

## Gerência de Memória

### Algoritmos de Substituição de Páginas

### Algoritmo NRU – Not Recently Used (1)

- Ou seja, algoritmo de substituição da página “não usada recentemente”
- Na maioria dos computadores com memória virtual, as entradas nas tabelas de páginas têm 2 bits de status
  - Reference bit (R) ; Modified bit (M)
- Algoritmo
  - Qdo o processo é iniciado, os bits R e M das páginas são zerados
  - Bits são sempre alterados quando a página é referenciada/modificada
  - Periodicamente o bit R é zerado (por exemplo, a cada tique de clock)
  - Quando acontece um *Page fault*, o S.O. inspeciona todas as páginas que encontram-se na memória e as separa em categorias...

### Introdução

- Quando ocorre um *Page Fault*, o S.O. deve escolher que página remover para abrir espaço em memória.
- Se a página foi alterada (bit *Modified* setado) é preciso salvá-la em disco. Senão foi, basta sobrescrevê-la.
- É melhor não escolher para remoção uma página que é usada freqüentemente, pois ela pode ter que voltar para a memória logo.
- Troca ótima de página
  - Substituir a página para a qual falta mais tempo até ser necessária novamente
    - Marcar p/ cada página, quantas instruções faltam p/ que ela seja referenciada
  - Solução ótima, mas inviável!

### Algoritmo NRU – Not Recently Used (2)

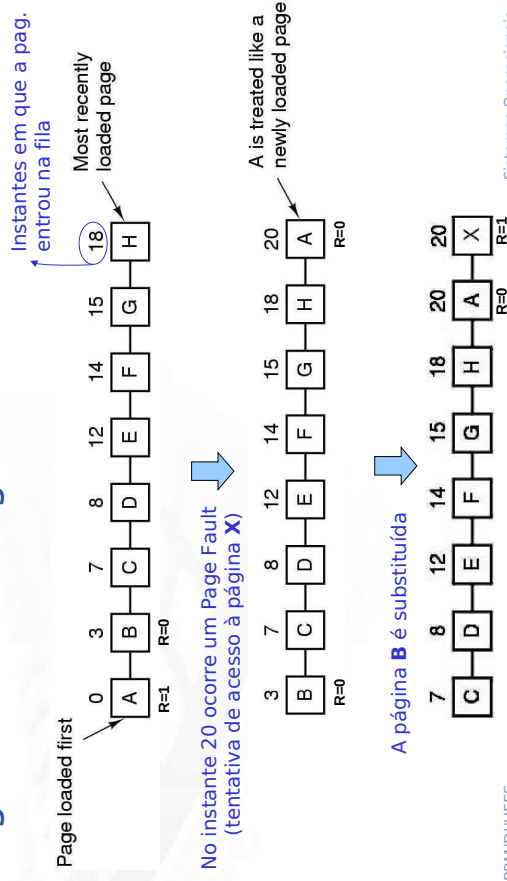
Isso pode ocorrer???

- Páginas são classificadas
  - Classe 0: *Not referenced, not modified* (R≠0, M=0)
  - Classe 1: *Not referenced, modified* (R=0, M=1)
  - Classe 2: *referenced, not modified* (R=1, M=0)
  - Classe 3: *referenced, modified* (R=1, M=1)
- O S.O. remove uma das páginas (aleatoriamente) da classe mais baixa não vazia.
- Vantagens
  - Algoritmo fácil de entender e implementar
  - Desempenho adequado

## Algoritmo FIFO

- Mantem-se uma lista encadeada de páginas ordenada pela chegada das páginas à memória.
- Quando ocorre um *Page Fault*, a página no início da lista (que é a mais antiga) é a escolhida para a troca
- Vantagem:
  - Baixo custo
- Desvantagem:
  - A página mais antiga pode ser também uma página usada muito frequentemente.
- Não empregado!

## Algoritmo SC - Segunda Chance (2)



## Algoritmo SC - Segunda Chance (1)

- Tenta melhorar o FIFO
- Cada página tem um bit R (referenciada)
- Antes de remover a página mais antiga (cabeça da fila), seu bit R é verificado
  - Se  $R=0$ , a página é substituída (a página referenciada ocupará o seu lugar na memória)
  - Se  $R=1$ , a página vai para fim da fila, como se houvesse sido carregada agora e seu bit é setado para 0
  - Verifica-se a página que virou “cabeça” da fila
- Se todas as páginas tiverem seu bit  $R=1$ , haverá uma volta completa

## Algoritmo do Relógio

- Visa melhorar o desempenho do algoritmo SC, diferenciando apenas na implementação da fila
- O ponteiro sempre aponta para a página mais antiga
- Na ocorrência de um Page fault
  - Se o bit R desta página for 0, ela é substituída, e o ponteiro “roda” uma casa
  - Se  $R=1$ , R é resetado e o ponteiro avança para a próxima página até encontrar uma página com  $R=0$

## Algoritmo LRU – Least Recently Used (1)

*Ou MRU – Menos Recentemente Usada*

- Assume que as páginas usadas recentemente voltarão a ser usadas em breve
  - Substitui páginas que estão há **mais tempo sem uso**.
- Para implementá-lo completamente, deve-se manter lista encadeada de todas as páginas que estão na memória (muito custoso!)
  - página usada mais recentemente vai para o início da lista;
    - lista é reordenada a cada referência a memória
  - qdo há Page Fault, escolhe-se a última página da fila

## Algoritmo LRU – Least Recently Used (2)

- Uma solução simples: manter uma idade para cada página.
  - Usar um contador  $C$  de 64 bits incrementado a cada instrução (em hardware)
  - Cada entrada da tabela de páginas deve ter um campo extra para armazenar o valor do contador
  - A cada referência à memória o valor corrente de  $C$  é armazenado na entrada da tabela de páginas na posição correspondente à página referenciada
  - Quando ocorre um Page Fault, a tabela de páginas é examinada, a entrada cujo campo  $C$  é de menor valor é a escolhida
  - Substitui página com o menor valor no campo do contador (maior idade)

## Algoritmo LRU – Least Recently Used (3)

Seq. de instruções: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 → Referências à Mem. Prin.

	P	Campo C
Pag. 0	1	1..10
Pag. 1	1	7
Pag. 2	0	
Pag. 3	1	4
Pag. 4	0	
Pag. 5	0	10

CONTADOR **10**

Page Fault

Página que será substituída!

Tabela de Páginas

## Algoritmo LRU – Least Recently Used (4)

- LRU usando matrizes
  - HW especial que mantém uma matriz  $n \times n$ , onde  $n$  é o número de molduras
  - Inicialmente todos os bits da matriz são 0
  - Sempre que a moldura  $k$  é referenciada, o hardware seta todos os bits da linha  $k$  para 1, e depois zera todos os bits da coluna  $k$  para zero
  - Deste modo, a qualquer instante a **linha** com o **menor valor binário** é a menos recentemente usada

## Algoritmo LRU - Least Recently Used (5)

- LRU usando matrizes (cont.)

Página na moldura 0				Página na moldura 1				Página na moldura 2				Página na moldura 3				
0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	
0	0	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	1	0	1	1	1	0	0	0	1	0	0	0
2	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Página na moldura 1				Página na moldura 0				Página na moldura 3				Página na moldura 2			
0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
0	0	0	0	0	1	1	1	0	1	1	0	0	1	0	0
1	0	1	1	0	0	1	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	1	1	1	0	1	1	1	0	1
1	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0

## Algoritmo LRU - Simulando LRU em Software (1)

- Problema das abordagens em HW
  - Dependem de um HW especial
  - Procurar uma solução em SW
- Simulando LRU em Software
  - Algoritmo NFU** - Not Frequently Used
    - Um contador por página na memória
    - A cada tick, o S.O. percorre todas as páginas na memória e soma o bit R (0 ou 1) de cada página ao seu respectivo contador
    - Na ocorrência de *Page Fault*, a página c/ o menor contador é substituída
    - Problema: o algoritmo nunca esquece (reseta) o contador

## Algoritmo LRU - Least Recently Used (5)

- LRU usando matrizes (cont.)

Página na moldura 0				Página na moldura 1				Página na moldura 2				Página na moldura 3				
0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	
0	0	1	1	1	0	0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	1	0	1	1	1	0	0	0	1	0	0	0
2	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1
3	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Página na moldura 1				Página na moldura 0				Página na moldura 3				Página na moldura 2			
0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
0	0	0	0	0	1	1	1	0	1	1	0	0	1	0	0
1	0	1	1	0	0	1	1	0	0	1	0	0	0	0	0
1	0	0	1	0	0	0	1	1	1	0	1	1	1	0	1
1	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0

## Algoritmo LRU - Simulando LRU em Software (2)

- Algoritmo Aging**
  - Desloca o contador de 1 bit p/ a direita
  - Soma R ao bit mais significativo do contador

Página	Bits R para páginas 0-5 em t0	Bits R para páginas 0-5 em t1	Bits R para páginas 0-5 em t2	Bits R para páginas 0-5 em t3	Bits R para páginas 0-5 em t4
0	10110111	11101010	11101000	11000110	01110100
1	10000000	11000000	11100000	11110000	01111000
2	10000000	10000000	11000000	01100000	10110000
3	10000000	00000000	00100000	00100000	10001000
4	10000000	11000000	01100000	10110000	00100000
5	10000000	01000000	10100000	01010000	00101000

## Algoritmo LRU - Simulando LRU em Software (1)

- Problema das abordagens em HW
  - Dependem de um HW especial
  - Procurar uma solução em SW
- Simulando LRU em Software
  - Algoritmo NFU** - Not Frequently Used
    - Um contador por página na memória
    - A cada tick, o S.O. percorre todas as páginas na memória e soma o bit R (0 ou 1) de cada página ao seu respectivo contador
    - Na ocorrência de *Page Fault*, a página c/ o menor contador é substituída
    - Problema: o algoritmo nunca esquece (reseta) o contador

## Algoritmo LRU - Simulando LRU em Software (2)

- Algoritmo Aging**
  - Desloca o contador de 1 bit p/ a direita
  - Soma R ao bit mais significativo do contador

Página	Bits R para páginas 0-5 em t0	Bits R para páginas 0-5 em t1	Bits R para páginas 0-5 em t2	Bits R para páginas 0-5 em t3	Bits R para páginas 0-5 em t4
0	10110111	11101010	11101000	11000110	01110100
1	10000000	11000000	11100000	11110000	01111000
2	10000000	10000000	11000000	01100000	10110000
3	10000000	00000000	00100000	00100000	10001000
4	10000000	11000000	01100000	10110000	00100000
5	10000000	01000000	10100000	01010000	00101000