

# Aula 14

# Sistemas Operacionais I

## **Gerenciamento de Memória - Parte 3**

Prof. Julio Cezar Estrella

jcezar@icmc.usp.br

*Material adaptado de*

*Sarita Mazzini Bruschi*

*baseados no livro Sistemas Operacionais Modernos de A. Tanenbaum*

# Memória Virtual

## Paginação

- Algumas questões:
  - Onde armazenar a tabela de páginas?
  - Qual a estrutura de uma entrada na tabela de páginas?
  - Quantas páginas reais serão alocadas a um processo?
  - Quando uma página deve ser carregada para a memória?
  - Como trazer uma página para a memória?
  - Como liberar espaço na memória?

# Memória Virtual

## Paginação

- Algumas questões:
  - Onde armazenar a tabela de páginas?
  - Qual a estrutura de uma entrada na tabela de páginas?
  - Quantas páginas reais serão alocadas a um processo?
  - Quando uma página deve ser carregada para a memória?
  - Como trazer uma página para a memória?
  - Como liberar espaço na memória?

# Memória Virtual

## Paginação

- A Tabela de páginas pode ser armazenada de três diferentes maneiras:
  - Em um conjunto de registradores, se a memória for pequena;
    - Vantagem: rápido
    - Desvantagem: precisa carregar toda a tabela nos registradores a cada chaveamento de contexto
  - Na própria memória RAM → MMU gerencia utilizando dois registradores:
    - Registrador Base da tabela de páginas (PTBR – *page table base register*): indica o endereço físico de memória onde a tabela está alocada;
    - Registrador Limite da tabela de páginas (PTLR – *page table limit register*): indica o número de entradas da tabela (número de páginas);
    - Precisa de dois acessos à memória: um para acessar a tabela de páginas e outro para acessar a posição de memória;

# Memória Virtual

## Paginação

- Em uma memória *cache* na MMU
  - Também conhecida como TLB (*Translation Lookaside Buffer* - *buffer* de tradução dinâmica);
  - Hardware especial para mapear endereços virtuais para endereços reais sem ter que passar pela tabela de páginas na memória principal;
  - Melhora o desempenho;

# Memória Virtual

## Paginação

- Memória Associativa (TLB)
  - De 64 a 4096 entradas

<b><i>Bit R</i></b>	<b>Página Virtual</b>	<b><i>Bit M</i></b>	<b><i>Bits de Proteção</i></b>	<b>Página Física</b>
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

# Memória Virtual

## Paginação

- Algumas questões:
  - Onde armazenar a tabela de páginas?
  - Qual a estrutura de uma entrada na tabela de páginas?
  - Quantas páginas reais serão alocadas a um processo?
  - Quando uma página deve ser carregada para a memória?
  - Como trazer uma página para a memória?
  - Como liberar espaço na memória?

# Memória Virtual

## Paginação

- Estrutura de uma tabela de páginas (normalmente 32 bits)



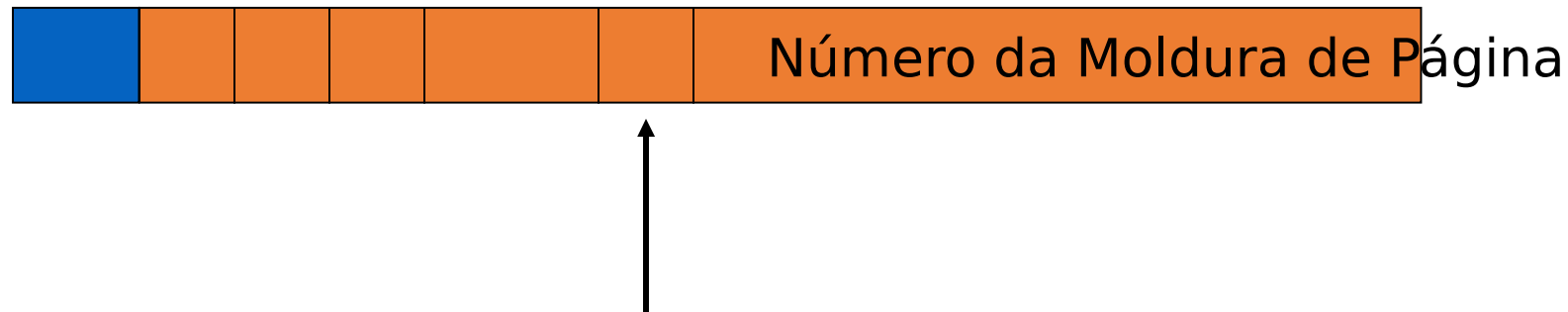
**Identifica a página real;  
Campo mais importante;**



# Memória Virtual

## Paginação

- Estrutura de uma tabela de páginas (normalmente 32 bits)



***Bit* de Residência:**

**Se valor igual 1, então entrada válida para uso;**

**Se valor igual 0, então entrada inválida, pois  
página virtual correspondente não está na memória;**

# Memória Virtual

## Paginação

- Estrutura de uma tabela de páginas (normalmente 32 bits)



***Bits* de Proteção:**  
**Indicam tipos de acessos permitidos:**

**1 bit → 0 - leitura/escrita**

**1 - leitura**

**3 bits → 0 - Leitura**

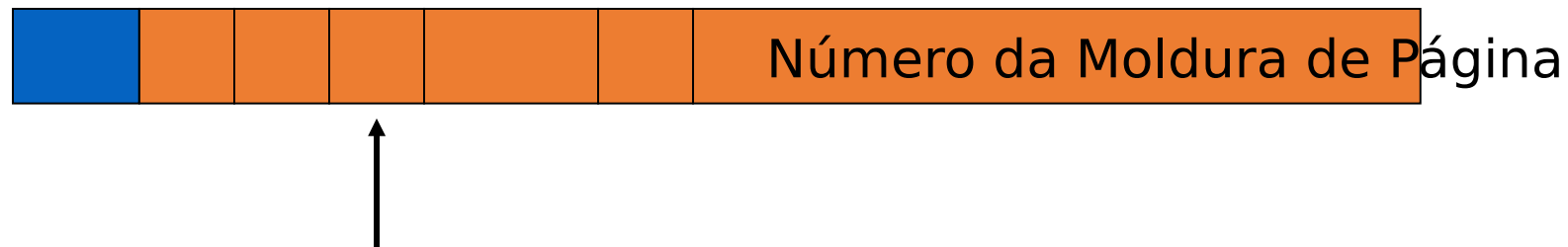
**1 - Escrita**

**2 - Execução**

# Memória Virtual

## Paginação

- Estrutura de uma tabela de páginas (normalmente 32 bits)



**Bit de Modificação (Bit M):**

**Controla o uso da página;**

**Se valor igual a 1, página foi escrita;  
página é copiada para o**

**disco**

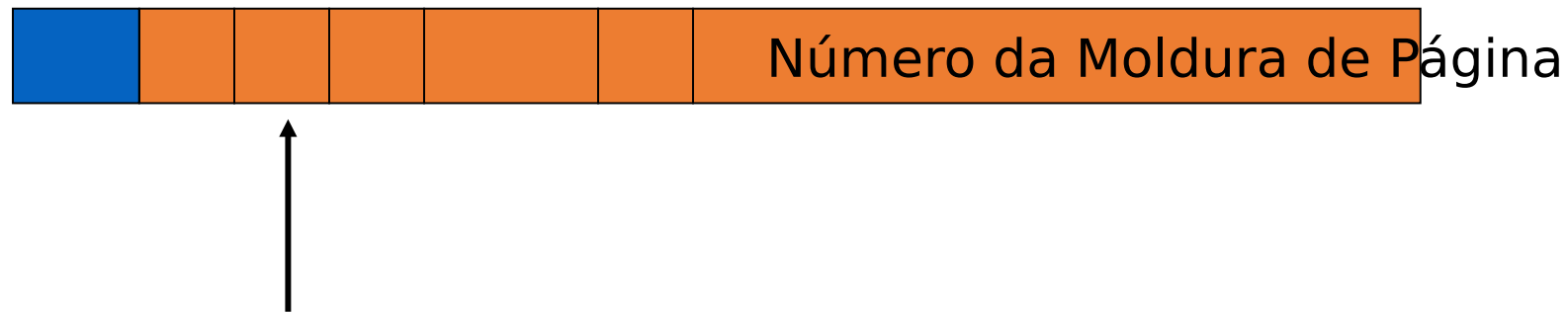
**Se valor igual a 0, página não foi  
modificada;**

**página não é copiada para o  
disco;**

# Memória Virtual

## Paginação

- Estrutura de uma tabela de páginas (normalmente 32 bits)



**Bit de Referência (Bit R):**

**Controla o uso da página;**

**Auxilia o SO na escolha da página que deve deixar a MP (RAM);**

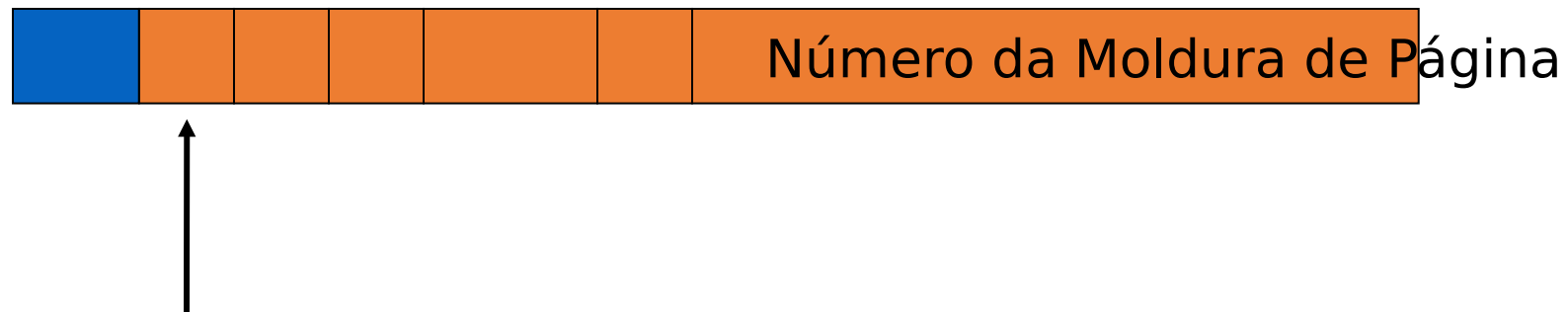
**Se valor igual a 1, página foi referenciada (leitura/escrita);**

**Se valor igual a 0, página não referenciada;**

# Memória Virtual

## Paginação

- Estrutura de uma tabela de páginas (normalmente 32 bits)



***Bit de Cache:***

**Necessário quando os dispositivos de entrada/saída são mapeados na memória e não em um endereçamento específico de E/S;**

# Memória Virtual

## Paginação

- Algumas questões:
  - Onde armazenar a tabela de páginas?
  - Qual a estrutura de uma entrada na tabela de páginas?
  - Quantas páginas reais serão alocadas a um processo?
  - Quando uma página deve ser carregada para a memória?
  - Como trazer uma página para a memória?
  - Como liberar espaço na memória?

# Memória Virtual

## Paginação - Alocação de Páginas

- Quantas páginas reais serão alocadas a um processo?
- Duas estratégias:
  - Alocação fixa ou estática: cada processo tem um número máximo de páginas reais, definido quando o processo é criado;
    - O limite pode ser igual para todos os processos;
    - Vantagem: simplicidade;
    - Desvantagens:
      - número muito pequeno de páginas reais pode causar muita paginação (troca de páginas da memória principal);
      - número muito grande de páginas reais causa desperdício de memória principal;

# Memória Virtual

## Paginação - Alocação de Páginas

- Quantas páginas reais serão alocadas a um processo?
- Duas estratégias (continuação):
  - Alocação variável ou dinâmica: número máximo de páginas reais alocadas ao processo varia durante sua execução;
    - Vantagem:
      - processos com elevada taxa de paginação podem ter seu limite de páginas reais ampliado;
      - processos com baixa taxa de paginação podem ter seu limite de páginas reais reduzido;
    - Desvantagem: monitoramento constante;



# Memória Virtual

## Paginação

- Algumas questões:
  - Onde armazenar a tabela de páginas?
  - Qual a estrutura de uma entrada na tabela de páginas?
  - Quantas páginas reais serão alocadas a um processo?
  - Quando uma página deve ser carregada para a memória?
  - Como trazer uma página para a memória?
  - Como liberar espaço na memória?

# Memória Virtual

## Paginação - Busca de Página

- Política de busca de página: determina quando uma página deve ser carregada para a memória
- Três estratégias:
  - **Paginação simples**:
    - Todas as páginas virtuais do processo são carregadas para a memória principal;
    - Sempre todas as páginas são válidas;
  - **Paginação por demanda** (*Demand Paging*):
    - Apenas as páginas referenciadas são carregadas na memória principal;
    - Quais páginas virtuais foram carregadas → Bit de controle (bit de residência);
    - Página inválida;
  - **Paginação antecipada** (*Antecipatory Paging*)
    - Carrega para a memória principal, além da página referenciada, outras páginas que podem ou não ser necessárias para o processo

# Memória Virtual

## Paginação

- Algumas questões:
  - Onde armazenar a tabela de páginas?
  - Qual a estrutura de uma entrada na tabela de páginas?
  - Quantas páginas reais serão alocadas a um processo?
  - Quando uma página deve ser carregada para a memória?
  - Como trazer uma página para a memória?
  - Como liberar espaço na memória?

# Memória Virtual

## Paginação - Busca de Página

- Página inválida: MMU gera uma interrupção de proteção e aciona o sistema operacional;
  - Se a página está fora do espaço de endereçamento do processo, o processo é abortado;
  - Se a página ainda não foi carregada na memória principal, ocorre uma **falta de página** (*page fault*);

# Memória Virtual

## Paginação - Busca de Página

- **Falta de Página:**

- Processo é suspenso e seu descritor é inserido em uma **fila especial** – fila dos processos esperando uma página virtual;
- Uma página real livre deve ser alocada;
- A página virtual acessada deve ser localizada no disco;
- Operação de leitura de disco, indicando o endereço da página virtual no disco e o endereço da página real alocada;

# Memória Virtual

## Paginação - Busca de Página

- Após a leitura do disco:
  - Tabela de páginas do processo é corrigida para indicar que a página virtual agora está válida e está na página real alocada;
    - *Pager*: carrega páginas específicas de um processo do disco para a memória principal;
  - O descritor do processo é retirado da **fila especial** e colocado na fila do processador;

# Memória Virtual

## Paginação

- Algumas questões:
  - Onde armazenar a tabela de páginas?
  - Qual a estrutura de uma entrada na tabela de páginas?
  - Quantas páginas reais serão alocadas a um processo?
  - Quando uma página deve ser carregada para a memória?
  - Como trazer uma página para a memória?
  - Como liberar espaço na memória?

# Memória Virtual

## Paginação – Troca de Página

Memória Principal

Memória Virtual

0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H

Tabela de Páginas  
Simplificada

0		i
1		i
2	10	v
3	3	v
4		i
5		i
6	4	v
7		i

Página  
Virtual

Página Real

0	
1	
2	
3	<b>D</b>
4	<b>G</b>
5	
6	
7	
8	
9	
10	<b>C</b>
11	
12	
13	
14	
15	



# Memória Virtual

## Paginação – Troca de Página

- Se todas as páginas estiverem ocupadas, uma página deve ser retirada: página vítima;
- Exemplo:
  - Dois processos P1 e P2, cada um com 4 páginas virtuais;
  - Memória principal com 6 páginas;

# Memória Virtual

## Paginação – Troca de Página

**Memória Virtual P1**

0	A
1	B
2	C
3	D

**Tabela de Páginas P1  
Simplificada**

0	1	v
1	5	v
2		i
3	0	v

**Memória Principal**

0	<b>D</b>
1	<b>A</b>
2	<b>F</b>
3	<b>E</b>
4	<b>G</b>
5	<b>B</b>

**Memória Virtual P2**

0	E
1	F
2	G
3	H

**Tabela de Páginas P2  
Simplificada**

0	3	v
1	2	v
2	4	v
3		i

**3 páginas de  
cada processo**

P2 tenta acessar página 3! **Falta de Página!**

# Memória Virtual

## Paginação – Troca de Página

**Memória Virtual P1**

0	A
1	B
2	C
3	D

**Tabela de Páginas P1  
Simplificada**

0	1	v
1	5	v
2		i
3	0	v

**Memória Principal**

0	<b>D</b>
1	<b>A</b>
2	<b>F</b>
3	<b>E</b>
4	<b>H</b>
5	<b>B</b>

**Memória Virtual P2**

0	E
1	F
2	G
3	H

**Tabela de Páginas P2  
Simplificada**

0	3	v
1	2	v
2		i
3	4	v

**3 páginas de  
cada processo**

Página 2 (virtual) do P2 é escolhida como vítima!

# Memória Virtual

## Paginação – Troca de Página

- Algoritmos:
  - Ótimo;
  - NRU;
  - FIFO;
  - Segunda Chance;
  - Relógio;
  - LRU;
  - *Working set*;
  - *WSClock*;

# Memória Virtual

## Paginação – Troca de Página

- Algoritmo Ótimo:
  - Retira da memória a página que tem menos chance de ser referenciada;
    - Praticamente impossível de se saber;
    - Impraticável;
    - Usado em simulações para comparação com outros algoritmos;

# Memória Virtual

## Paginação – Troca de Página

- Algoritmo *Not Recently Used Page Replacement* (NRU) ou *Não Usada Recentemente* (NUR)
  - Troca as páginas não utilizadas recentemente:
  - 02 bits associados a cada página → R (referência) e M (modificação)
    - Classe 0 (R = 0 e M = 0) → não referenciada, não modificada;
    - Classe 1 (R = 0 e M = 1) → não referenciada, modificada;
    - Classe 2 (R = 1 e M = 0) → referenciada, não modificada;
    - Classe 3 (R = 1 e M = 1) → referenciada, modificada;
  - R e M são atualizados a cada referência à memória;

**Uma vez que a página foi lida, o valor do bit R será sempre igual a 1 até que o SO reinicialize-o**

# Memória Virtual

## Paginação – Troca de Página

- NRU:
  - Periodicamente, o bit R é limpo para diferenciar as páginas que não foram referenciadas recentemente;
    - A cada *tick* do relógio ou interrupção de relógio;
    - Classe 3 → Classe 1;
  - Vantagens: fácil de entender, eficiente para implementar e fornece bom desempenho;

# Memória Virtual

## Paginação – Troca de Página

- Algoritmo *First-in First-out Page Replacement* (FIFO)
  - SO mantém uma lista das páginas correntes na memória;
    - A página no início da lista é a mais antiga e a página no final da lista é a mais nova;
  - Simples, mas pode ser ineficiente, pois uma página que está em uso constante pode ser retirada;
  - Pouco utilizado;



# Memória Virtual

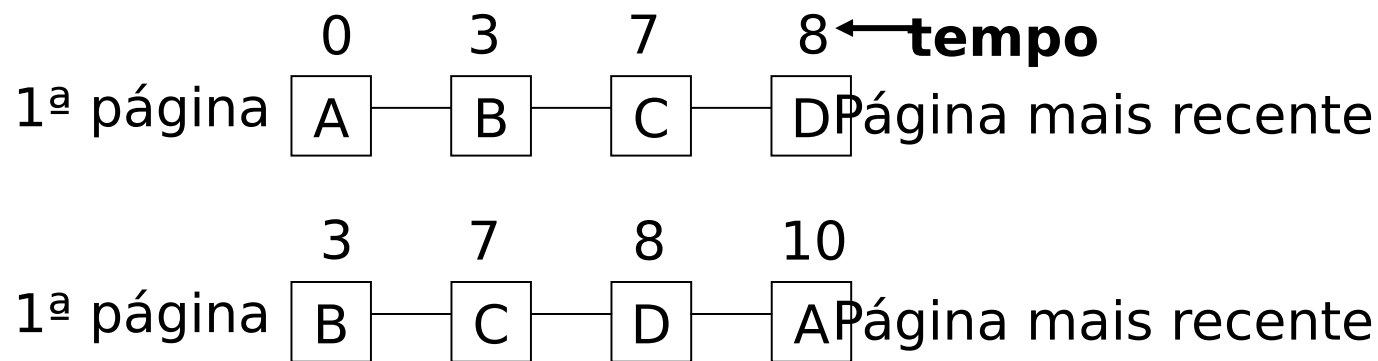
## Paginação – Troca de Página

- Algoritmo da Segunda Chance
  - FIFO + *bit* R;
  - Página mais velha é candidata em potencial;

*Se o bit  $R=0$ ,*

*então página é retirada da memória,*

*senão  $R=1$  e se dá uma nova chance à página colocando-a no final da lista;*



**Se página A com  $R=1$ ; e falta de página em tempo 10;  
Então  $R=0$  e página A vai para final da lista;**

# Memória Virtual

## Paginação – Troca de Página

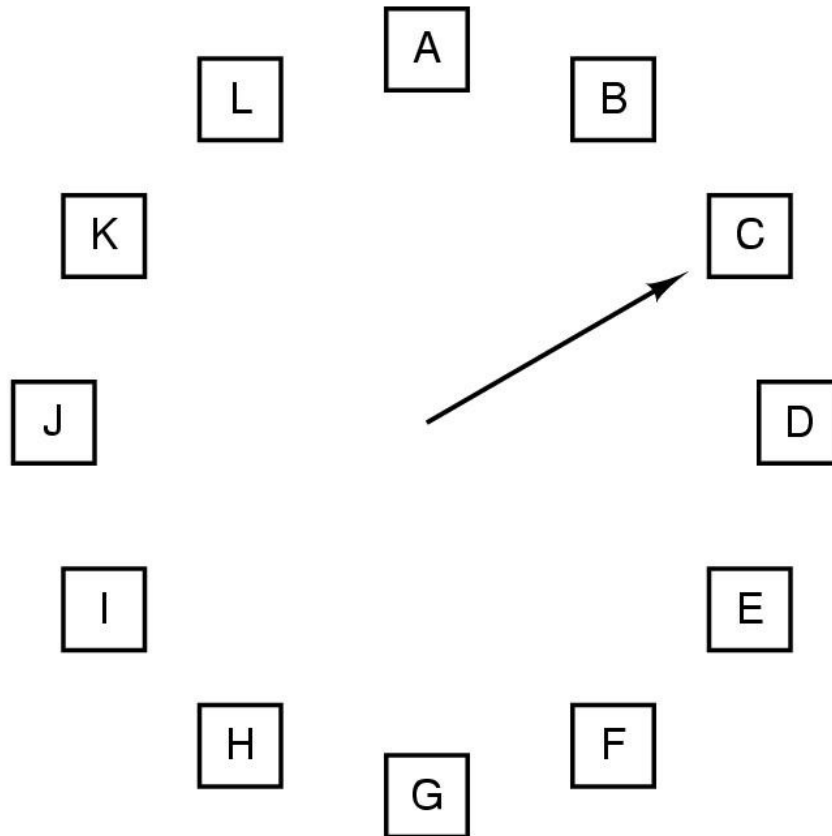
- Algoritmo do Relógio
  - Lista circular com ponteiro apontando para a página mais antiga
  - Algoritmo se repete até encontrar  $R=0$ ;

Se $R==0$ <ul style="list-style-type: none"><li>•troca de página</li><li>•desloca o ponteiro</li></ul>	Se $R==1$ <ul style="list-style-type: none"><li>•<math>R = 0</math></li><li>•desloca o ponteiro</li><li>•continua busca</li></ul>
--	---

# Memória Virtual

## Paginação – Troca de Página

- Algoritmo do Relógio



When a page fault occurs,  
the page the hand is  
pointing to is inspected.  
The action taken depends  
on the R bit:

R = 0: Evict the page

R = 1: Clear R and advance hand

# Memória Virtual

## Paginação – Troca de Página

- Algoritmo *Least Recently Used Page Replacement* (LRU) ou *Menos Recentemente Usada* (MRU)
  - Troca a página menos referenciada/modificada recentemente;
  - Alto custo
    - Lista encadeada com as páginas que estão na memória, com as mais recentemente utilizadas no início e as menos utilizadas no final;
    - A lista deve ser atualizada a cada referência da memória;

# Memória Virtual

## Paginação – Troca de Página

- Algoritmo *Least Recently Used* (LRU)
  - Pode ser implementado tanto por hardware quanto por software:
    - Hardware: MMU deve suportar a implementação LRU;
      - 1a. opção
        - Contador em hardware (64 *bits*) – conta instruções executadas;
        - Após cada referência à memória, o valor do contador é armazenado na entrada da tabela de páginas referente à página acessada;
        - Quando ocorre falta de página, o SO examina todos os contadores e escolhe a página que tem o menor valor

# Memória Virtual

## Paginação – Troca de Página

- LRU – Hardware - 2a. Opção
  - Matriz  $n \times n$  bits
  - Quando se faz uma referência à página  $k \rightarrow$   
 todos os bits da linha  $k$  recebem valor 1  
 todos os bits da coluna  $k$  recebem valor 0

	Page			
	0	1	2	3
0	0	1	1	1
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

(a)

	Page			
	0	1	2	3
0	0	0	1	1
1	1	0	1	1
2	0	0	0	0
3	0	0	0	0

(b)

	Page			
	0	1	2	3
0	0	0	0	1
1	1	0	0	1
2	1	1	0	1
3	0	0	0	0

(c)

	Page			
	0	1	2	3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0

(d)

	Page			
	0	1	2	3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	1
3	1	1	0	0

(e)

	Page			
	0	1	2	3
0	0	0	0	0
1	1	0	1	1
2	1	0	0	1
3	1	0	0	0

(f)

	Page			
	0	1	2	3
0	0	1	1	1
1	0	0	1	1
2	0	0	0	1
3	0	0	0	0

(g)

	Page			
	0	1	2	3
0	0	1	1	0
1	0	0	1	0
2	0	0	0	0
3	1	1	1	0

(h)

	Page			
	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	1	1	0	1
3	1	1	0	0

(i)

	Page			
	0	1	2	3
0	0	1	0	0
1	0	0	0	0
2	1	1	0	0
3	1	1	1	0

(j)

0  
1  
2  
3  
2  
1  
0  
3  
2  
3

# Memória Virtual

## Paginação – Troca de Página

- Algoritmo *Least Recently Used Page Replacement* (LRU)
  - Pode ser implementado tanto por hardware quanto por software:
    - Software: duas maneiras
      - NFU (*Not frequently used*) ou LFU (*least frequently used*);
      - *Aging* (Envelhecimento);

# Memória Virtual

## Paginação – Troca de Página

- Software: NFU ou LFU (*least*)
  - Para cada página existe um contador → iniciado com zero e incrementado a cada referência à página;
    - Página com menor valor do contador é candidata a troca;
    - Esse algoritmo não se esquece de nada
      - Problema: pode retirar páginas que estão sendo referenciadas com frequência;
        - Compilador com vários passos: passo 1 tem mais tempo de execução que os outros passos → páginas do passo 1 terão mais referências armazenadas;



# Memória Virtual

## Paginação – Troca de Página

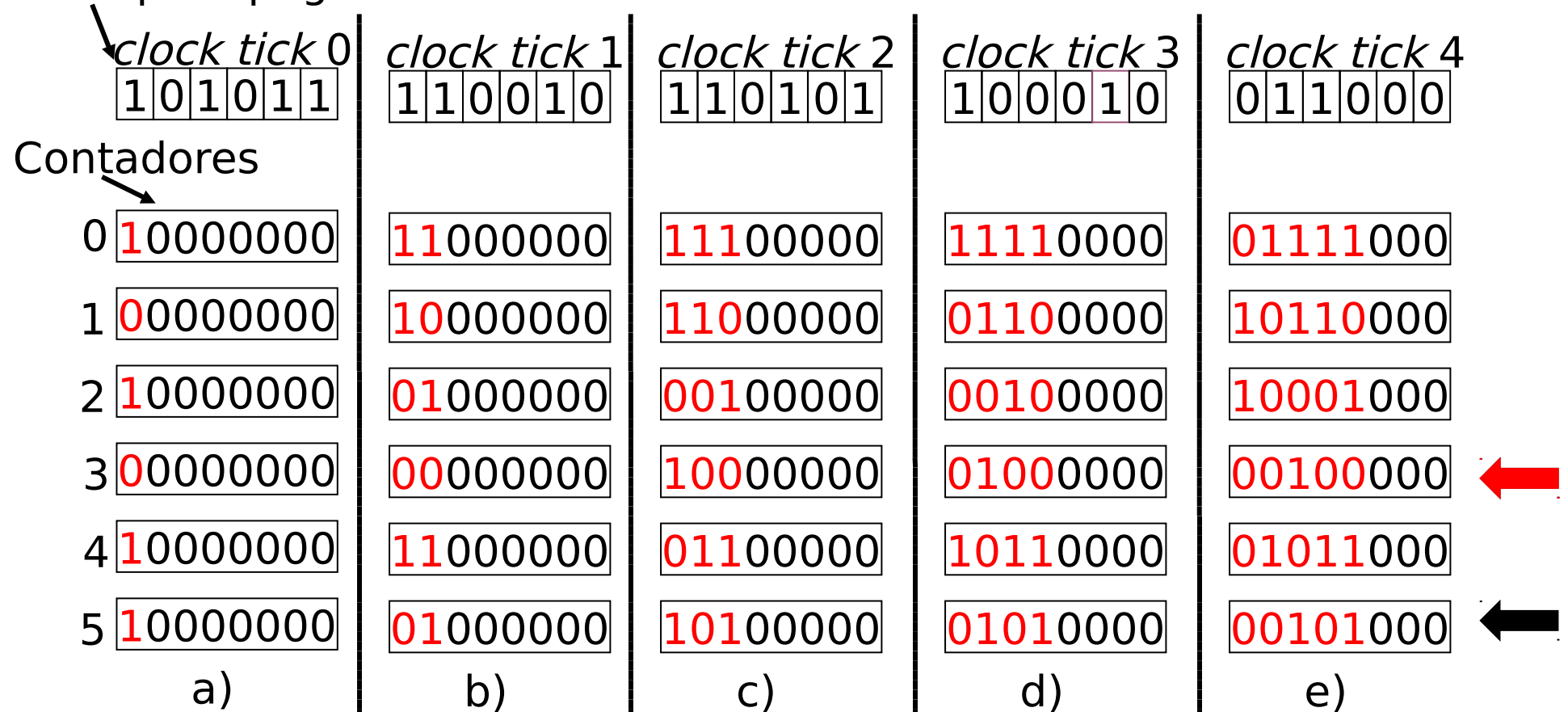
- Software: Algoritmo *aging* (envelhecimento)
  - Modificação do NFU, resolvendo o problema descrito anteriormente;
    - Além de saber **quantas vezes** a página foi referenciada, também controla **quando** ela foi referenciada;

# Memória Virtual

## Paginação – Troca de Página

- Algoritmo *aging*

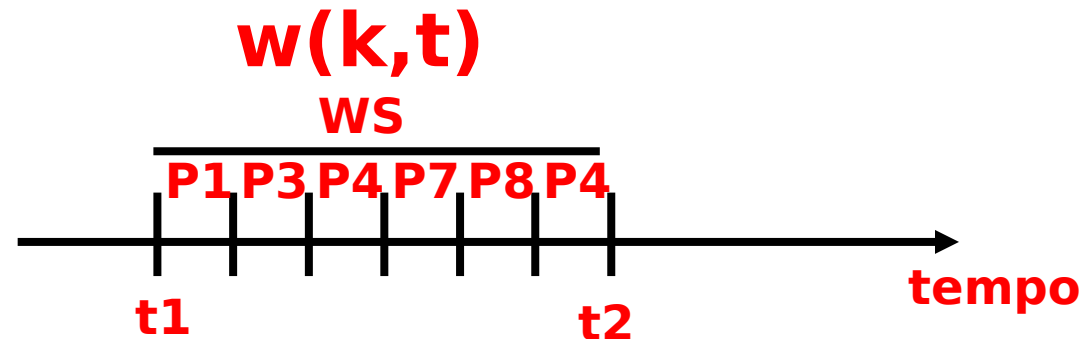
Bits R para páginas 0-5



# Memória Virtual

## Paginação – Troca de Página

- Algoritmo *Working Set (WS)*:
  - Paginação por demanda → páginas são carregadas na memória somente quando são necessárias;
  - Pré-paginação → *Working set*
    - Carregar um conjunto de páginas que um processo está efetivamente utilizando (referenciando) em um determinado tempo  $t$  antes de ele ser posto em execução;



# Memória Virtual

## Paginação – Troca de Página

- Algoritmo *Working Set (WS)*:
  - Objetivo principal: reduzir a falta de páginas
    - Um processo só é executado quando todas as páginas necessárias no tempo  $t$  estão carregadas na memória;
    - SO gerencia quais páginas estão no *Working Set*;
  - Para simplificar  $\rightarrow$  o *working set* pode ser visto como o conjunto de páginas que o processo referenciou durante os últimos  $t$  segundos de tempo;
  - Utiliza *bit* R e o tempo de relógio (tempo virtual) da última vez que a página foi referenciada;

# Memória Virtual

## Paginação – Troca de Página

Tempo virtual atual (CVT): 2204  
 $age = CVT - TLU$   
(Ex.:  $2204 - 2084 = 120$ )  
 $\tau = \text{múltiplos } clock \text{ ticks}$

- Algoritmo Working Set

\* Se todas as páginas estiverem com  $R=1$ , uma página é escolhida aleatoriamente;  
\*\* Se todas as páginas estiverem no WS, a página mais velha com  $R=0$  é escolhida;

Tempo do Último Uso (TLU)		Bit R
→ 2084		1
2003		1
1980		1
1213		0
2014		1
2020		1
2032		1
1620		0

**Tabela de Páginas**

Percorrer as páginas examinando os bit R;  
Se  $(R=1)^*$   
página foi referenciada;  
faz TLU da página igual ao CVT;  
Se  $(R=0 \text{ e } age > \tau)$   
página não está no *working set*;  
remove a página;  
Se  $(R=0 \text{ e } age \leq \tau)^{**}$   
página está no *working set*;  
guarda página com maior *age*;

# Memória Virtual

## Paginação – Troca de Página

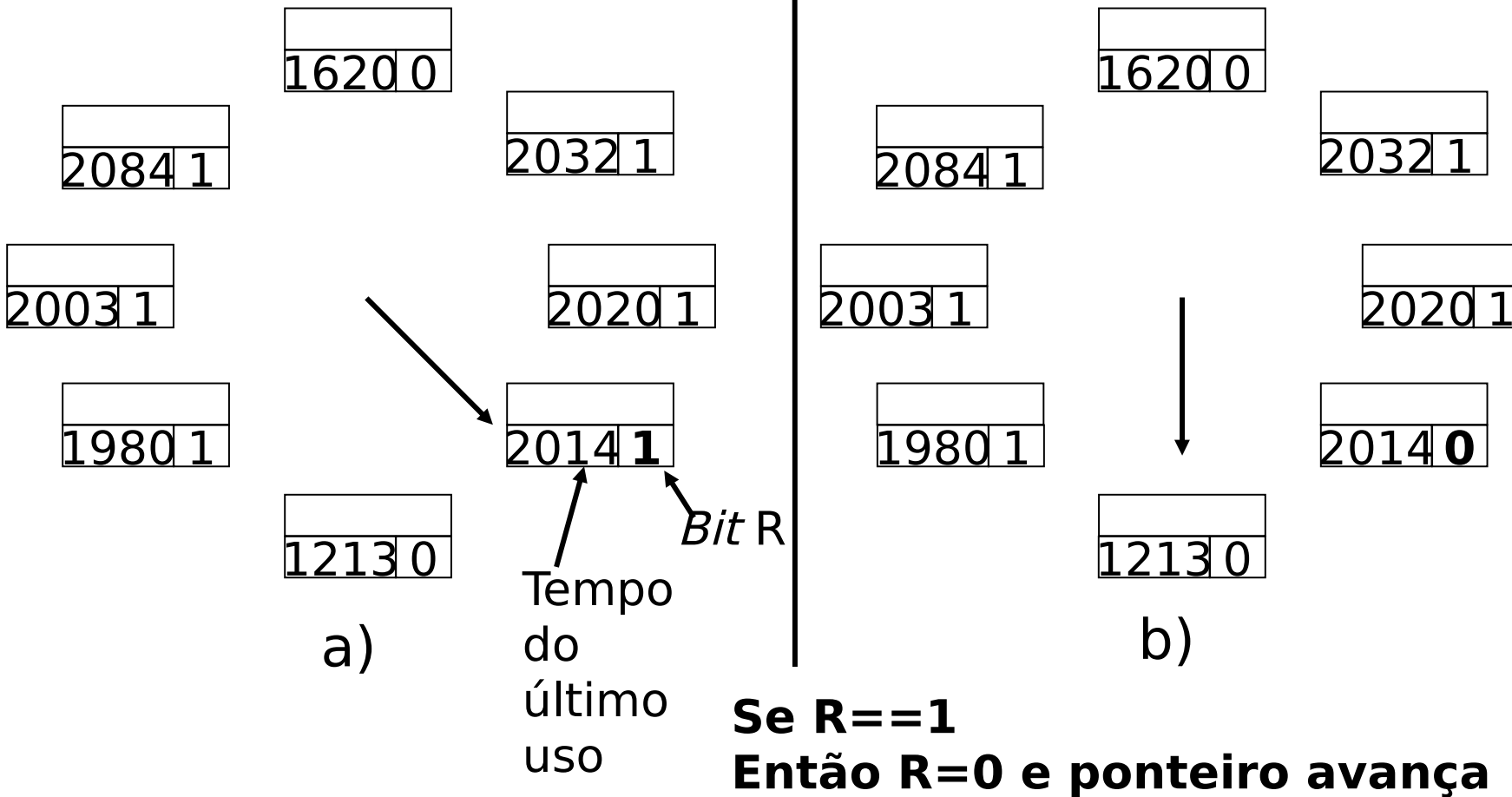
- Algoritmo *WSClock*:
  - *Clock + Working Set*;
  - Lista circular de molduras de páginas formando um anel a cada página carregada na memória;
  - Utiliza *bit R* e o tempo da última vez que a página foi referenciada;
  - *Bit M* utilizado para agendar escrita em disco;

# Memória Virtual

## Paginação – Troca de Página

- Algoritmo *WSClock*:

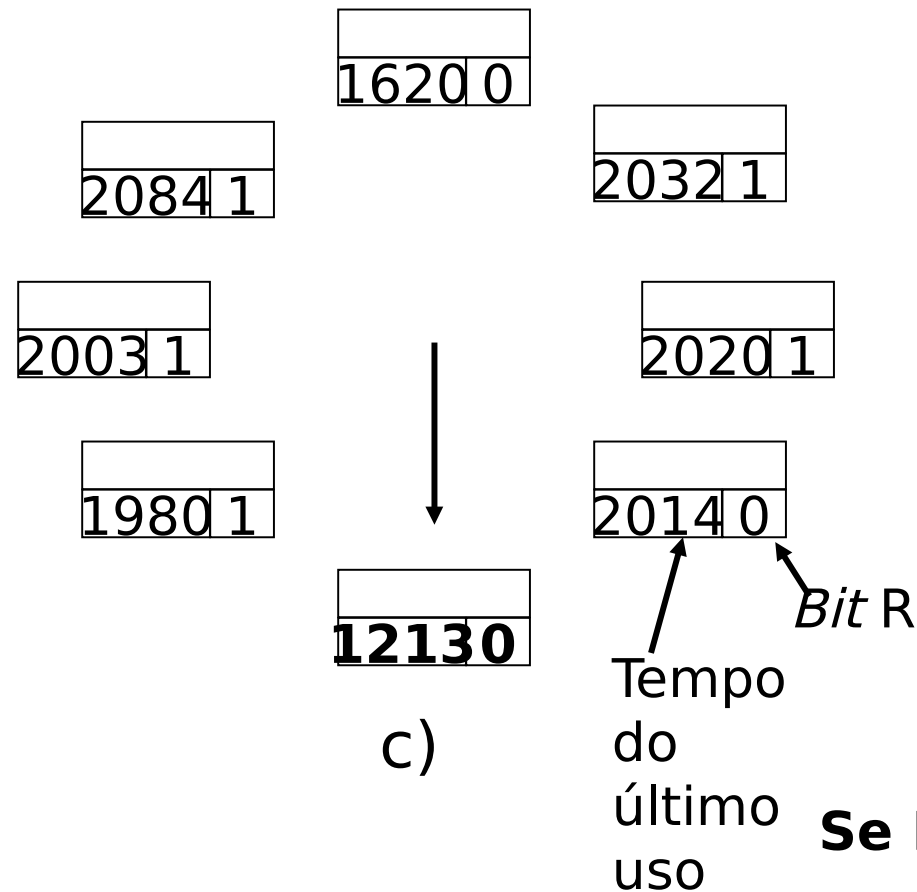
Tempo virtual atual: 2204



# Memória Virtual

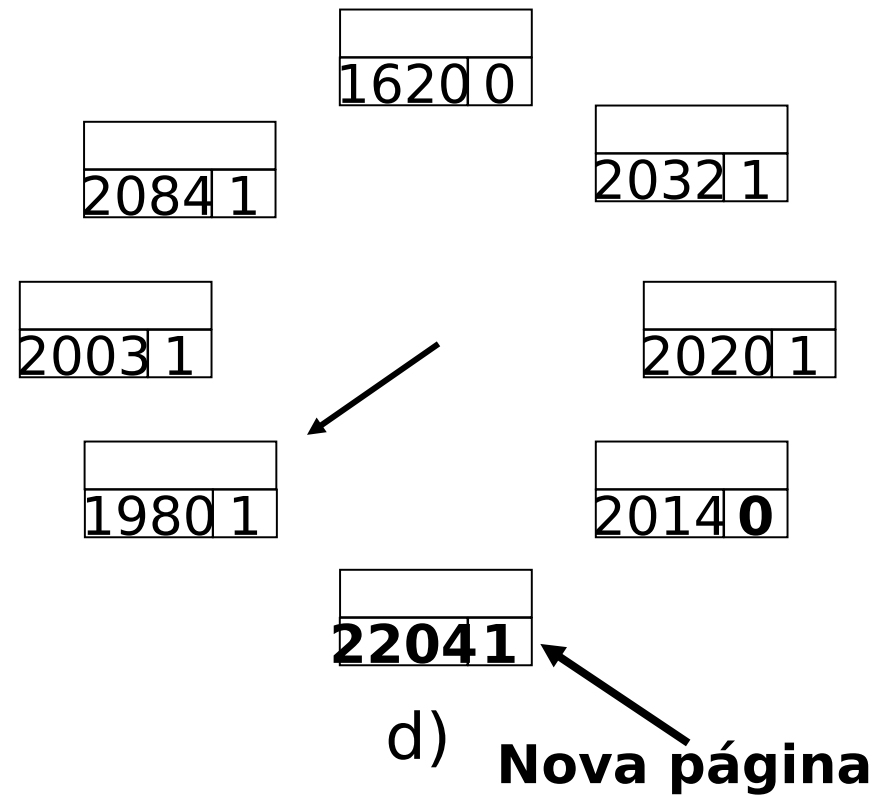
## Paginação – Troca de Página

- Algoritmo *WSClock*:



c)

Tempo virtual atual: 2204



d)

Nova página

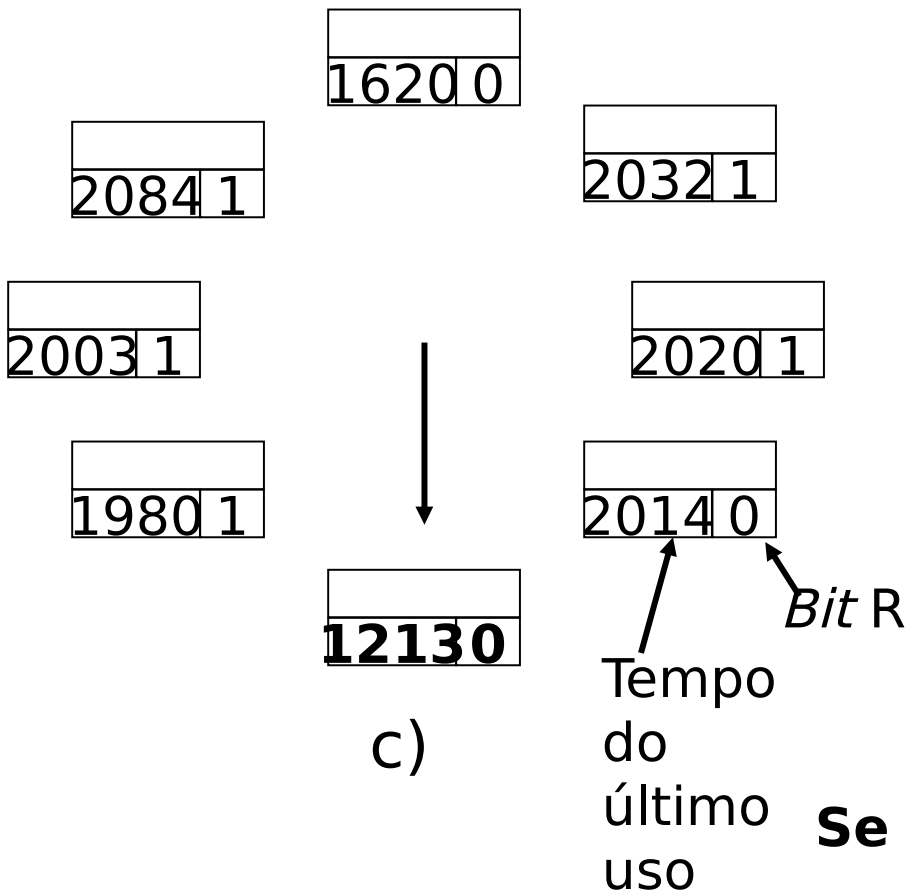
**Se  $R==0$  e  $age > t$  e  $M==0$   
não agenda escrita  $\rightarrow$  troca**



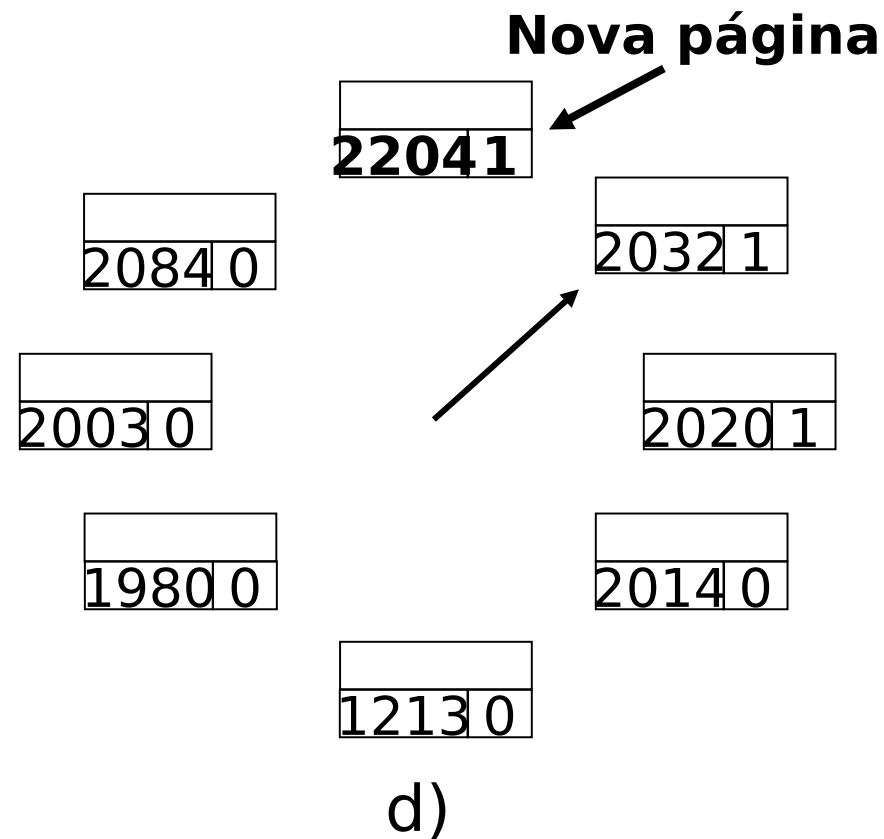
# Memória Virtual

## Paginação – Troca de Página

- Algoritmo *WSClock*:



Tempo virtual atual: 2204

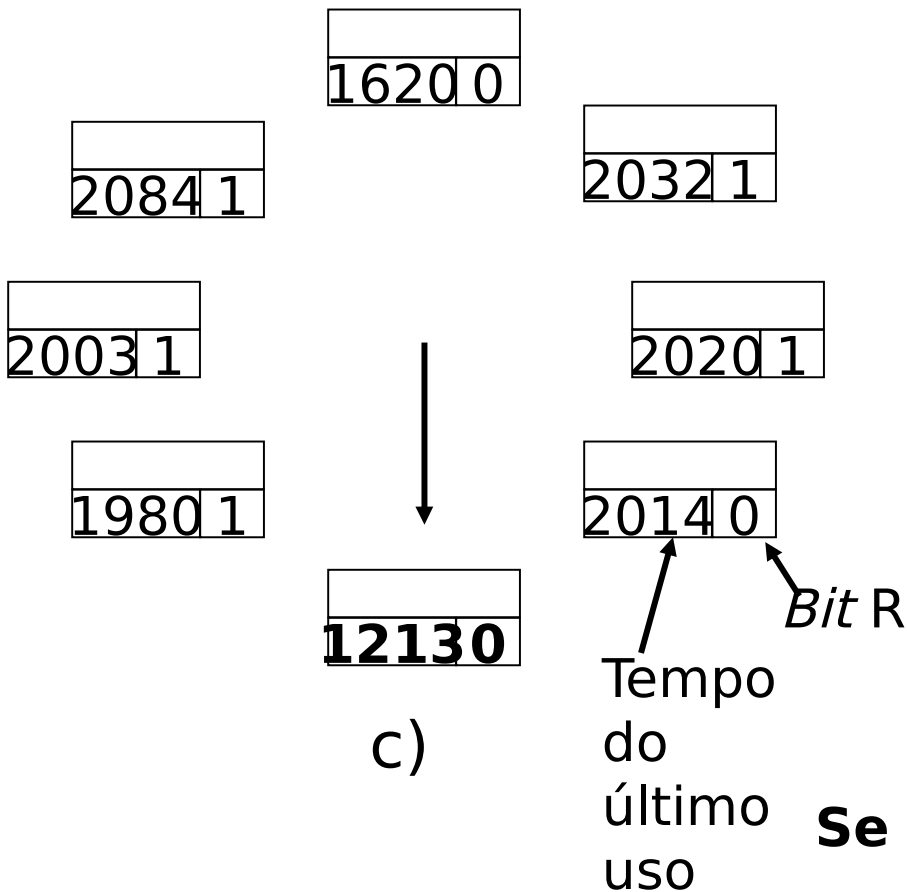


**Se  $R==0$  e  $age > t$  e  $M==1$   
agenda escrita e continua procura**

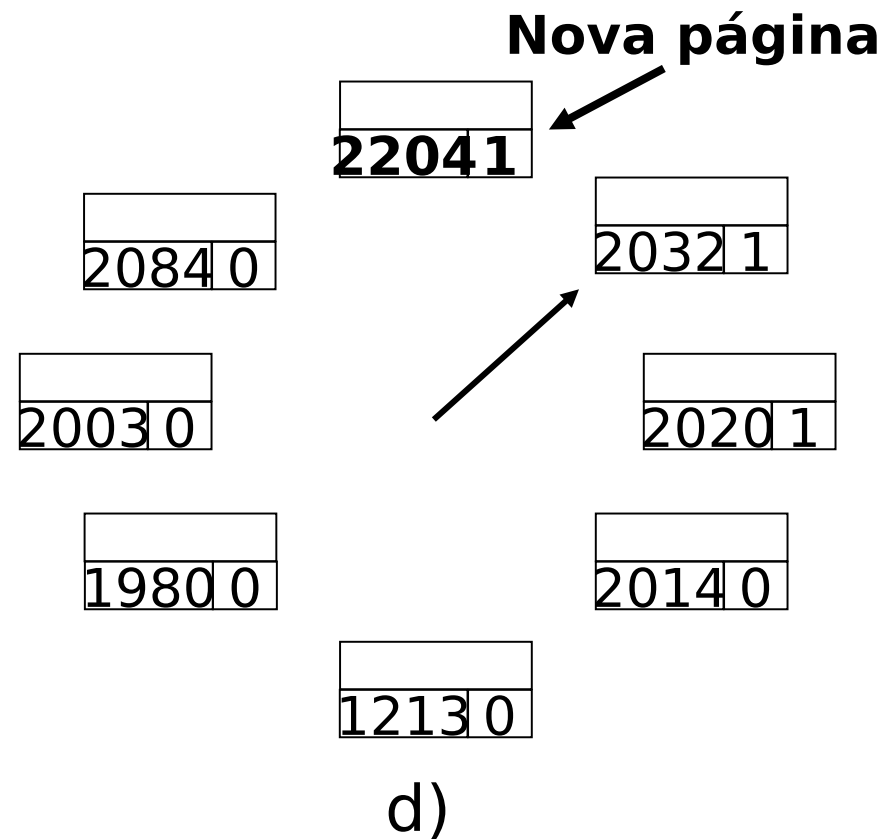
# Memória Virtual

## Paginação – Troca de Página

- Algoritmo *WSClock*:



Tempo virtual atual: 2204



**Se  $R==0$  e  $age > t$  e  $M==1$   
agenda escrita e continua procura**