```
Unit Loader;

Interface

Uses Crt, Global, BCP, Memory;

        Procedure Loader_Inicio;
        Procedure Loader_Fim;

Implementation

{ inicializa a tabela de paginas de um Programa e carrega sua primeira pagina
{ de código na Memoria para que fique em condicao de ser executado }
procedure  Loader_Inicio ;
  var I : integer;
  Begin

    if   ( RC.Disco = 0 ) and              { Disco disponível }
         ( not Fila_Vazia_BCP ( 2 ) )      { Programa residente em disco }

      then Begin

              Aloca_Pag_Mem ( Tab_Loader );        { pag p/ tabela de pag }
              Aloca_Pag_Mem ( Pag_Loader );        { pag p/ 1ª pag de código }

              if ( Tab_Loader <> Nil ) and (Pag_Loader <> nil )
                 then Begin

                       if Informa
                          then begin
                                  WriteLn (Acompanhamento, 'Loader inicio' );
                                  Delay ( TempoEspera );
                                end;

                       Executou := TRUE;

                       Tab_Loader^.TabPag    := Tab_Loader^.Pag;
                       Fila_BCP [2]^.EndTabPag       :=Tab_Loader^.Pag;
                       Pag_Loader^.TabPag    ;= Tab_Loader^.Pag;
                       Pag_Loader^.PagLog    := 0; { 1ª pag codigo }

                       { atualiza tab pag com o Endereco da pag 0 }
                       Memória [ Tab_Loader^.Pag, 0 ].Pag := Pag_Loader^.Pag;

                       for I := 0 to 7 {inicializa tabela de paginas }
                             do With Memoria [Fila_BCP [2]^.EndTabPag, I ] do
                                   begin
```

```
                                        TIPO  := 'T';
                                        CAMPO1      := 1; {CAMPO1 = 1 : paginas
                                    ausentes }
                                    End;

                            { Ativa disco para leitura de pagina }
                            COMANDA ( 3, 3 );
                        End

                else begin
                        if Tab_Loader <> nil
                                then begin
                                        Insere_Pag_Mem ( Tab_Loader, Fila_Mem_Livre
                                );
                                        Tab_Loader :=NIL;
                                end;
                        if Pag_Loader <> nil
                                then begin
                        Insere_Pag_Mem ( Pag_Loader, Fila_Mem_Livre );
                        Pag_Loader := NIL;
end;
end;
end;

End; { Loader_Inicio}
procedure Loader_Fim;
var NumPag : integer;
Begin
if ProcSimples [6] { 2 partes do processo assinalada }
then Begin

if Informa
then begin
WriteLn ( Acompanhamento, 'Loader fim' );
Delay ( TempoEspera );
end;

Executou := true;
ProcSimples [6] := false;

{ insere o pont da tab na fica de pág não substitutiveis }
Insere_Pág_Mem ( Tab_Loader, Fila_Não_Substitutiveis );
Tab_Loader := nil;

{ insere o pont da pag 0 na fila de pag substitutiveis }
Insere_Pag_Mem ( Pág_Loader, Fila_Substitutiveis );
Pag_Loader := nil;
```

```
With Fila_BCP [2]^ do
begin
{ carrega pag do Disco na Memoria }
Transf_Pag_Prog_Mem ( InicProg,EndTabPag,FP);

Memória [ EndTabPag, 0 ]. Campo1 := 0; { presente }
end;
```

109

```
Fila_BCP [ 2 ]^.TS := TS_Padrão;

{ insere job na fila de Prog prontos p/ execução }
Troca_Fila (2,3);
End;
End; { Loader_Fim }

End. {Loader}
```

110

```
Unit Paginac;
Interface
Uses Crt, Global, B_C_P, Memory;
Procedure Paginacao_Inicio;
Procedure Paginacao_Fim;

Implementation

{carrega na memoria a pagina em falta do programa do usuario}
procedure Paginação_Inicio;
begin

if ( RC.Disco = o ) and       { Disco disponivel }
(not Fila_Vazia_BCP ( 6 ) )          { Programa aguardando paginacao }

then
begin
Aloca_Pag_Mem ( Nova_Pag );

if Nova_Pag <> Nil
then begin
```

```
if informa
then begin
WriteLn ( Acompanhamento, 'Paginacao inicio' );
Delay ( TempoEspera );
end;
Executou := true;
Nova_Pag^.TabPag := Fila_BCP [6]^.EndTabPag;
Nova_Pag^.PagLog := Fila_BCP [6]^.FP;

{ atualiza tab pagina com o endereco da nova pagina }
With Fila_BCP [6]^
do Memoria [ EndTabPag, FP ].Pag := Nova_Pag^.Pag;

{ Ativa disco para leitura de pagina }
Comanda ( 3,4 );
end;
end; { then }
end; {Paginacao_Inicio }
```

111

```
procedure Paginacao_Fim;
begin

if ProcSimples [5]
then begin

if Informa
then begin
WriteLn ( Acompanhamento, 'Paginacao fim' );
Delay ( TempoEspera );
end;

Executou := true;
ProcSimples [5] := false;

If Tipo_Pag_Falta = 'S' { aloca nas substituiveis }
then Insere_Pag_Mem ( Nova_Pag, Fila_Substituíveis )
else Insere_Pag_Mem (Nova_Pag, Fila_Nao_Substituiveis );

Nova_Pag := nil;

with Fila_BCP [6]^
do Transf_Pag_Prog_Mem ( InicProg,EndTabPag,FP );
{ atualiza presenca da pagina }
```

```
With Fila_BCP [6]^
d Memoria [ EndTabPag, FP ].Campo1 := 0;

{ insere job na fila dos pontos }
Troca_Fila ( 6,3 );

end;

end; { Paginacao_Fim }

end. {Paginac }
```

112

```
Unit In_Out;

Interface

Uses Crt, Global, B_C_P, Memory, Disk;

Procedure EsUsuario_Inicio;
Procedure EsUsuario_Fim;

Implementation

{ executa a leitura ou impressao de paginas do programa do usuario }
procedure EsUsuario_Inicio;
bejín

if ( RC.Disco = 0) and        { Disco disponivel }
( not Fila_Vazia_BCP ( 5 ) )        { Programa aguardando E/S }

then begin

if Informa
then begin
WriteLn ( Acompanhamento, 'E/S ususario inicio' );
Delay ( TempoEspera );
end;

Executou := true;

{ Ativa disco para leitura de pagina }
Comanda ( 3,5 );
```

```
end;

end; {ESUsuario_Inicio }

procedure ESUsuario_Fim;
var I, PAGD, PAGM : integer;
begin
if ProcSimples [7]
then begin

if Informa
then begin
WriteLn ( Acompanhamento, 'E/S usuario fim' );
Delay ( TempoEspera );
end;
```



```
Executou := true;
ProcSimples [7] := false;

if Fila_BCP [5]^.CC [6]      {Operacao RD : le do disco }
then begin

PAGD := Fila_BCP [5]^.CorrDados^.PAG;
Fila_BCP [5]^.CorrDados:= Fila_BCP [5]^.CorrDados^.Pro (CORTADO)

{carrega conteúdo da pag do disco na pag da memoria}
With Fila_BCP [5]^
do PagM := Memoria [ EndTabPag, FP ].Pag;

for I := 0 to 7
do with Memoria [ PagM, I ] do
begin
Tipo := 'D';
Campo1 := Disco [PagD, I ]. Campo1;
end;
end

else if Fila_BCP [5]^.CC [7]        { Op PRN : escreve no disco }
then begin

PagD := Fila_BCP [5]^.CorrImpr^.Pag;
Fila_BCP [5]^.CorrImpr := Fila_BCP [5]^.
                                                CorrImpr^.Prox;
```

```
{ carrega conteúdo da pag disco na pag memoria }
With Fila_BCP [5]^
        do Pagm := Memoria [ EndTabPag, FP].Pag;

for I := 0 to 7
do with Disco [ PagD,I ]
do
begin
Tipo := Memoria [ PAGM,I ].Tipo;
Operação := Memoria [ PAGM,I ].Operação;
Campo1 := Memoria [ PAGM,I ].Campo1;
Pag := Memoria [PAGM,I ].Pag;
Desl := Memoria [ PAGM,I ].Desl
end;
end;

{ insere job na fila dos prontos }
Troca_Fila ( 5,3 );
end;
end; { ESUsuario_Fim }

end.
```

114

```
Unit Sp_Out;
Interface
Uses Crt, Global, Buffer, B_C_P, Disk;

{carrega as paginas do programa do usuario em Buffers que }
{ serão inseridos na fila de Buffers para a impressao }
Procedure Spool_Out_Inicio;

Procedure Spool_Out_Fim;

Implementation

Procedure Spool_Out_Inicio;
begin

if ( RC.Disco = 0 ) and
( ( not Fila_Vazia_BCP ( 7 ) ) or ( not Fila_Vazia?_BCP ( 8 ) ) ) and
( not Fila_Vazia_Buffer ( Fila_Buffer_Livre ) )

then begin
```

```
if informa

then begin
WriteLn ( Acompanhamento, 'Spool Out inicio' );
Delay ( TempoEspera );
end;

Executou := true;
if Fila_Vazia_BCP ( 8 )      { ainda nao existe job em SPOOL_OUT }
then Troca_Fila ( 7, 8 );

{ Aloca um Buffer livre }
BufSaida := Pega_Buffer ( Fila_Buffer_Livre );
NumBuffLivre := NumBuffLivre – 1;

{ Ativa disco para leitura de pagina }
Comanda ( 3, 2 );

end;

end; { Spool_Out_Inicio }
```

115

```
{ procedimento que preenche o Buffer de impressao com a pagina do disco }
procedure Preencher_Buffer ( PagD : integer );
var I : integer;
begin

for I:= 0 to 7
do with BufSaida^ do
begin
Linha [I].Tipo          := Disco [PagD,I].Tipo;
Linha [I].Operacao  := Disco [PagD,I].Operacao;
Linha [I].Campo1    := Disco [PagD,I].Campo1;
Linha [I].Pag           := Disco [PagD,I]. Pag;
Linha [I].Desl          := Disco [PagD,I]. Desl
end;
end; { Preencher_Buffer }

Procedure Spool_Out_Fim;
var P3, PontAux : Pont3;
P1 : Pont1;
I: integer;
begin

if ProcSimples [4]    { 2 parte do processo assinalada }
```

```
then
begin

if Informa
then begin
WriteLn ( Acompanhamento, 'Spool Out fim' );
Delay ( TempoEspera );
end;

Executou := true;
ProcSimples [4] := false;

P3 := nil;

{ identificador do job em SPOOL_OUT }
BufSaida^.N := Fila_BCP [8]^.Identif;

if not Fila_Vazia_Disco ( Fila_BCP [8]^.InicProg )
then begin    { preenche o Buffer com pagina de programa }

P3 := Fila_BCP [8]^.InicProg;
Fila_BCP [8]^.InicProg := P3^.Prox;
BufSaida^.Tipo := 'C';
end
```



```
else begin
if not Fila_Vazia_Disco ( Fila_BCP [8]^.InicDados )
then begin    { preenche o Buffer com pagina de dados }

P3 := Fila_BCP [8]^.InicDados;
Fila_BCP [8]^.InicDados := P3^.Prox;
BufSaida^.Tipo := 'D';
end
else begin
if Fila_BCP [8]^.InicImpr <> Fila_BCP [8]^.CorrImpr
then begin { preenche Buffer com pagina de impressao }

P3 := Fila_BCP [8]^.InicImpr;
Fila_BCP [8]^.InicImpre := P3^.Prox;
BufSaida^.Tipo := 'I';
end

else begin { libera paginas impressao nao utilizadas }

P3 := Fila_BCP [8]^.InicImpr;
```

```
while P3 <> nil do
begin
PontAux := P3^.Prox;
Insere_Pag_Disco ( P3, Fila_Disco_Livre );
P3 := PontAux;
Num_Disco_Livre := Num_Disco_Livre + 1;
end;

{ preenche Buffer de mensagem }
BufSaida^.Tipo := 'M';
for I := 0 to 2
do if Fila_BCP [8]^.CC [I]
then BufSaida^.Linha [I].Campo1:= 1
else BufSaida^.Linha [I].Campo1:= 0;
if Fila_BCP [8]^.CC [4]
then BufSaida^.Linha [3].Campo1 := 1
else BufSaida^.Linha [3].Campo1 := 0;
if Fila_BCP [8]^.CC [8]
then BufSaida^.Linha [4].Campo1 := 1
else BufSaida^.Linha [4].Campo1 := 0;
if Fila_BCP [8]^.CC [12]
then BufSaida^.Linha [5].Campo1 := 1
else BufSaida^.Linha [5].Campo1 := 0;

Libera_BCP (8);

end;   { else }
end;   { else }
end;   { else }
```

Pag.116

```
if P3 <> nil
        then benig
                Preencher_Buffer ( P3^.PAG );

                if not Fila_Vazia_BCP ( 8 )
                  then With Fila_BCP [8]^
                        do Libera_Pag_Disco ( P3 )
                  else begin
                        P1 := Fila_BCP [7];
                        While P1^.Lig <> nil
                             Do P1 := P1.lig;
                        Libera_Pag_Disco ( P3 )
                        end;
        end;
```

```
      { insere o Buffer na fila de Buffers p/ SPOOL_OUT }

       BufSaida := nil;
     end; { then }
  end;   { Spool_Out_Fim }
end.   { Sp_Out }
```



```
Unit Impress;

Interface

Uses Crt, Global, Buffer;

        Procedure Impressao_Inicio;
        Procedure Impressao Fim;

Var
        Estado_Impressao : 1..4;  { Estados do automato de impressao de buffers }
```

```
Implementation

{ Imprime o conteudo dos Buffers da fila de impressao }
procedure Impressao_Inicio;
  begin

        if ( RC.Impresso = 0) and                          { Disco disponivel }
          (  not Fila_Vazia_Buffer (Fila_Buffer_Impr) ) { Buffer para impressao }

          the begin

                if Informa
                        then begin
                                WriteLn ( Acompanhamento, 'Impressao inicio' );
                                Delay( TempoEspera );
                           end;

                Executou := true;

                { pega um Buffer da fila de Buffers p/ SPOOL_OUT }
                BufImpr := Pega_Buffer ( Fila_Buffer_Impr );

                { Ativa impressora para impressao dos dados do buffer }
            end;
end; { Impressao_Inicio }
```



```
{ procedimento que Imprime um Buffer da fila de impressao }
procedure Imprime_Buffer;
  var I : integer;
begin
```

```
 case BufImpr^.Tipo of

   'C' : begin{ pagina de codigo }

        if Estado_Impressao = 1    { 1a pag de código }
                then begin
                        writeln ( Saida);
                        writeln ( Saida, ' *****        Impressao de Job    ***** '  );
                        writeln ( Saida);
                        writeln ( Saida,  '             Job Numero : ' ,BufImpr^.N );
                        writeln ( Saida);
                        writeln ( Saida,  '             Inicio do programa ' );
                        Estado_Impressao := 2;
                   end;  { then }

                writeln ( Saida );
                for I := 0 to 7 do
                    with BufImpr^.LINHA [I]
                        do writeln ( Saida, ' ' : 8, OPERACAO : 3, ' ',
                                        PAG : 5, " ", DESL : 5 );
                end;

'D' : begin    { pagina de dados }

                if Estado_Impressao = 2    { 1a pag de dados }
                        then begin
                                writeln ( Saida);
                                 writeln ( Saida, 'Pagina de dados' : 24 );
                                Estado_Impressao := 3;
                              end;      { then }

                writeln ( Saida );
                for I := 0 to 7
                        do writeln ( Saida, '' : 15, BufImpr^.LINHA [I].CAMPO1 );
                end;
 'I' : begin    { pagina de impressao }

        if ( Estado_Impressao = 2 ) or ( Estado_Impressao = 3 )
                then begin    { 1a pág impr }
                        writeln ( Saida);
                        writeln ( Saida, 'Paginas de Impressao' : 26 );
                        Estado_Impressao := 4;
        end;   { then }

        writeln ( Saída );
        for I := 0 to 7
           do writeln ( Saida, '' : 15, BufImpr^.LINHA [I].CAMPO1 );
```

```
    end;

'M':    { pagina de mensagem}

        Estado_Impressão:= 1;
        writeln ( Saida );
        writeln ( Saida, 'Impressao de Mensagem': 27);
        writeln ( Saida );
        writeln ( Saida, 'Interrupcao Ocorrida': 27);
        writeln( Saida );
        with BufImpr^ do
          if LINHA [0].CAMPO1 = 1;
                then writeln ( Saida, 'Protecao de Memoria': 25 )

          else if LINHA [1].CAMPO1 = 1
                 then writeln ( Saida, 'Codigo de Operacao Invalido': 29 )

                 else if LINHA[2].CAMPO1 = 1
                       then writeln ( Saida, 'Overflow': 20 )

                       else if LINHA [3].CAMPO1 = 1
                                 then writeln ( Saida, 'Timer': 17 )

                                 else if LINHA [4].CAMPO1 = 1
                                         then writeln ( Saida. 'Fim de Programa': ?(não
                esta terminado ) )

                                         else if LINHA [1].CAMPO1 = 1
                                                 then writeln ( Saida. 'Erro de Entrada': ? );

        writeln ( Saida );
        writeln ( Saida, 'Job Nun: ': 19, BufImpr^.N );
        writeln ( Saida );
        writeln ( Saida );
       end { M }

   end { case };

procedure Impressao_Fim;
  begin

   if ProcSimples [2]          { 2 parte do processo assinalada }
     then begin
```

```
          if Informa
            then begin
                    WriteLn ( Acompanhamento, 'Impressao fim' );
                    Delay ( TempoEspera );
                 end;

          Executou:= true;
          ProcSimples [2]:= false;
          Imprime_Buffer;
          Libera_Buffer ( BufImpr );

        end;

 end;  { Impressão_Fim }

begin

     Estado_Impressão:= 1;
End.   { Impress }
```

```
Unit Processos_Hardware;

Interface

        Uses Crt, Global, B_C_P, Memory, L_R_U,
                Leitura, Sp_In, Loader, Paginac, In_Out, Sp_Out, Impress;

        Procedure Supervisor;

Implementation

function Testa_Int : integer;
 var I : integer;
 begin

        I := -1;
        repeat
                Inc ( I );
         until CC [I] or ( I = 12 );

        if CC [I]
                then Testa_Int := I
                else Testa_Int := -1;

 end;  { Testa_Int }

{ verifica se ja se cumpriu o tempo de simulacao dos Perifericos }
procedure Simula_Periféricos;
        var I : integer;
        begin

                if RC.Leitora <> 0
                   then CC [9] := T [1] <= CK;
                if RC.Impressao <> 0
```

```
            then CC[10] := T[2] <= CK;
          if RC.Impressao <> 0
            then CC[11] := T[3] <= CK;

    end;   { Simula_Perifericos }
```

Pag122

```
{ proc que Atualiza o valor do Clock e verifica estouro }
procedure Atualiza_Clock;
    var I : integer;
    begin

        CK := CK + 1;

        If CK >= 30000;
          then begin
              CK := CK – 30000;
              for I := 1 to 3
                  do T [I] := T [I] – 30000;
              end;
        end;   { Atualiza_Clock }

{ verifica o tempo de Perifericos e de Execucao de Programa }
procedure Testa_Tempo_Periféricos;
    begin

        Simula_Periféricos;
        CC [5] := TS <= 0;
        CC [4] := Timer <= 0;
        Atualiza_Clock;

    end;   { Testa_Tempo_Periféricos }

{ Simula a Espera do SOMAT ate que ocorra alguma Interrupção }
procedure Espera;
```

```
        var cont : integer;
        begin

                if Informa
                  then begin
                                WriteLn ( Aconmpanhamento, ‘ Espera ‘ );
                                Delay ( TempoEspera );
                        end;
```

Pag123

```
Cont := 0;
Repeat

        Simula_Perifericos;
        Interripcao := Testa_Int;
        If Interrupcao = -1
          then Atualiza_Clock;
        cont := cont + 1;

    until ( Interrupcao <> -1 ) or ( cont > 41 );

    if ( not ForEver );
     then ForEver := Cont > 41;

  end;          { Espera }

{ Decrementa o valor do PC }
procedure Decrementa_PC;
        begin

                CP.Desl := (CP.Desl + 7 ) mod 8;
                if CP.Desl = 7
```

```
            then CP.Pag := CP.Pag – 1;
        end;   { Decrementa_PC }

{ Incrementa o valor do PC }
procedure Incrementa_PC;
        begin

            CP.Desl := (CP.Desl + 1 ) mod 8;
            if CP.Desl = 0
              then CP.Pag := CP.Pag + 1;
        end;   { Incrementa_PC }

{ faz a busca da instrução apontada pelo contador de Programa }
procedure Fetch;
        var I, pagina : integer;
        begin

            if Informa
              then begin
                    WriteLn ( Acompanhamento, 'Fetch : Busca de instrucao' );
                    Delay ( TempoEspera );
                  end;
```

Pág 124

```
If ( CP.Pág >= TP.TAM ) or ( CP.Pag < 0 )       { protecao de memoria }
  then CC [0] := true

  else begin
            { endereco da pag que contem a instrucao }
            PpagEf := Memória [ TP.Pag, CP.Pág ];

            if PpagEf.Campo1 = 1       { pag ausente }
              then begin
                        FP := CP.Pag;                { nro da pag ausente }
                        TIPO_Pág_FALTA := 'S';  { pág substituível }
                        CC [3] := true;
                   End
            Else begin    { pag presente }
                   INSERE_LRU ( PPagEf.Pag );
```

```
                    { le nova instrução }
                    RI :=Memoria [ PPagEf.Pag, CP.Desl ];

                    If ( RI.Campo1 > 8 ) or ( RI.Campo1 < 0 )
                      then CC [1] := true          { cod Operacao invalido }
                      else Incrementa_PC;
                    end;
              end;

  end;          { Fetch }

{ Executa uma Operacao do Programa do Usuario }
procedure Executa_Operação;
        var AUX : integer;
        begin

              case RI.Campo1 of
                    0 : begin       { pare }
                          CC [8] := true;

                          Atualiza_Clock;
                          Dec ( Timer );
                          Dec ( TS );
                        end;

                    1: begin        { leia }
                          if Fila_BCP [4]^.CORRDADOS = nil       { nao ha + pag
de dados }

                            then CC [0] := true;

                            else begin
                                  CC [6] := true;
                                  FP := RI.Pag;
                               end;

pag 125
        Atualiza_Clock;
        Dec ( Timer );
        Dec ( TS );
      end;

2 : begin       { imprima }
        if Fila_BCP [4]^.CORRIMPR = nil          { nao ha + pag impressao }
          then CC [0] := true

          else begin
```

```
                    CC [7] := RI.Pag;
                End;
          Atualiza_Clock;
          Dec ( Timer );
          Dec ( TS );
        End;

3 : begin       { load }

        ACC := Memoria [ PPagOp.Pag, RI.Desl ].Campo1;

                Atualiza_Clock;
                Atualiza_Clock;
                Dec ( Timer );
                Dec ( Timer );
                Dec ( TS );
                Dec ( TS );
              end;

4 : begin       { store }

        Memoria  [ PPagOp.Pag, RI.Desl ].Campo1 := ACC;

                Atualiza_Clock;
                Atualiza_Clock;
                Dec ( Timer );
                Dec ( Timer );
                Dec ( TS );
                Dec ( TS );
      end;

5 : begin       { sub }
        AUX := Memoria [ PPagOp.Pag, RI.Desl ].campo1;

        if (( ACC > 0 ) and ( AUX < 0 )) or (( ACC < 0 ) and ( AUX > 0 ))
          Then CC [2] := maxint – ABS ( ACC ) < ABS ( AUX );

        if not CC [2]
          then ACC := ACC – AUX;

Atualiza_Clock;
Atualiza_Clock;
Dec ( Timer );
Dec ( Timer );
Dec ( TS );
Dec ( TS );
end;
```

```
6  : begin       {  som  }

          AUX  := Memoria    [  PpagOp . Pag , RI . Desl  ] . Campo;

          If  ((ACC > 0 ) and (AUX > 0)) or ((ACC < 0 ) AND (AUX < 0 );
              Then  ACC[2] := maxint  -  ABS  (ACC)  <  ABS  (AUX);

          If  not  CC [2]
               Then  ACC  := ACC  + AUX ;

          Atualiza_Clock ;
          Atuliza _Clock ;
          Dec  ( Timer );
          Dec  ( Timer );
          Dec  ( TS );
          Dec  ( TS );
      end ;

7  :  begin        {   jump   }

          CP. Pag  :=  RI . Pag;
          CP . Desl  :=  RI . Desl ;

           Atualiza_Clock;
           Dec  ( Timer  );
           Dec  ( TS );
        end ;

8  :  begin        {   jump  neg   }

          if   ACC  <  0
               then    bengin
                              CP . Pag  : = RI . Pag;
                              CP . Desl : = RI . Desl;
          end ;
          Atualiza_Clock;
          Dec   (  Timer   ) ;
          Dec   (  TS   );
       end ;
end    {   case   }

end;  {  Executa_Operacao  }
{ proc que verifica a validade de uma instrucao e a executa }
procedure Execucao ;
   begin
```

```
      if  informa
            the  bengin
                  WritLn  (Acompanhamento,  'Execucao de instrucao'  );
                  Delay (   TempoEspera   );
              End ;

       If  RI . Pag  >= TP. TAM          {  protecao de memoria  }
          Then CC [0]  :=  true

          Else begin

                If not  (RI . Campo1 in  [ 0, 7, 8 ] )
                   Not  begin

                        { pag referida pela instrucao }
                         PpagOp   := Memoria [TP. Pag, RI . Pag ];

                         If PPagOp . Campo1  =  {  pag ausente  }
                            then bengin
                                    CC [3] :  = true;    {falta pag }
                                     Decrementa_PC;
                                      FP := RI . Pag ;  { pag refirida p/ instr }

                                       If RI . Campo1 = 1   { Operacao  leia  }
                                          then  TIPO_Pag_FALTA ; = ´ N ´
                                          else   TIPO_Pag_FALTA := ´ S ´
                                    end
                              else bengin
                                       INSERE_LRU  (PpagOp . Pag);
                                       EXECUTA_Operacao;
                                    end;

                                end { then }

                        else EXECUTA_Operacao;
                  end;

end ;  { execucao }
{ simulacao do funcionamento do hardware }
procedimento hardwre ;
   begin
      if informa
        then begin
                WriteLn   ( Acompanhamento, ´Hardwre´ )
                 Delay   (TempoEpera  );
             End;
Repeat
```

```
   FETCH;                                    {   busca de instrucao   }
   interrupcao := TESTA_INT;                 {   Verifica se houve interrupcao   }

   if  interrupcao = -1
     Then begin
                Execucao;                     {executa instrucao}
                Testa_Tempo_Perifericos;
                 Interrupcao : = Testa_Int
           End

    Until interrupcao <> -1

End ; { hardware }

{ proc que Escala um programa p/ Execucao }
procedure  Escala_Prog_Usuario;
      begin

          if informa
            then begin
                    writeLn ( Acampanhamento , 'Escala programa de usuario');
                     Delay (TempoEspera);
                end ;

Troca_Fila (3,4); {insere na fila de prog em Execucao}

Timer       : = Fila_BCP [4]^.Timer;
ACC         : = Fila_BCP [4]^.ACC;
TS          : = Fila_BCP [4]^.TS;
CP.Pag      : = Fila_BCP[4]^.CP.Pag;
CP.Desl,    : = Fila_BCP[4]^.CP.Desl;
TP.Tam      : = Fila_BCP[4]^.EndTabPag;
TP.Tam      : = Fila_BCP[4]^.TamProg;

End; {Escala_Prog_Usuario}

{ salva o estado do hardware no BCP do processo interrompido}
procedure Salva_Hard_BCP;
     Var I : integer;
      Begin

          If Informa
            The begin
                    WriteLn (Acompanhmento, 'Salva conteudo do Hardware');
                     Delay (TempoEspera);
```

```
                    end;

Fila_BCP  [4] ^. Pag : =. Pag;
Fila_BCP  [4] ^. Desl : = CP . Desl ;
Fila_BCP  [4] ^. Timer : = Timer ;
Fila_BCP  [4] ^. ACC : = ACC;
Fila_BCP  [4] ^. FP : = FP ;
Fila_BCP  [4] ^. TS : = TS;

for  I : =0 to 12
    do Fila_BCP [4]^.CC [ I ];

end ; {Salva_Hard_BCP}
{ pro que trata as interrupcoes assinaladas no vetor CC e Atualiza }
{ a fila das paginas substituiveis pelo criterio do LRU                    }
procedure Trata_Interrupcao;
   var informa
   begin

      if  informa
         then begin
                writeLn (Acompanhamento , 'Trata interrupcao);
                Delay (TempoEspera );
             end ;
       if not Fila_Vazia_BCP (4) {havia prog em Execucao };
         then Salva_Hard_BCP;

       interrupcao :=Testa_Int;

       while interrupcao <> -1 do
          begin

             case interrupcao of
                0,2,3,4,8 : begin  {int fatais : ProtMem, OpInV, Overf, Timer, Pare }
                              for I : = 0 to 8
                                 do CC [ I ] : false;

                              { Libera pag de memoria utilizadas}
                              with Fila_BCP [4]^
                                 do Libera_pag_Mem (End TabPag);

                              { insere Prog na fila dos acabados }
                              Troca_Fila (4,7);
                            end ;

                3 : begin { int por falsa pagina }
                       CC [33] : = false ;
```

```
                    CC {5}  : =  false ; {timer slice }

                     { insere prog na fila de paginacao }
                    Troca_Fila (4,6);
                end ;
5 : begin                 { int por time-slice }
      CC [5]  : = false ;
      Fila_BCP [ 4 ]^. TS : = TS_Padao;
      {  insere Prog na fila dos prontos  }
     Troca_Fila (4,3);
  end;

6,7 : begin {int de I/O pelo Usuario }
        CC [Interrupcao ] : = false ;
        CC  [ 5 ] : = false;
        {insere programa na fila de E/S}
        Troca_Fila  ( 4,5 );
    end ;

9,10.11; begin {int Perifericos }
           if not Fila_Vazia_BCP ( 4 )
             then  begin { havia programa sendo executado }
                     if TS < 5 {tempo de processamento do prog .}
                        then begin {limite de tempo ultrapassado }
                                Fila_BCP [ 4 ] ^. TS : =TS_Padrao;
                                Troca_Fila ( 4,3 )
                             end
                        else insere_comeco;
                   end;

           Case interrupcao of
             9 : begin {leitora }
                    RC . LEITURA := 0;
                    ProcSimples [ 1 ] : =true ;
                 end ;
            10 : begin  {IMPRESSORA }
                    RC . IMPRESSO : = 0 ;
                    ProcSimples  [ 2 ]  : = true ;
                 end ;
            11 : begin  { disco }
                    Case RC . DISCO of
                       1  : {  disco usado p/ sp_in : ativa sp_in fim  }
                            ProcSimples  [ 3 ] : = true ;
                       2  :   {  disco usado p/ sp_out : ativa sp_out fim  }
                              ProcSimples [ 4 ] : = true ;
                       3   : { disco usado p/ loader  : ativa loader fim  }
                              ProcSimples [ 6 ] : = true;
```

```
                        4   :   { usado p/ paginacao : ativa paginacao fim }
                              ProcSimples [ 5 ] : = true ;
                        5   : {  usado p/ ESUsuario fim };
                              ProcSimples [ 7 ] : = true ;

                      end;


                            RC . DISCO : = 0 ;
                          end ;
                        end;

                        CC   [Interrupcao ] : = false ;
                      end ;
        end ; { case }

        Interrupcao : = Testa_Int ;

    end ; { while }

Atualiza_LRU;

end ; {Trata_Interrupcao }

{ procedimento que realiza os Processos_Simples }
procedure processos_Simples ;
    begin
        if informa
          then begin
                  writeLn  ( Acompanhamento , 'Processos Simpres' );
                  Delay   (  TempoEspera   );
            end ;

    Repeat
        Executou : = false ;

        {  segundas partes  }

        Impressora_fim;
        Spool_In_Fim ;
        Spool_Out_Fim;
        Leitura_Fim ;
        ESUsuario_Fim :
        Paginacao_Fim;

        {  primeiras partes  }
```

```
            ESUsuario_Inicio ;
            Loader_Inicio;
            Spool_Out_Inicio ;
            Spool_In_Inicio ;
            Leitura_Inicio ;
            Paginacao_Inicio ;

        Until not Executou

End ;  {Processos_Simpres }

{  Procedimento que faz a simulacao do Spervisor do SOMAT  }
procedure Supervisor ;
      begin
          if informa
          then begin
                    writeLn (Acompanhamento , 'Supervisor ' );
                    Delay (TempoEspera)
                  end;

     repeat

        Processos_Simples ;

        If  not Fila_Vazia_BCP  ( 3 )
            Then begin
                      Escala_Prog_Usuario;
                      Hardware ;
                    end
              else Espera ;

         Trata_Interrupcao;

      until Forever;

end ; { Supervisor }

end . { Processos_hardware }
```

```
        { programa principal }

        programa SistemaOperacional;

       Uses Crt, Global , ProcHard;

       Var Nome_Arquivo  :String  [ 20 ] ;
            Resposta                : Char ;

begin

    ClrScr;
    WriteLn ( 'Sistema Operacional Simplificado ' : 56  );
    Gotoxy ( 15 , 4 );
    Write ( 'Aquivo de entrada : ' );
    Readin (Nome_Aquivo );
    Assign (Entrada , Nome_Arquivo );
    Reset (Entrada );
    Gotoxy ( 15, 6 );
    Write (  'Arquivo de saida :')
    ReadLn  ( Nome_Arquivo );
    Assingn (Saida , Nome_Aquivo );
    Rewrite  (Saida );

    Gotoxy (15, 8 ) ;
    Write ( 'Desaeja acompanhamento do programa ( S/N ) ? ' ) ;
    Repeat
        Resposta   : = UpCase (ReadKey ) ;
    Until Resposta in [ 'S' , 'N' ] ;

    Informa  : ( Resposta  = 'S' );

    if informa
```

```
        Then begin
                Gotoxy  (15, 10 );
                Write  ( ‘ acompanhamento na tela ou impressora (T/I ) ? ‘ );
                Rpeat
                  Resposta : = UpCase (ReadKey );
                Until Resposta in [ ‘T’ , ‘T’ , ‘I’ ]

                if Resposta = ‘T’
                   Then  begin
                            Nome_Arquivo  : = ‘ Com ‘;
                             TempoEspera  := 100;
                         end
                    else  begin
                             Nome_Aquivo := ‘Lpt1’ ;
                         end;


                     Assingn (Acompanhamento , Nome_Aquivo );
                     Rewrite ( Acompanhamento );
                     WriteLn (Acompanhamento );
                     WriteLn ( Acompanhamento );
                   end ;

      Supervisor ;

      if Informa
         then Close (Acompanhamento );
Close (Entrada )
Close (Saida );
End .
```