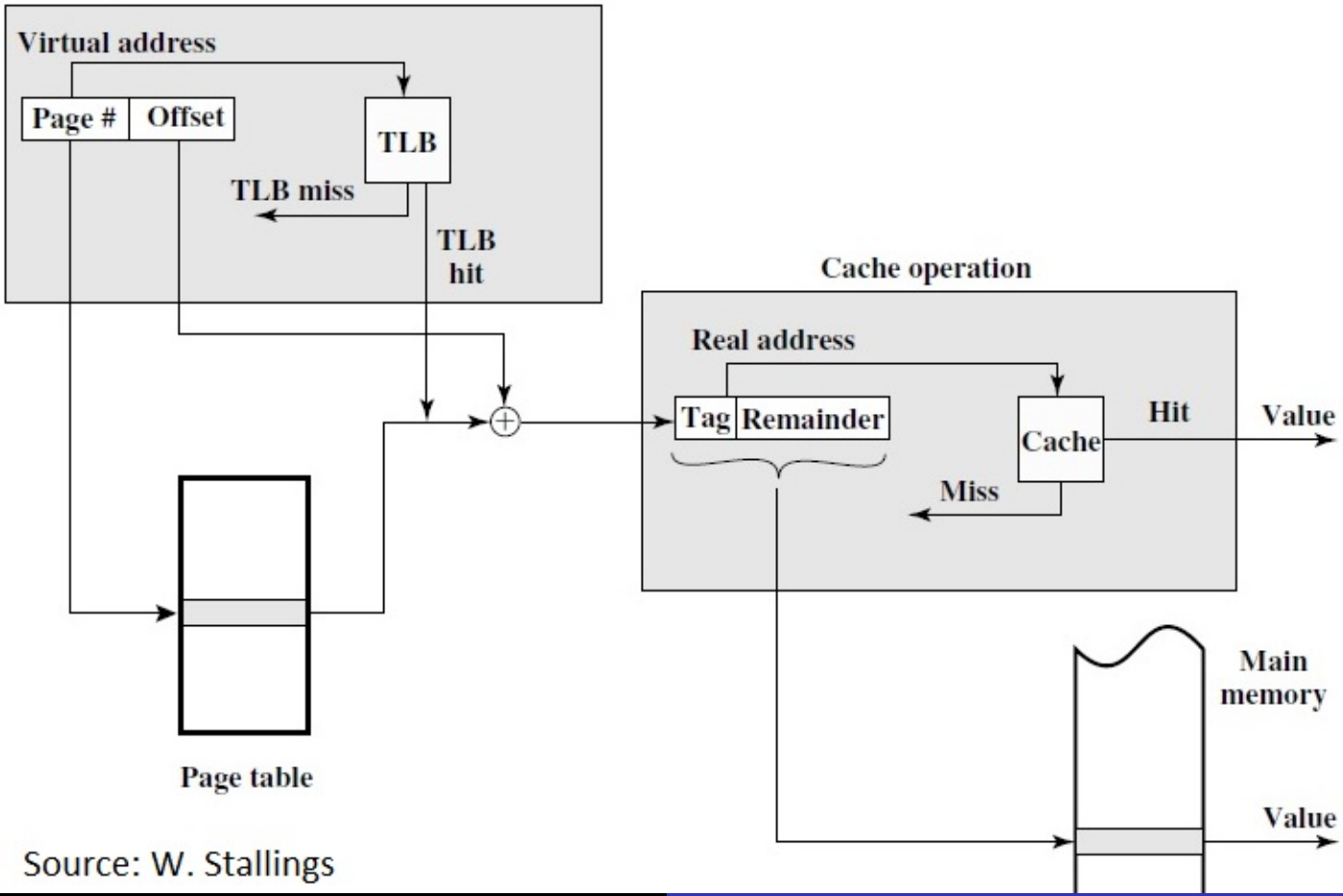
- Mesmo para um programa grande, num período de tempo curto, a execução pode se restringir a apenas um pequeno trecho do programa e talvez a uma ou duas estruturas de dados. É o Princídio de localidade:
  - Localidade temporal: reuso de dados ou instruções dentro de um curto período de tempo.
  - Localidade espacial: reuso de dados relativamente próximos.
- O mecanismo de paginação viabilizou a implementação de memória virtual, em que um programa reside no disco e apenas páginas necessárias são trazidas à memória, sob demanda, é a chamada Paginação sob Demanda.
- Se a execução precisa uma página que não está na memória, o S.O. será acionado através de uma interrupção de falha de página (ou *page fault*) a fim de trazer a página para a memória.
- Para isso, um bloco livre é usado para receber a página. Se não há blocos livres, então o S.O. seleciona e desocupa uma página na memória, dando o bloco liberado para a nova página. Algum algoritmo de substituição de página é usado, e.g. LRU.

Implementação da Tabela de Páginas

- Cada processo tem uma Tabela de Páginas, onde há uma entrada para cada página do processo.
- Com a memória virtual, um processo pode consistir de um grande número de páginas, impossibilitando alocar a Tabela de Páginas dentro da memória física.
- As Tabelas de Páginas desses processos são implementadas em memória virtual (disco): Apenas parte de cada tabela está armazenada em blocos de memória física.
- Isso cria uma ineficiência: cada referência à memória virtual pode acarretar em dois acessos à memória física: um para acessar entrada desejada da Tabela de Páginas, e outro para a página desejada.
- Para evitar esse problema, é usada uma memória cache especial *Translation Lookaside Buffer (TLB)*, para entradas da Tabela de Páginas.

Translation Lookaside Buffer (TLB)

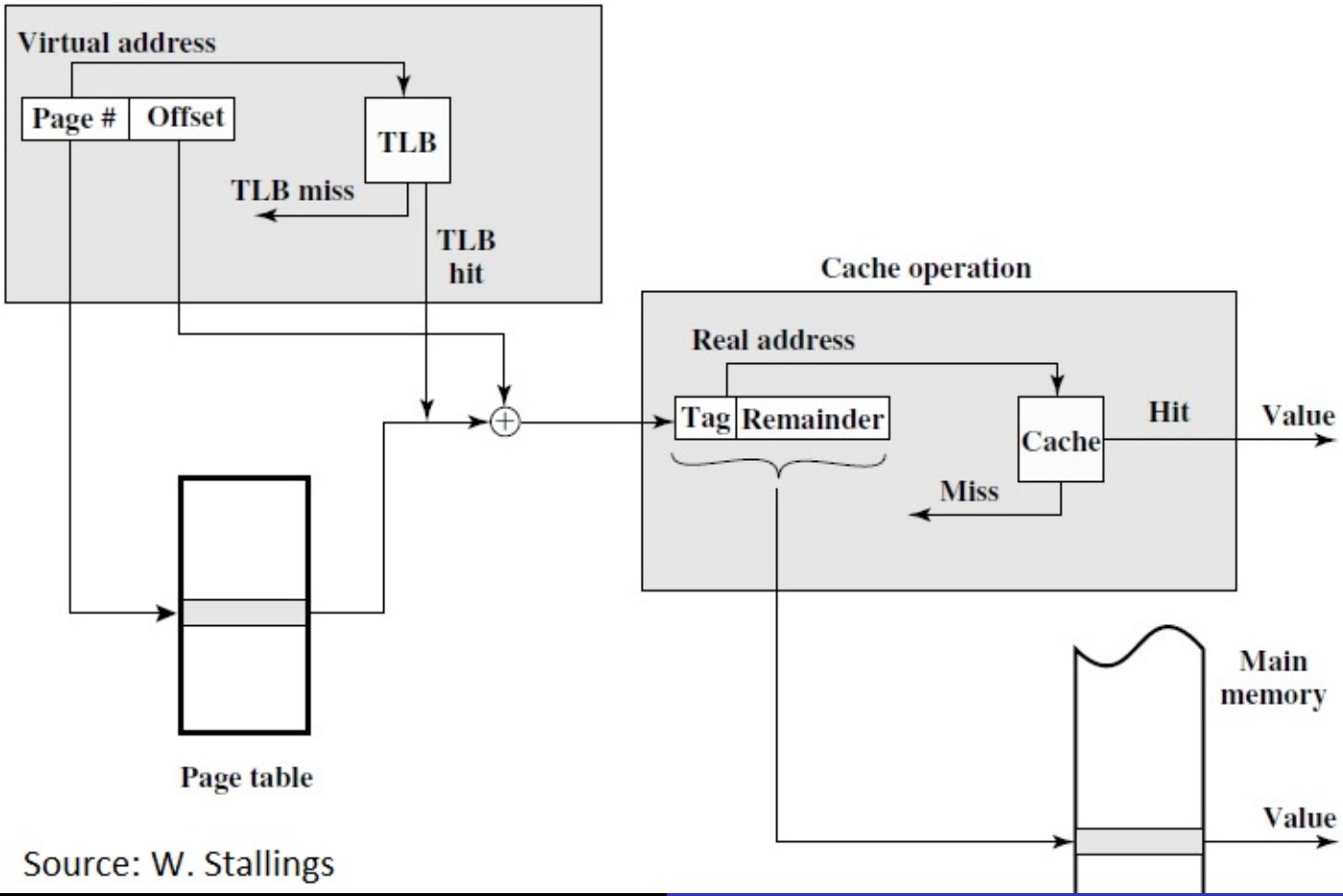
- O *Translation Lookaside Buffer* TLB funciona como uma memória cache e contém as entradas (i.e. as linhas) da Tabela de Páginas mais recentemente usadas.
- Para localizar uma dada página, TLB é consultado. Se encontrar (*TLB hit*), o número do bloco correspondente é obtido. Pelo princípio de localidade, há grande probabilidade de isso ocorrer.



Source: W. Stallings

Translation Lookaside Buffer (TLB)

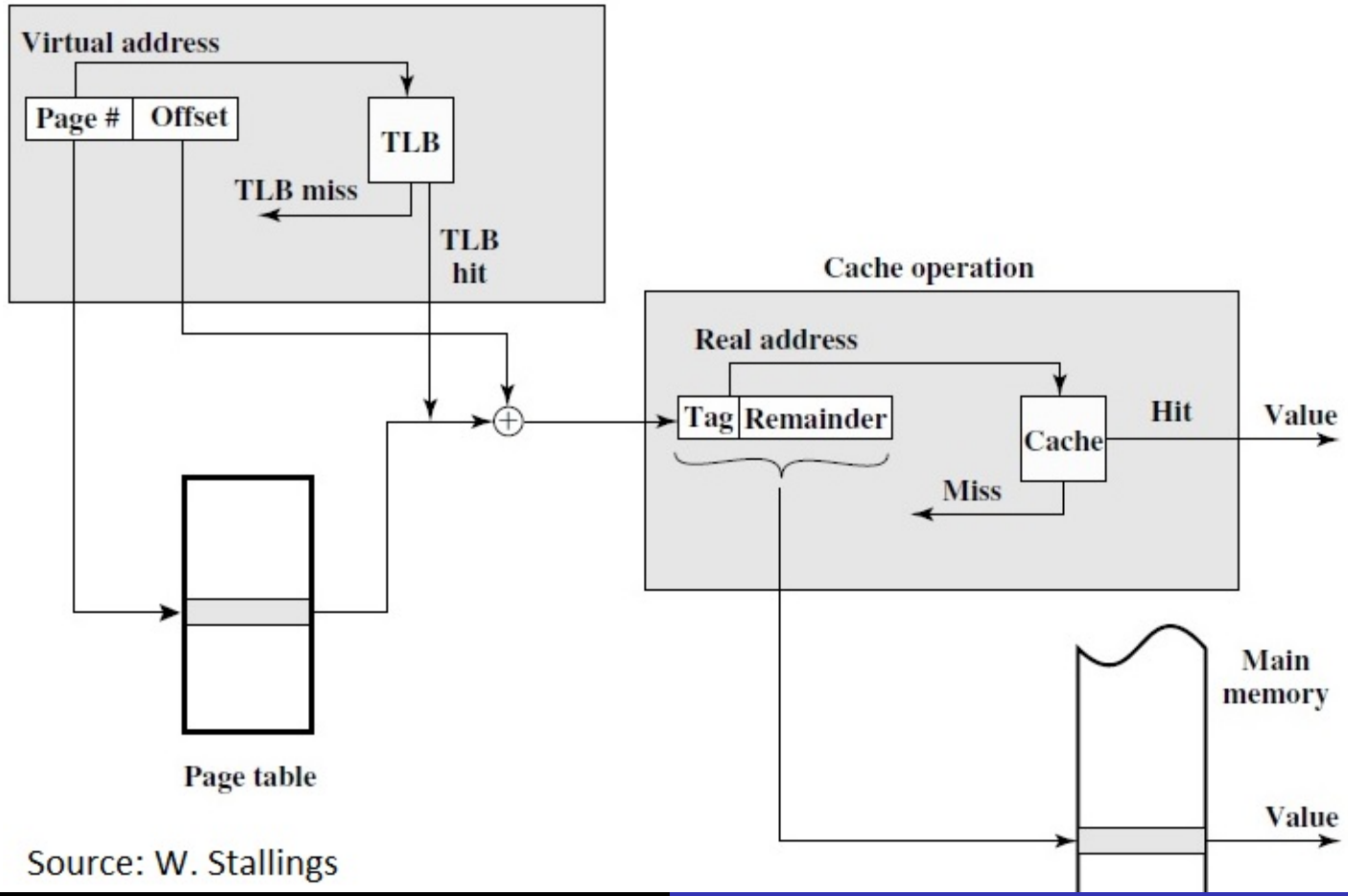
- Se a parte da Tabela de Páginas presente no TLB não contém a página desejada (*TLB miss*), então passa-se a acessar a parte da Tabela de Páginas na memória.
- Se não encontrar na memória, então ocorre uma interrupção *page fault* e é trazida a entrada desejada da Tabela de Páginas do disco, atualizando a parte de Tabela de Páginas na memória física.



Source: W. Stallings

Translation Lookaside Buffer (TLB)

- O endereço físico obtido (bloco + endereço relativo) correpsondente ao endereço lógico é então buscado na memória cache normal. No caso de *cache miss*, a memória física é acessada.
- Veja a complexidade envolvida em uma simples referência à memória: 1) uma referência à Tabela de Páginas, que pode estar no TLB, na memória, ou disco, e 2) o endereço referenciado pode estar na cache, memória ou disco.



Source: W. Stallings

Segmentação

- Segmentação é uma outra maneira de subdividir a memória. Difere da paginação nos aspectos:
  - Paginação: invisível ao programador, serve para prover um espaço maior de endereçamento. Memória dividida em páginas de igual tamanho, com qualquer conteúdo.
  - Segmentação: em geral visível ao programador, serve para organizar programas de dados, associando atributos de privilégio e proteção a instruções e dados. Memória dividida em segmentos de programas e segmentos de dados de tamanhos variados e dinâmicos.
- Pode haver vários segmentos de programas e de dados, com diferentes direitos de acesso atribuídos a cada um.
- Referência ou endereço à memória consiste de (número de segmento, endereço relativo ou deslocamento).

Segmentação

Vantagens da segmentação:

- Simplifica o crescimento no tamanho de estruturas de dados. O S.O. pode aumentar ou diminuir o tamanho de um segmento contendo uma estrutura de dado.
- Em caso de alteração de alguns segmentos, facilita a recompilação sem ter que religar e recarregar todo o programa.
- Facilita o compartilhamento entre processos: um usuário pode colocar um utilitário ou uma tabela útil em um segmento que pode ser endereçado por outros usuários.
- Facilita a proteção: privilégios de acesso podem ser atribuídos de maneira conveniente.

Segmentação combinada com paginação:

- Por outro lado, a paginação tem a vantagem de prover uma forma eficiente de gerenciamento de memória.
- A fim de combinar as vantagens de ambas, alguns sistemas equipam o hardware e S.O. para prover segmentação e paginação.

Gerenciamento de memória do Intel Pentium II

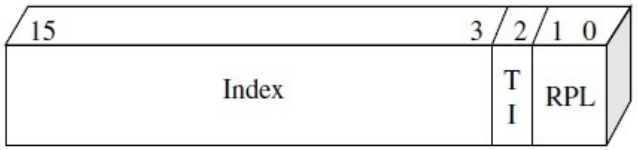
- O Intel Pentium II possui hardware para segmentação e paginação. Ambos os mecanismos podem ser desativados, permitindo ao usuário a escolha de uma das 4 formas de visualizar a memória:
  - Memória sem segmentação e sem paginação: o endereço virtual é igual ao endereço físico, útil para aplicações de controle de baixa complexidade e alto desempenho.
  - Memória sem segmentação e com paginação: usado no S.O. Berkeley Unix.
  - Memória com segmentação e sem paginação: tem a vantagem de ter a Tabela de Segmentos *on chip* quando o segmento está na memória.
  - Memória com segmentação e com paginação: segmentação usada para definir partições lógicas de memória e paginação usada para gerenciar a alocação de memória dentro da partição. Usado em Unix System V.

Gerenciamento de memória do Intel Pentium II

- Um endereço virtual ou lógico consiste de:
  - Um seletor de segmento de 16 bits (dos quais 2 bits são para o mecanismo de proteção, sobrando 14 bits para especificar o segmento).
  - Um deslocamento (*offset*) de 32 bits.
- Qual o espaço total da memória virtual?
  - O tamanho de um segmento é no máximo $2^{32} = 4$ Gbytes.
  - Com segmentação, o espaço total da memória virtual pode ser $2^{14+32} = 2^{46} = 64$ TBytes.
- Dois bits especificam o nível de privilégio de cada segmento, que determina qual segmento de programa pode acessar qual segmento de dado:
  - 4 níveis de privilégio: Do nível 0 (mais protegido) ao nível 3 (menos protegido).
  - Um segmento de progama de nível de privilégio i pode acessar um segmento de nível de privilégio j se $i \leq j$.

Trdução de endereços - do virtual para o linear

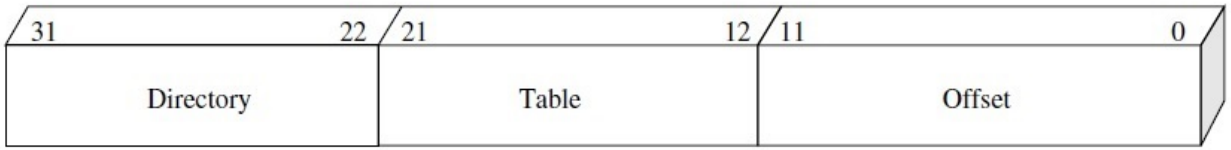
- Segmentação pode estar ativada ou desativada.
  - Quando ativada: o endereço usado no programa é um end. virtual = um seletor de segmento (16 bits) + um *offset* de 32 bits. Esse end. virtual é primeiro transformado em um end. linear.
  - Quando desativada: endereço linear é usado em programas.



+ Offset de 32 bits

TI = Table indicator
RPL = Requestor privilege level

(a) Segment selector + Offset = Endereço Virtual

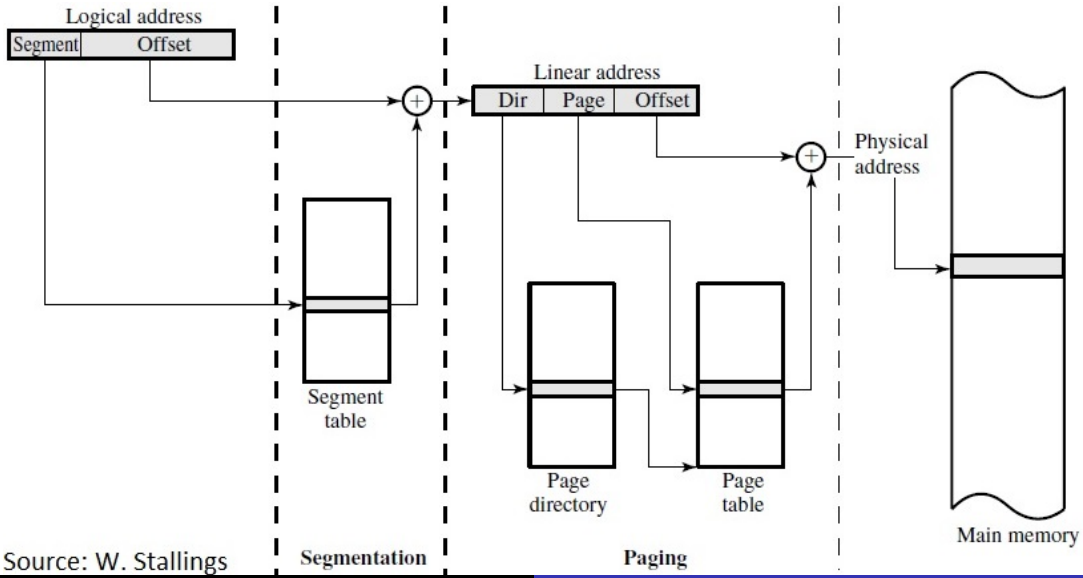


(b) Linear address

Source: W. Stallings

Pentium - tradução de endereço de memória

- Para converter o endereço linear ao endereço final de 32 bits:
  - Os primeiros 10 bits do end. linear (*Dir*) acessa a chamada *Page Directory* que reside na memória principal, levando a uma das 1.024 grupos de páginas, cada um com a sua *Page Table*.
  - Cada *Page Table*, de 4 Mbytes de tamanho, contém 1.024 entradas para páginas 4 Kbytes cada.
  - Os próximos 10 bits do end. linear (*Page*) são usados para acessar a *Page Table*, cujo valor somado ao *offset* dá o endereço final.
  - As *Page Tables* podem residir em memória virtual. Uma *TLB* ou *Translation Lookaside Buffer* (cache) armazena 32 entradas a *Page Tables*.
  - Ao invés de páginas de 4 Kbytes, pode-se usar páginas de 4 Mbytes = tamanho da *Page Table*.



Como foi o meu aprendizado?

- Sabemos da importância do apoio de hardware no bom desempenho do S.O. Cite exemplos desses apoios de hardware.

- Conversão do endereço lógico para endereço físico quando o processo é carregado na memória. (Ver slide 9.)
- *TLB* ou *Translation Lookaside Buffer*: uma memória cache para entradas da tabela de páginas. (Ver slides 14 a 17.)
- Mecanismo de conversão do endereço virtual para o endereço final, com segmentação e paginação. (Ver slides 22 e 23.)

Como foi o meu aprendizado?

- Sabemos da importância do apoio de hardware no bom desempenho do S.O. Cite exemplos desses apoios de hardware.

- Conversão do endereço lógico para endereço físico quando o processo é carregado na memória. (Ver slide 9.)
- *TLB* ou *Translation Lookaside Buffer*: uma memória cache para entradas da tabela de páginas. (Ver slides 14 a 17.)
- Mecanismo de conversão do endereço virtual para o endereço final, com segmentação e paginação. (Ver slides 22 e 23.)