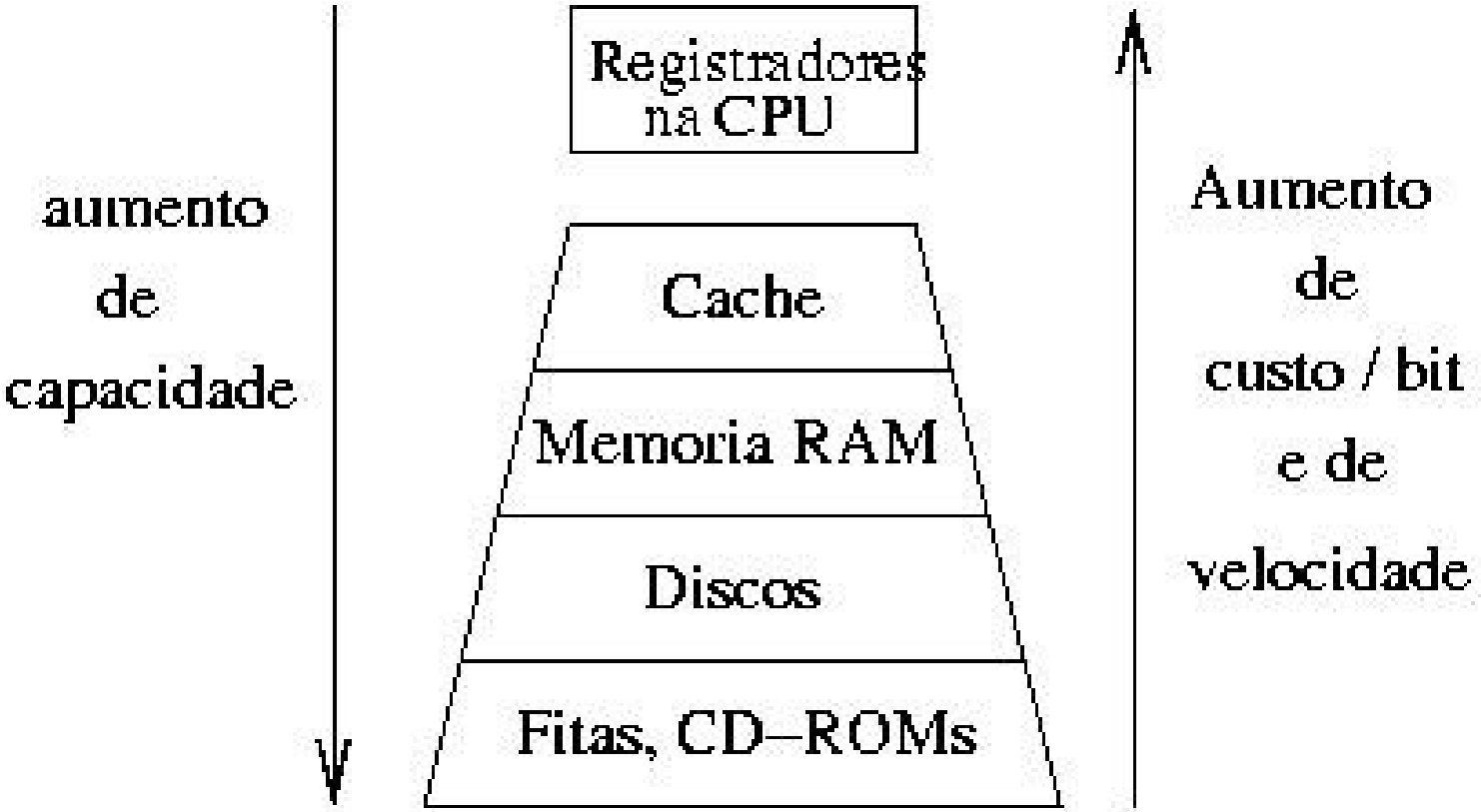
Gerenciamento de memória

O que faz?

- Controla o uso dos espaços em memória
- Controla os modos de endereçamento dos processos
- Como funciona a memória?

Hierarquia de memória



A falta de desempenho

- A velocidade de memória dobra a cada sete anos
- A velocidade do processador dobra a cada 18 meses
- Isso exige melhor gerenciamento de memória

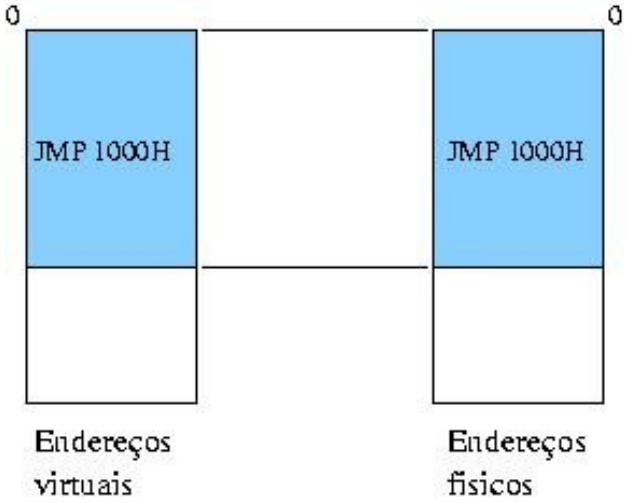
	XT (1980)	PII (1999)	P4 (2005)
processador	210ns	3ns	0,3ns
memória	200ns	50ns	30ns

Tecnologias de construção

- Dinâmica → DRAM
  - memórias capacitivas, necessitando refresh. Tempo efetivo de acesso da ordem de 60ns
- Estática → SRAM
  - usam portas lógicas, dispensando refresh. Tempo de acesso da ordem de 7ns

Endereçamento

- A função de endereçamento diz respeito ao modo em que os **endereços lógicos** (existentes no código armazenado em disco) são transformados em **endereços físicos** (endereços na memória de fato)

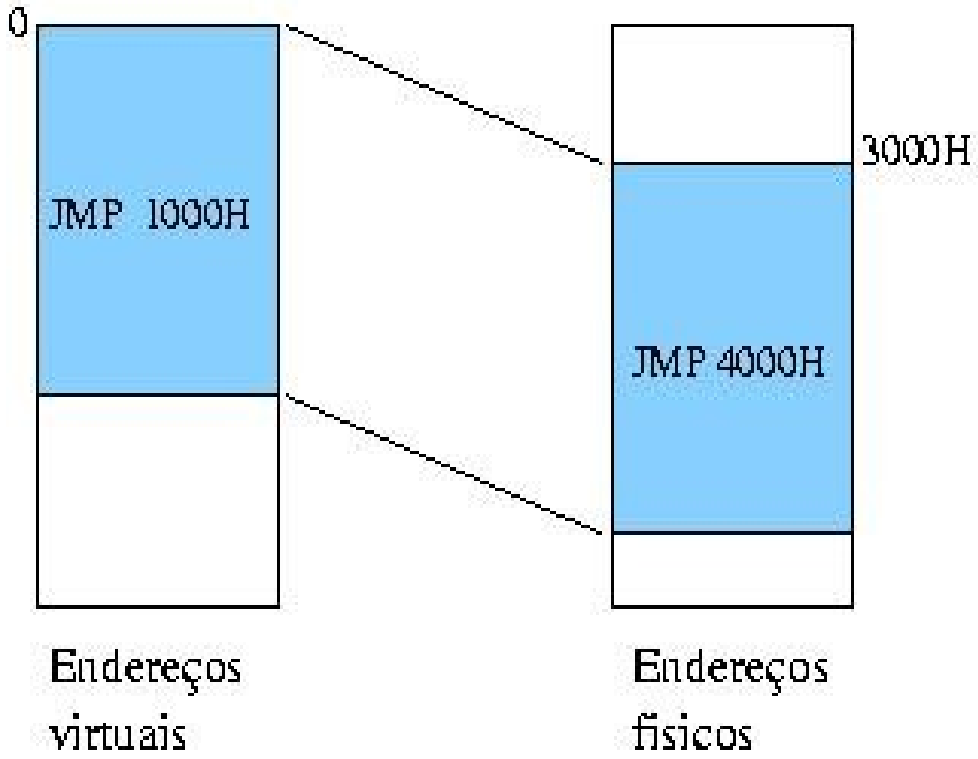


Formas de endereçamento

- Relocação estática
  - endereços são convertidos durante o carregamento do programa
- Relocação dinâmica
  - endereços são convertidos apenas durante a execução do programa
- Segmentação
  - faz relocação dinâmica separando segmentos estruturais dos programas

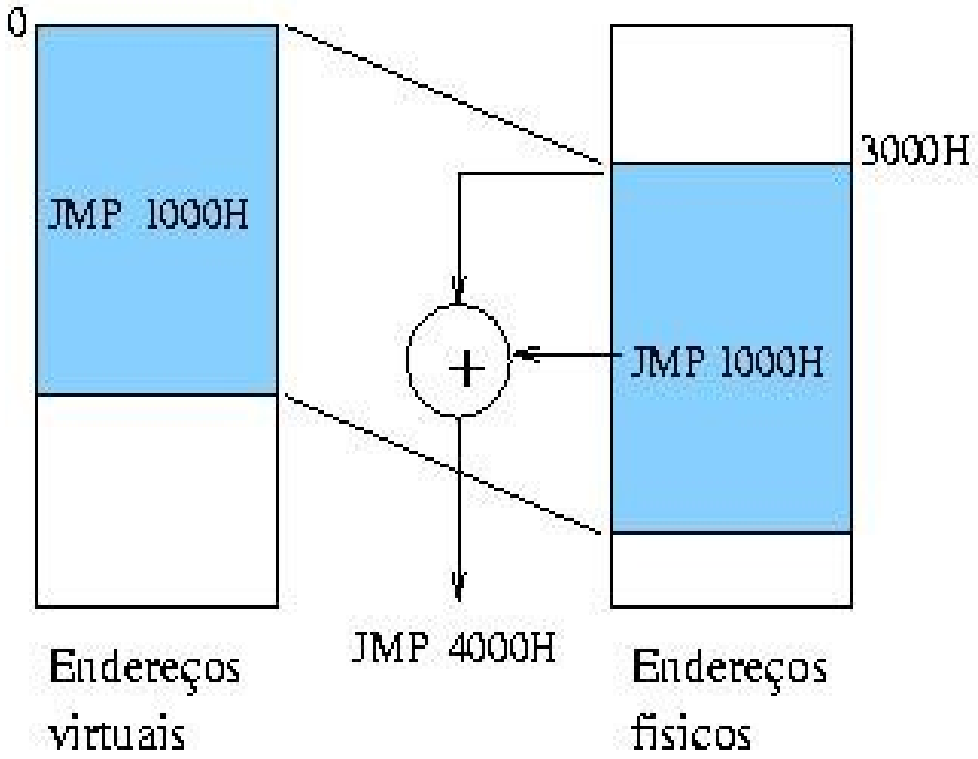
Relocação estática

- Endereços físicos são calculados no momento do carregamento



Relocação dinâmica

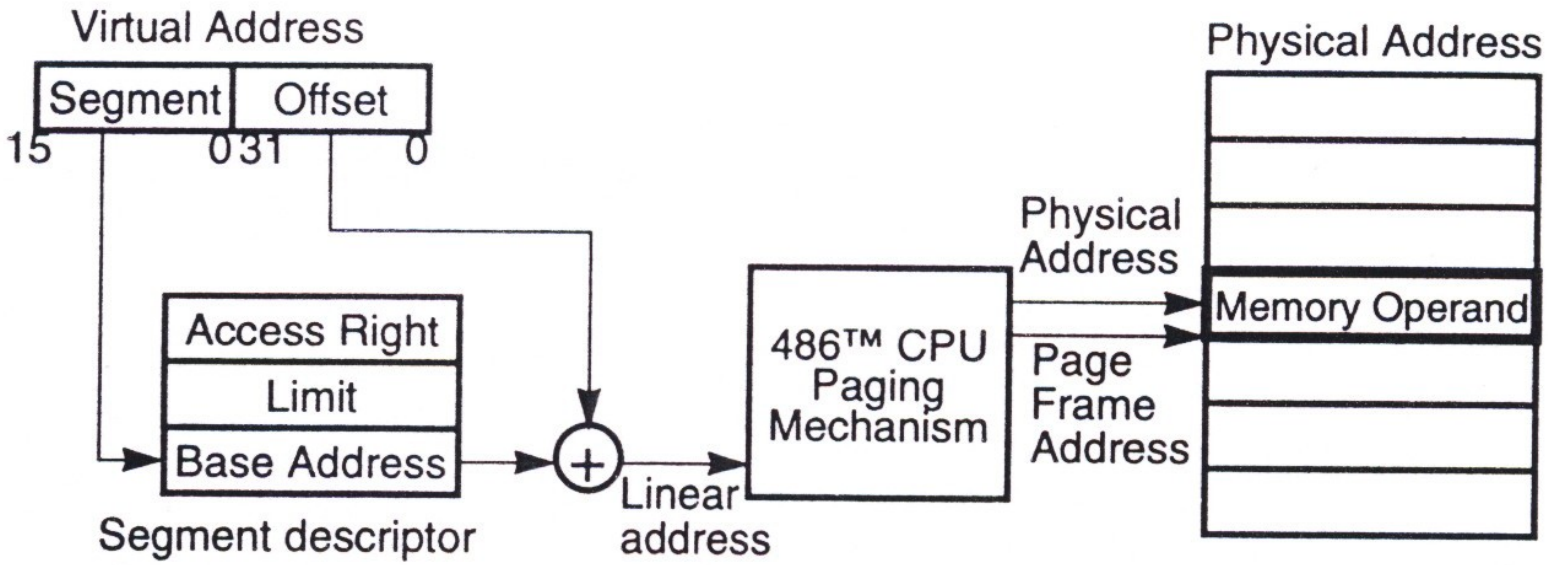
- Endereços físicos são calculados apenas em tempo de execução



Hardware para endereçamento

- Uso de TLB (Translation Lookahead Buffer) ou MMU (Memory Management Unit)
- Faz a conversão de endereços virtuais (vindos da relocação dinâmica) em endereços físicos (com a localização real na memória)

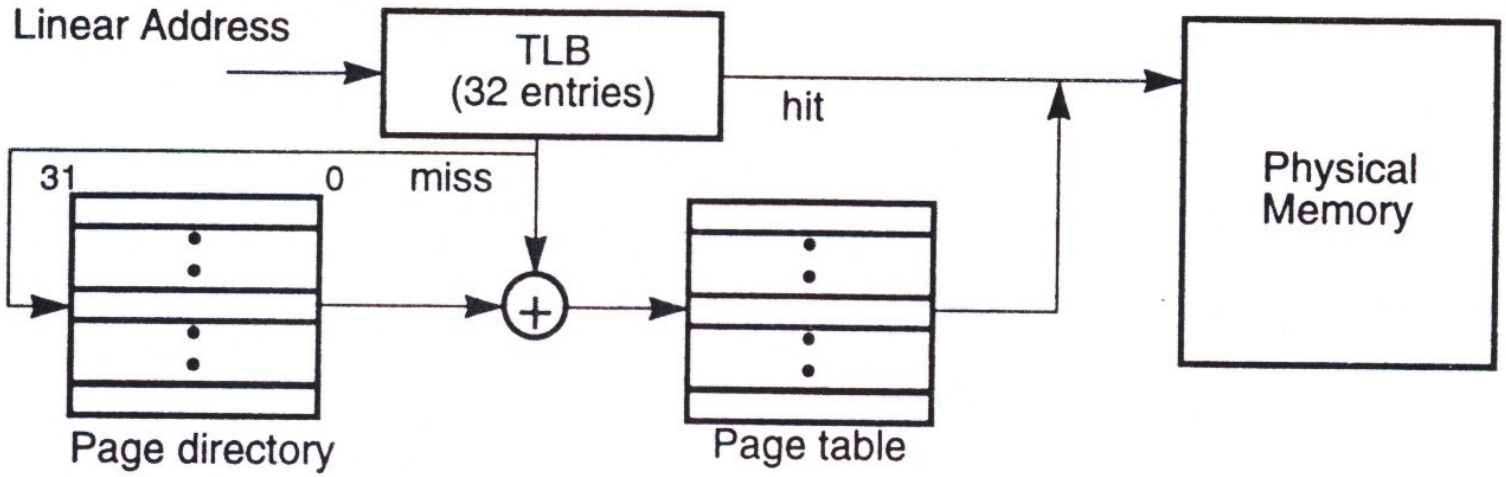
Mecanismos do i486



(a) Segmentation to produce the linear address

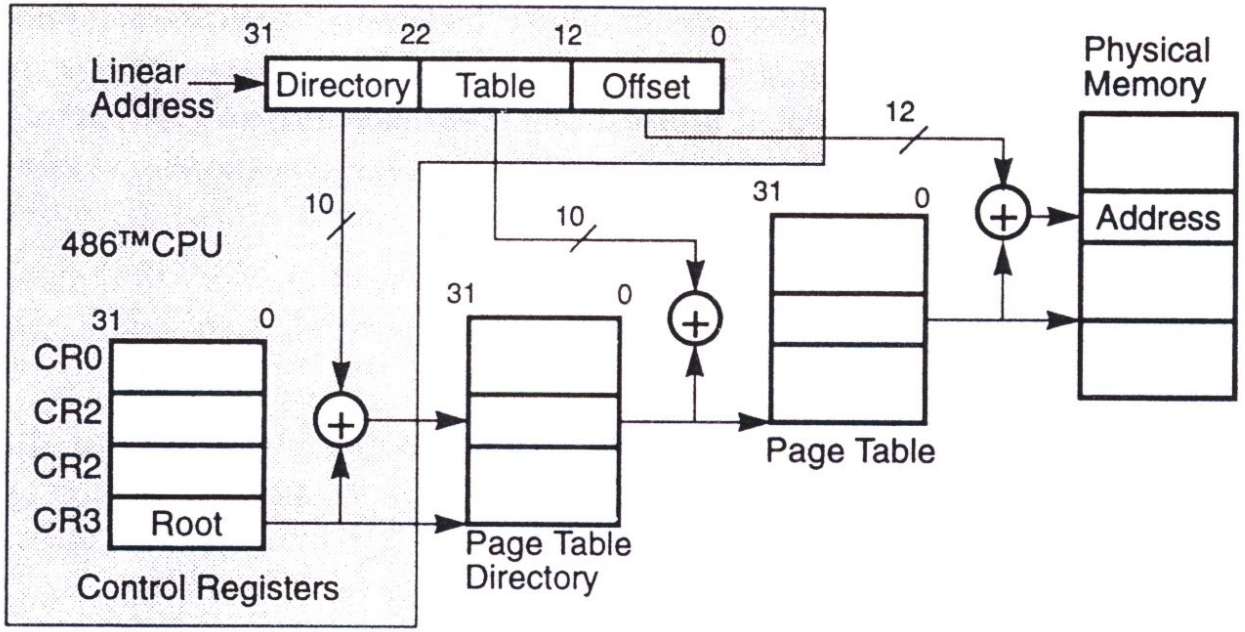
Mecanismos do i486

- Operação da TLB



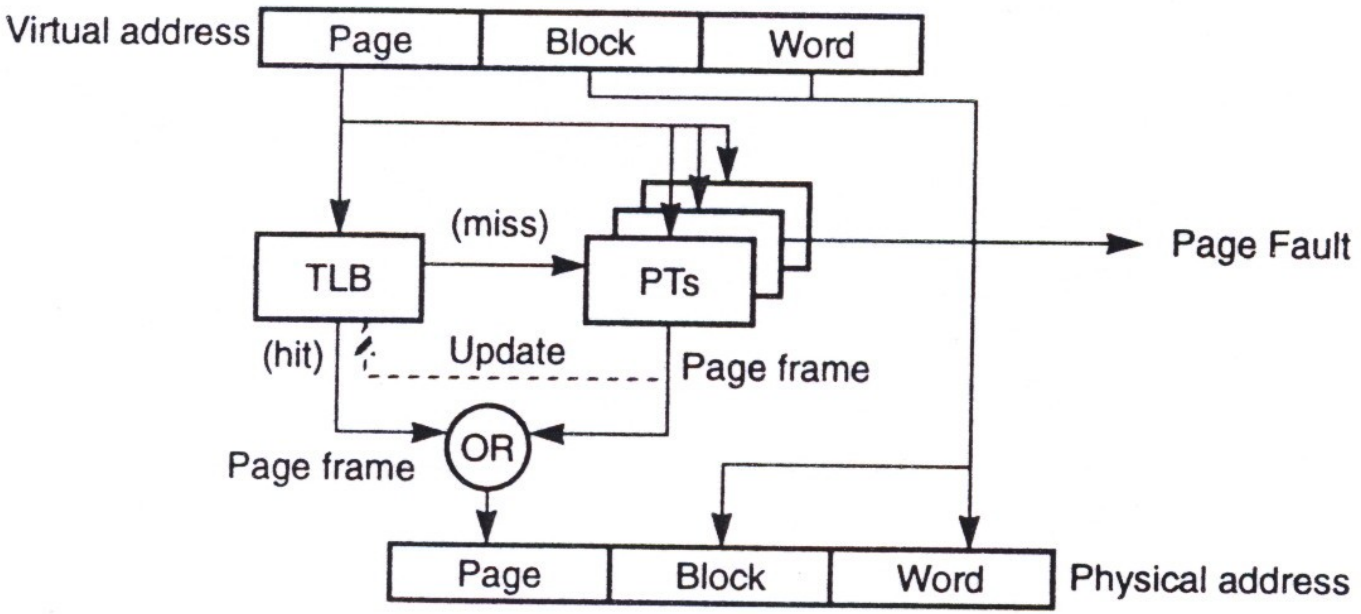
Mecanismos do i486

- Esquema de paginação



(c) A two-level paging scheme

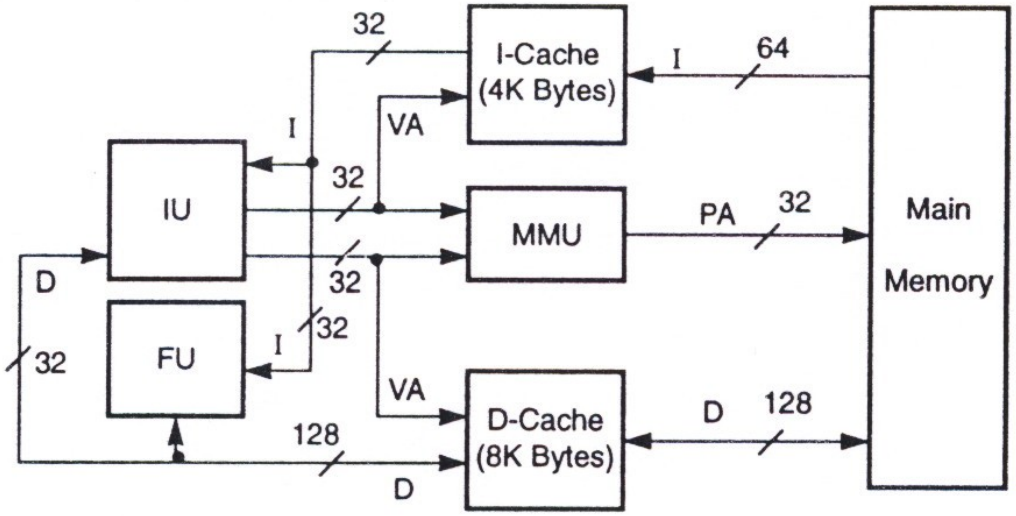
Esquema geral da TLB



(b) Use of a TLB and PTs for address translation

Exemplo de implementação

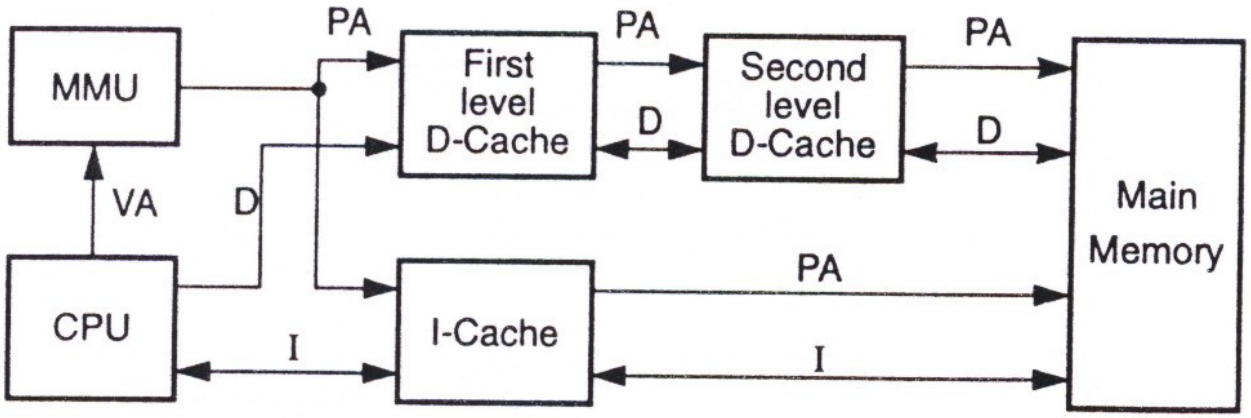
- Modelo de endereço virtual (split cache)



(b) A split cache accessed by virtual address as in the Intel i860 processor

Exemplo de implementação

- Modelo de endereço físico (split cache)



(b) Split caches accessed by physical address in the Silicon Graphics workstation

Alocação de Memória

- Controla o espaço físico de memória que cada processo irá usar

- Usa dois modelos básicos de alocação:
  - Em espaços contíguos
  - Em blocos

Alocação em espaços contíguos

- Todos os bytes do segmento são armazenados seqüencialmente
- Problemas com mapeamento dos espaços vazios
- Necessidade de constantes recompactações para corrigir o problema de fragmentação externa (existência de espaço disponível em posições não seqüenciais)
- Soluções como uso de espaços de tamanho fixo, sistemas Buddy e outras não corrigem esses problemas, criando ainda a fragmentação interna

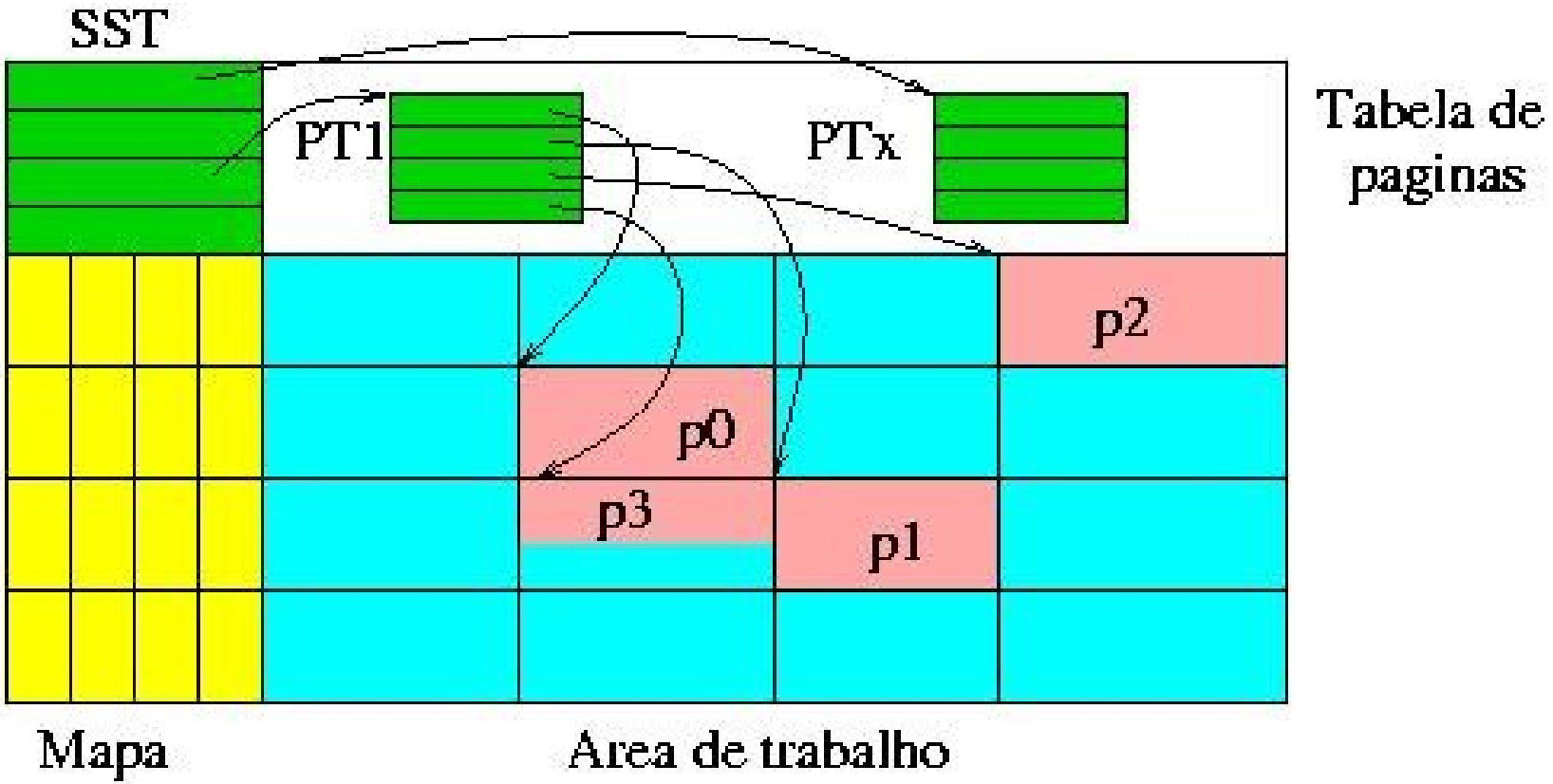
Alocação em espaços contíguos

- Disso resulta que:
  - Apresenta problemas relacionados ao uso da memória (grande desperdício por fragmentação externa)
  - Usada apenas para sistemas dedicados

Alocação em blocos

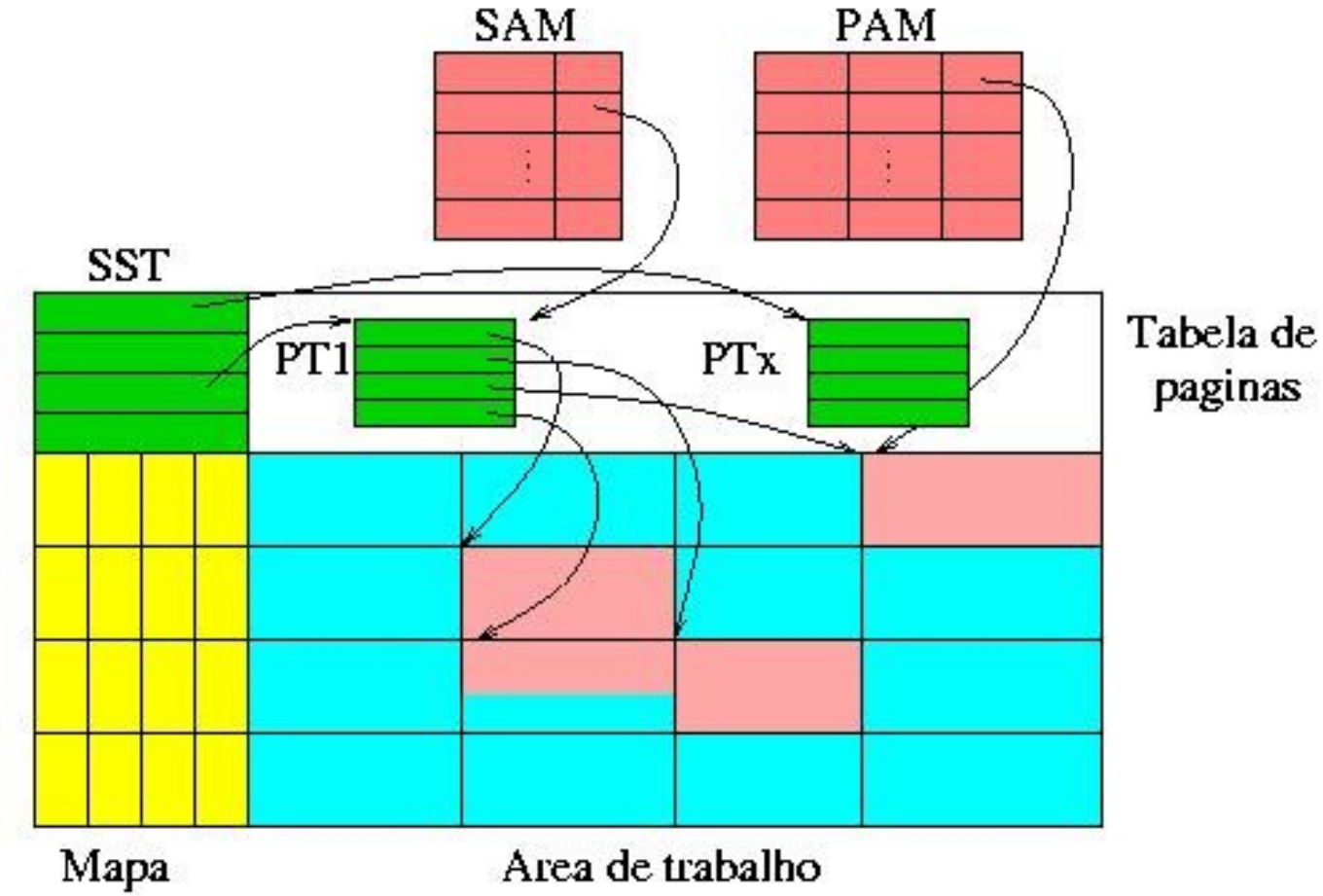
- Os bytes dos segmentos são armazenados em páginas de tamanho fixo, não necessariamente seqüenciais
- Elimina a fragmentação externa, porém mantém a fragmentação interna
- Volume do desperdício com fragmentação interna depende do tamanho do bloco (ou página)
- Esse tamanho é determinado pelo compromisso entre complexidade de gerenciamento e redução na fragmentação

Memória em blocos



Memória em blocos (com cache)

- Uso de memórias associativas



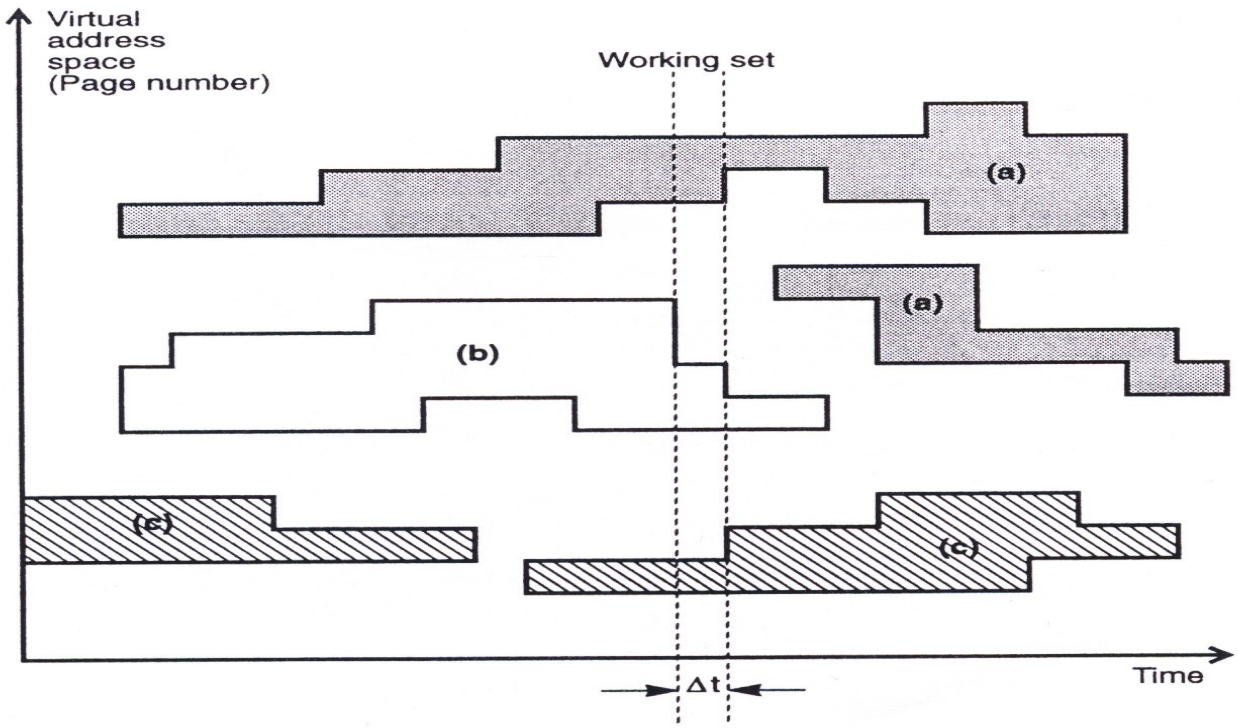
Memória virtual

- Elimina o limite de tamanho da memória para a execução de programas
- Trabalha com o conceito de paginação por demanda (blocos ou páginas vão para a RAM apenas se requisitadas)
- Usa algoritmos de retirada de páginas a partir do Princípio da Localidade

Princípio da Localidade

- Diz que existem endereços mais prováveis de serem acessados do que outros.
- Na prática, se um processo executa uma instrução da página X, com dados da página Y, então a próxima instrução a ser executada deverá também estar na página X e acessar dados da Y

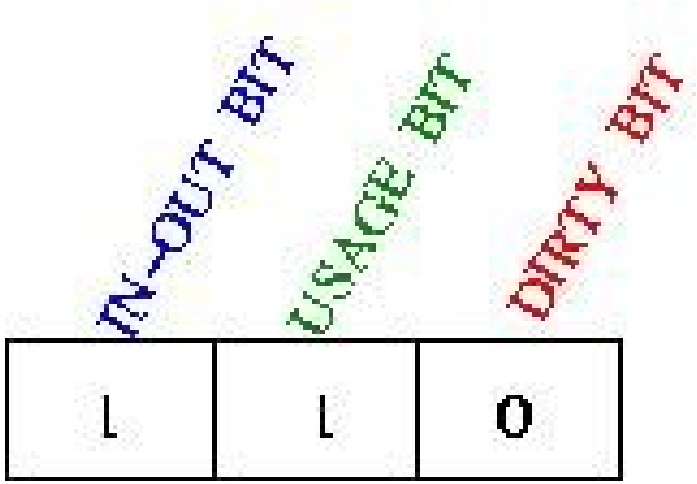
Localidade de referências



Parâmetros importantes

- Seqüência de referência
- Conjunto residente
- Tamanho do conjunto residente
- Faltas de página
- Swap-in e Swap-out
- Bits de controle de uso

Bits de controle de uso



- IN-OUT BIT: bloco está na memória (1) ou no disco (0)
- USAGE BIT: bloco usado recentemente (1)
- DIRTY BIT: bloco foi modificado na memória (1)

O átomo

Algoritmos de paginação

- A eficiência da MV depende do algoritmo de retirada de páginas
- Exemplos de algoritmos:
  - FIFO
  - LRU
  - opt
  - Segunda chance

FIFO

- Retira a página que entrou primeiro na memória
- Funcionamento e implementação simples
- Pode apresentar piora de desempenho ao aumentar-se o tamanho do conjunto residente (anomalia de Belady)

Anomalia de Belady

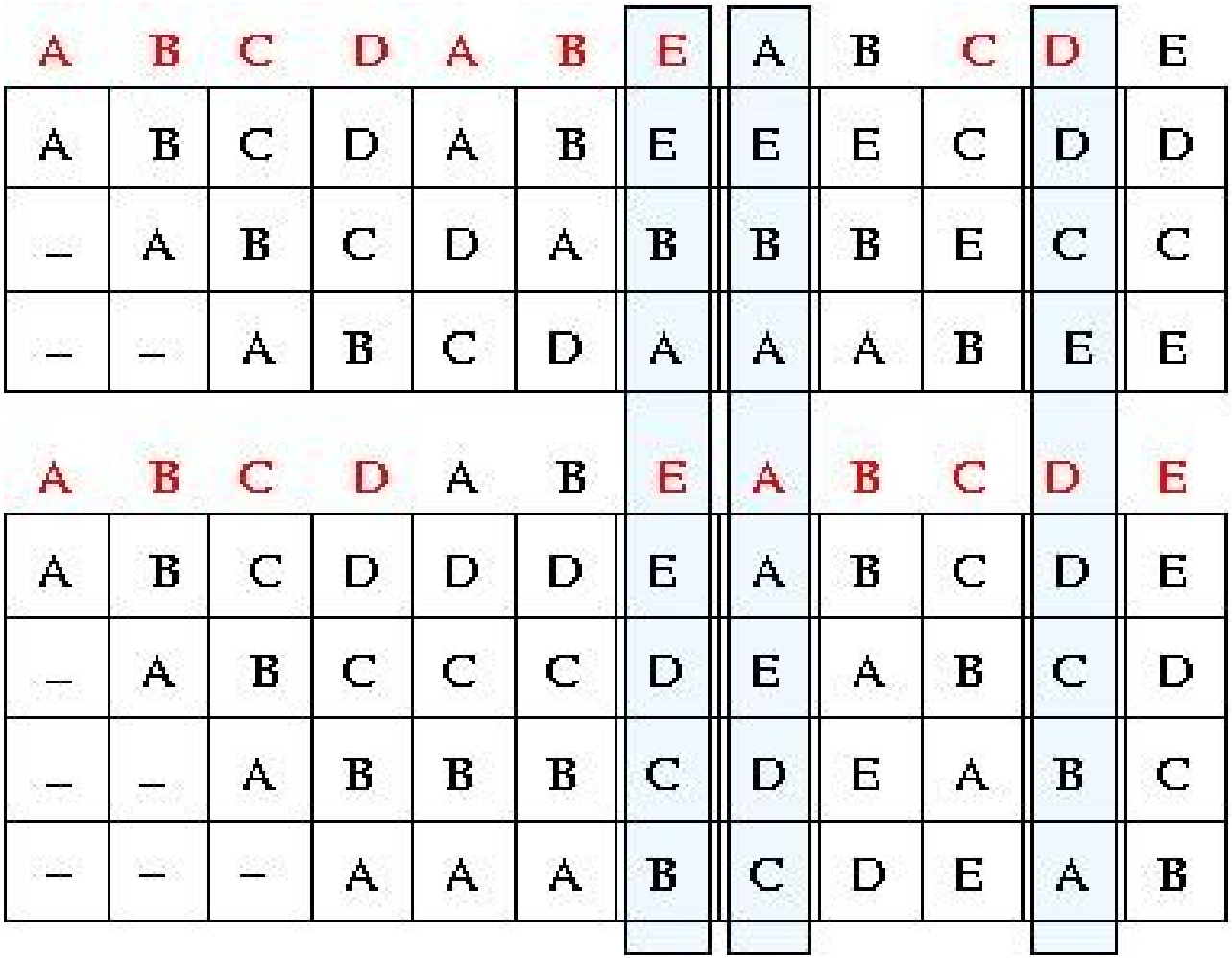
A	B	C	D	A	B	E	A	B	C	D	E
A	B	C	D	A	B	E	E	E	C	D	D
_	A	B	C	D	A	B	B	B	E	C	C
_	_	A	B	C	D	A	A	A	B	E	E

Anomalia de Belady

A	B	C	D	A	B	E	A	B	C	D	E
A	B	C	D	A	B	E	E	E	C	D	D
_	A	B	C	D	A	B	B	B	E	C	C
_	_	A	B	C	D	A	A	A	B	E	E

A	B	C	D	A	B	E	A	B	C	D	E
A	B	C	D	D	D	E	A	B	C	D	E
_	A	B	C	C	C	D	E	A	B	C	D
_	_	A	B	B	B	C	D	E	A	B	C
_	_	_	A	A	A	B	C	D	E	A	B

Anomalia de Belady



A	B	C	D	A	B	E	A	B	C	D	E
A	B	C	D	A	B	E	E	E	C	D	D
–	A	B	C	D	A	B	B	B	E	C	C
–	–	A	B	C	D	A	A	A	B	E	E

A	B	C	D	A	B	E	A	B	C	D	E
A	B	C	D	D	D	E	A	B	C	D	E
–	A	B	C	C	C	D	E	A	B	C	D
–	–	A	B	B	B	C	D	E	A	B	C
–	–	–	A	A	A	B	C	D	E	A	B

LRU (Least Recently Used)

- Corrige o problema da anomalia de Belady através do conceito de pilha
- Retira sempre a página que está no topo da pilha, que é a página usada há mais tempo
- Assim, o aumento no tamanho do conjunto residente implica apenas no aumento do tamanho da pilha

OPT (optimal)

- Também trabalha com o conceito de pilha, retirando a página que está no seu topo
- Aqui, porém, a página que fica no topo é aquela que demorará mais para ser necessária
- Tem uso apenas teórico, como referencial para comparação de algoritmos

Segunda Chance

- E uma implementação prática (simplificada) do conceito de pilha
- Baseia-se numa fila de páginas, sendo que a página que sairá da memória deve ser uma que tenha pouco uso e não tenha sido alterada (preferencialmente)

Comparação de algoritmos

	PF	0	1	2	4	2	3	7	2	1	3	1	Hit Ratio
LRU	*a*	0	0	0	4	4	4	7	7	7	3	3	$\frac{3}{11}$
	b		1	1	1	1	3	3	3	1	1	1	
	c			2	2	2	2	2	2	2	2	2	
	Faults	*	*	*	*		*	*		*	*		
OPT	*a*	0	0	0	4	4	3	7	7	7	3	3	$\frac{4}{11}$
	b		1	1	1	1	1	1	1	1	1	1	
	c			2	2	2	2	2	2	2	2	2	
	Faults	*	*	*	*		*	*			*		
FIFO	*a*	0	0	0	4	4	4	4	2	2	2	2	$\frac{2}{11}$
	b		1	1	1	1	3	3	3	1	1	1	
	c			2	2	2	2	7	7	7	3	3	
	Faults	*	*	*	*		*	*	*	*	*		

Implicações de desempenho

- Taxa de faltas de página
- Nível de multiprogramação
- Thrashing
- Coerência de cache

Taxa de faltas de página

- O uso de memória virtual implica em maior tempo de processamento na ocorrência de faltas de página
- Assim, quanto maior essa taxa, pior será o desempenho do sistema

Nível de multiprogramação

- Compartilhamento da memória entre muitos segmentos diminui o espaço de trabalho de cada processo

- Isso implica em menos páginas fazendo parte do conjunto residente do processo

Thrashing

- A diminuição do conjunto residente implica numa maior taxa de faltas de páginas
- Dependendo do nível de multiprogramação, o volume de faltas de páginas pode levar o sistema ao fenômeno de thrashing
- Num sistema em thrashing o desempenho cai severamente para um pequeno aumento no nível de multiprogramação

Coerência de cache

- Múltiplas cópias devem permanecer idênticas
- Técnicas para manutenção de coerência:

  write-through – escreve a alteração na memória imediatamente

  write-back – escreve na memória quando a linha sair do cache