Universidade Positivo Sistemas de Informação 11/02/2019 - 12:35:13.0
Estruturas de Dados e Arquivos Prof Dr P Kantek (pkantek@up.edu.br)
Manipulação de filas em Assembler VIVO323a V: 3.21
0
Exercício : 1 _____ / _____ / _____

Filas em Assembler

Neste exercício serão examinadas implementações de filas circulares seqüenciais e encadeadas com 2 apontadores em baixo nível, ou seja usando apenas as funções básicas de manipulação de memória. Para poder estudar o funcionamento dos algoritmos usaremos a memória M, formada por 100 inteiros numerados (endereçados) de 00 a 99. A obtenção de memória é fornecida pela função OBTEMEMO que recebe como parâmetro a quantidade de inteiros pedida e devolve o endereço do primeiro deles (se houve sucesso) ou um número negativo, se não houve sucesso. Esta função trabalha baseada na variável global PPL que indica qual a Próxima Posição Livre. No início da simulação PPL ← 0.

Alocação seqüencial, fila circular Haverá um descritor composto por 6 inteiros

Lim_s	Entrada	Lim_i	Saída	Tamanho	Quantidade

```
 1: inteiro função FILCCRI (inteiro TAMANHO, QTD)
 2: inteiro X
 3: X ← OBTEMEMO (6 + TAMANHO × QTD)
 4: se X ≥ 0 então
 5:     M[X+1] ← M[X+2] ← M[X+3] ← X + 6
 6:     M[X] ← 6 + X + (TAMANHO × QTD)
 7:     M[X+4] ← TAMANHO
 8:     M[X+5] ← 0
 9:     devolva X
10: senão
11:     devolva -1
12: fim se
13: fim função
```

```
 1: inteiro função FILCINS (inteiro F, ENDEREÇO)
 2: se M[F+5] < (M[F]-M[F+2]) ÷ M[F+4] então
 3:     se M[F+1]≥M[F] então
 4:         M[F+1] ← M[F+2]
 5:     fim se
 6:     para Y de 0 até M[F+4]-1 faça
 7:         M[M[F+1]+Y] ← M[ENDEREÇO + Y]
 8:     fim para
 9:     M[F+1] ← M[F + 1] + M[F+4]
10:     M[F+5]++
11:     devolve 0
12: senão
13:     devolva -1
14: fim se
15: fim função
```

```
 1: inteiro função FILCEXC (inteiro F)
 2: se M[F+5] > 0 então
 3:     SALVA ← M[F+3]
 4:     M[F+3] ← M[F+3] + M[F+4]
 5:     se M[F+3] ≥ M[F] então
 6:         M[F+3] ← M[F+2]
 7:     fim se
 8:     M[F+5]- -
 9:     devolva SALVA
10: senão
11:     devolva -1
12: fim se
13: fim função
```

Alocação encadeada, fila com 2 apontadores

& saída	& entrada	tamanho

```
 1: inteiro função FILTCRI (inteiro TAMANHO)
 2: inteiro ONDE
 3: ONDE ← OBTEMEMO(3)
 4: se ONDE ≥ 0 então
 5:     M[ONDE] ← M[ONDE+1] ← -1
 6:     M[ONDE+2] ← TAMANHO
 7:     devolva ONDE
 8: senão
 9:     ... erro ...
10: fim se
```

```
 1: função FILTINS (inteiro F, DADOS)
 2: inteiro TAM, ONDE
 3: TAM ← M[F+2]
 4: ONDE ← OBTEMEMO(TAM+1)
 5: se ONDE ≥ 0 então
 6:     se M[F] = -1 ∧ M[F+1] = -1 então
 7:         M[F] ← M[F+1] ← ONDE
 8:         M[ONDE] ← -1
 9:         para Y de 0 até TAM-1 faça
10:             M[ONDE+Y+1] ← M[DADOS+Y]
11:         fim para
12:     senão
13:         M[ONDE] ← -1
14:         M[M[F+1]] ← ONDE
15:         M[F+1] ← ONDE
16:         para Y de 0 até TAM-1 faça
17:             M[ONDE+Y+1] ← M[DADOS+Y]
18:         fim para
19:     fim se
20: senão
21:     ...erro...
22: fim se
```

```
 1: inteiro função FILTEXC (inteiro F)
 2: se M[F] = -1 ∧ M[F+1] = -1 então
 3:     devolva -1 {erro, fila vazia}
 4: senão
 5:     devolva M[F]+1
 6:     se M[F] = M[F+1] então
 7:         M[F] ← M[F+1] ← -1
 8:     senão
 9:         M[F] ← M[M[F]]
10:     fim se
11: fim se
```

Na prática Suponha uma memória M de 100 inteiros, endereçados de 00 a 99, devidamente inicializada com zeros nas primeiras 80 posições e contendo de 1 a 20 nas posições 80 a 99.

Seja a seguinte seqüência de comandos de manipulação de filas

```
P1  <-  FILTCRI(2)          R8  <-  FILTINS(P3,90)
P2  <-  FILCCRI(2,4)        R9  <-  FILTINS(P3,86)
P3  <-  FILTCRI(4)          R10 <-  FILTINS(P1,85)
R2  <-  FILTINS(P3,84)      R11 <-  FILCINS(P2,83)
R3  <-  FILTINS(P1,87)      R12 <-  FILCEXC(P2)
R4  <-  FILTINS(P3,92)      R13 <-  FILCEXC(P2)
R5  <-  FILTEXC(P1)         R14 <-  FILTEXC(P3)
P4  <-  FILCCRI(2,3)        R15 <-  FILCEXC(P2)
P5  <-  FILTCRI(3)          R16 <-  FILCEXC(P4)
R6  <-  FILTINS(P3,82)      R17 <-  FILTEXC(P3)
R7  <-  FILCEXC(P2)
```

Que dará origem à seguinte disposição de M

M	0	1	2	3	4	5	6	7	8	9
0	63	63	2	17	11	9	11	2		4
1	5							48	58	4
2	28	5	6	7	8	-1	8	9	48	13
3	14	15	16	45	39	39	39	2		
4						-1	-1	3	53	3
5	4	5	6	58	11	12	13	14	-1	7
6	8	9	10	-1	6	7				
7										
8	1	2	3	4	5	6	7	8	9	10
9	11	12	13	14	15	16	17	18	19	20

☞ Para você fazer

Inicialize M como no começo do exemplo. Depois, execute a seguinte seqüência de comandos de pilha. Ao final, informe as posições de M que são pedidas.

```
 0:P1 =FILTCRI( 2)
 1:P2 =FILCCRI( 3, 4)
 2:P3 =FILTCRI( 2)
 3:R2=FILTINS(P3, 80)
 4:R3=FILTINS(P1, 86)
 5:R4=FILTINS(P3, 82)
 6:R5=FILTEXC(P1)
 7:P4 =FILCCRI( 2, 2)
 8:R20=FILCINS(P4, 80)
 9:R12=FILCEXC(P2)
10:R15=FILCEXC(P2)
11:R19=FILTEXC(P1)
12:R14=FILTEXC(P3)
13:R16=FILCEXC(P4)
14:R9=FILTINS(P3, 84)
15:R21=FILCINS(P2, 82)
16:R6=FILTINS(P3, 81)
17:R17=FILTEXC(P3)
18:P5 =FILTCRI( 4)
19:R23=FILTINS(P1, 84)
```

5	11	33	36	52

para quem quiser implementar

```
1: global M[0..79]← 0;M[80..99]← 1..20;PPL← 0
2: inteiro função OBTEMEMO (inteiro X)
3: se PPL+X ≥ 99 então
4:     devolva -1
5: senão
6:     devolva PPL
7:     PPL ← PPL + X
8: fim se
```


- 0 - /