Movimentação de bloco no Ramses

soluções para o exercício 4

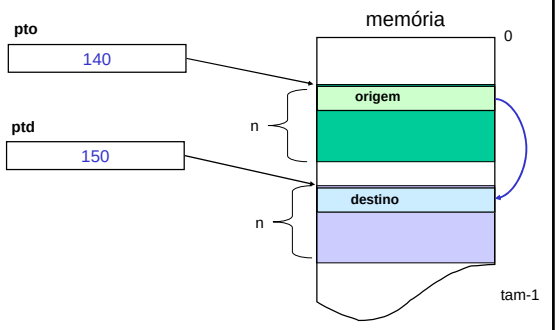
movimento de bloco

- mover n posições na memória
- posição 128 – número de posições (n)
- posição 129 – posição inicial de origem (pto)
- posição 130 – posição inicial de destino (ptd)

n, pto e ptd são variáveis

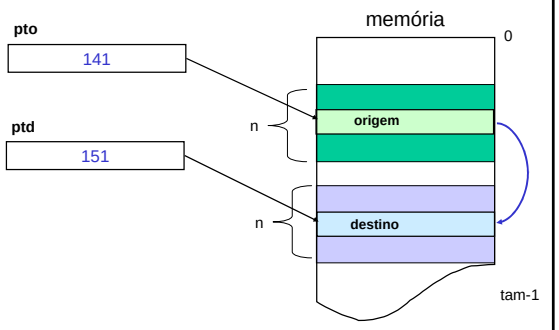
2

solução com ponteiros



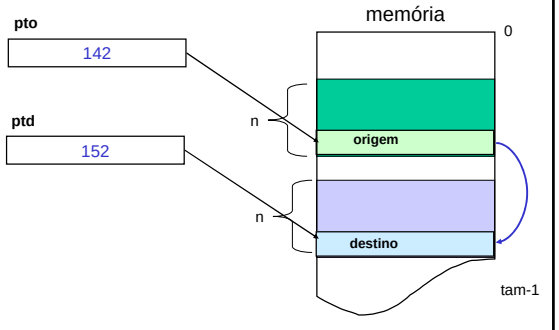
3

solução com ponteiros



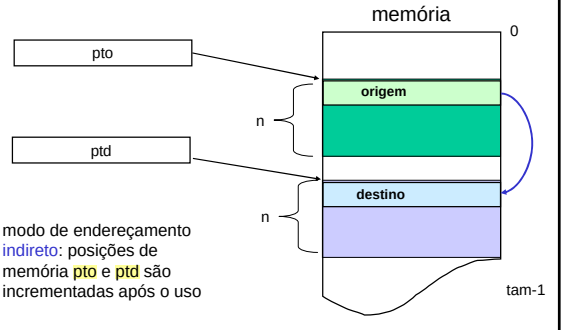
4

solução com ponteiros



5

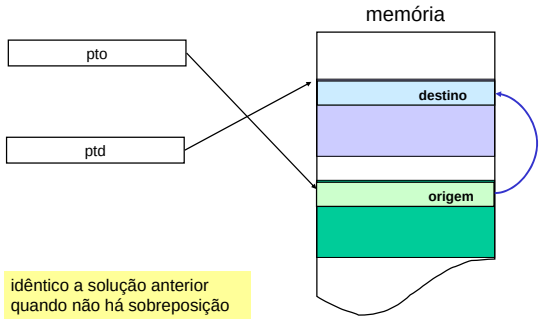
solução com ponteiros



modo de endereçamento indireto: posições de memória pto e ptd são incrementadas após o uso

6

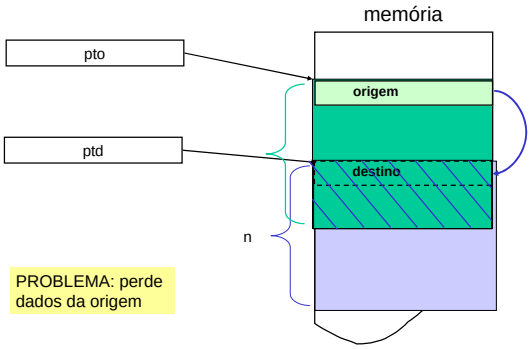
endereço origem maior



idêntico a solução anterior
quando não há sobreposição

7

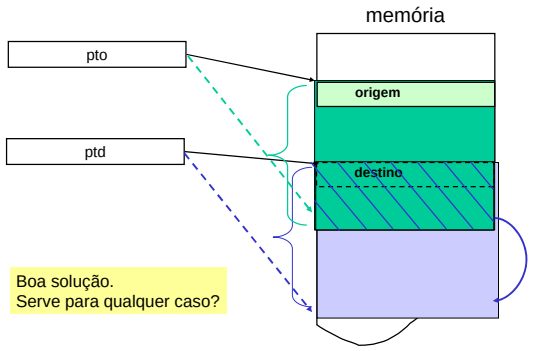
endereço origem menor com sobreposição



PROBLEMA: perde
dados da origem

8

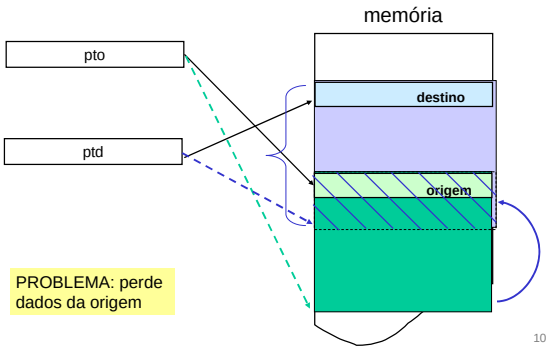
endereço origem menor com sobreposição



Boa solução.
Serve para qualquer caso?

9

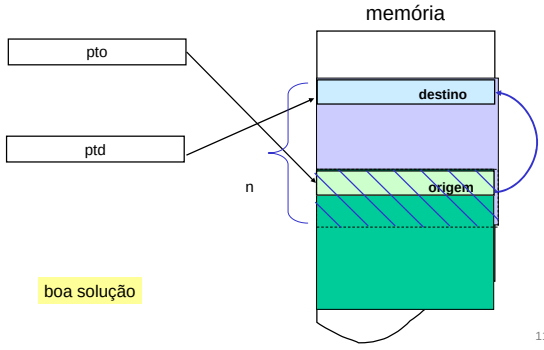
endereço origem maior com sobreposição



PROBLEMA: perde
dados da origem

10

endereço origem maior com sobreposição



boa solução

11

?

- o que fazer para evitar copiar em cima dos dados de origem que ainda não foram movidos?
- que modos de endereçamento são mais apropriados para a movimentação?
- como comparar ponteiros?
- como otimizar o código?

12

?

- o que fazer para evitar copiar em cima dos dados de origem que ainda não foram movidos?
  - determinar qual o ponteiro é o maior
    - **se** o destino for maior que a origem **então** posicionar o ponteiros para os endereços no final do bloco

13

?

- que modos de endereçamento são mais apropriados para a movimentação?
  - indireto e indexado
    - mas como há apenas um registrador de índice fica muito difícil apenas usar o indexado

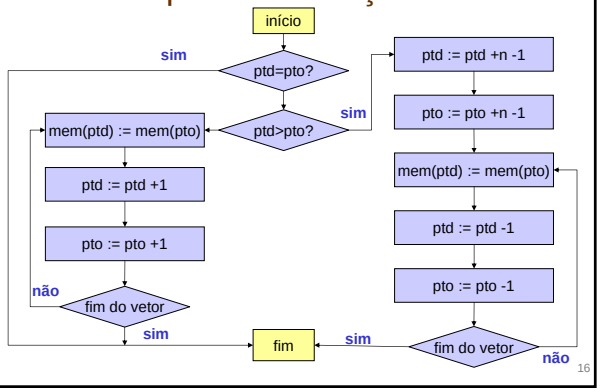
14

?

- como comparar ponteiros?
  - ponteiros são endereços de memória, logo são números que podem ser manipulados por operações aritméticas
  - por exemplo: subtração seguido por teste de CC
- como otimizar o código?
  - otimização é arte

15

primeira solução



16

primeira solução – parte 1

```
inicio:     ldr a ptd
            sub a pto
            jz fim                  ; ponteiros iguais - não faz nada
            jc ptd_maior_que_pto    : endereço de destino é maior
ptd_menor_que_pto:                  ; endereço de destino é menor
            ldr x n
laco_1:     ldr a pto,i             ; incrementa ponteiros de origem
            str a ptd,i             ; e destino usando n para controle
            ldr a pto               ; do número de incrementos
            add a #1
            str a pto
            ldr a ptd
            add a #1
            str a ptd
            sub x #1                ; teste de fim do laço
            jz fim
            jmp laco_1
```

17

parte 2

```
ptd_maior_que_pto:
            ldr x n
            ldr a pto               ;ajusta pto para final do bloco
            add a n
            sub a #1
            str a pto
            ldr a ptd               ;ajusta ptd para final do bloco
            add a n
            sub a #1
            str a ptd
laco_2:     ldr a pto,i
            str a ptd,i
            ldr a pto
            sub a #1
            str a pto
            ldr a ptd
            sub a #1
            str a ptd
            sub x #1
            jz fim
            jmp laco_2
```

```
fim:        hlt

            org 128
n:          db 16
pto:        db 140
ptd:        db  200

            org 140
            dab 1,2,3,4,5,6,7,8,9
            dab 10,11,12,13,14,15,16,17
```

18

Outras soluções

- solução 1 – ponteiros
  - pós-incremento, pós-decremento
- solução 2 – pequena otimização
  - primeiro decrementa, depois move
  - pós incremento, pré-decremento
- solução 3 – modo indexado
- solução 4 – indexado
  - armazenando endereços nas instruções
- solução 5 – indexado
  - com diferença entre endereços

19

+ exercícios

- exercício 2
  - somar duas variáveis de 16 bits (pg 173)
- exercício 10
  - subrotina para contar o número de bits com valor 1 em uma palavra de 8 bits (pg 174)

- soluções no fim do livro
  - estudar!

20