Maria Filomena dos Santos Sustelo

ANEXOS DO SIMULADOR EM MATLAB DE ALGORITMOS PARALELOS EM ARQUITECTURAS HOMOGÉNEAS E HETEROGÉNEAS

Universidade do Algarve, 1998

ANEXOS DO SIMULADOR EM MATLAB DE ALGORITMOS PARALELOS EM ARQUITECTURAS HOMOGÉNEAS E HETEROGÉNEAS

Unidade de Ciências Exactas e Humanas

da

Universidade do Algarve

Campus de Gambelas - Faro - Portugal

Maria Filomena dos Santos Sustelo

Dezembro de 1998

2395 T.

ÍNDICE

ANEXOS

ANEXOS

ANEXO A

Programa Secsim

Seguem-se as listagens em Matlab que implementam o código do programa **Secsim** de inicialização da simulação do sistema constituído por *Transputers*, *C40's* ou misto:

```
% This file Secsim, inicialize the simulator of Transputer, C40, and mixed system
clear

sim_glob
%if 0 % bypass

NUM_FUN=input('how many different functions=');

NUM_PROC=0;
NUM_CHANNELS=0;

for i=1:NUM_FUN
 disp(' ');

 texto=['FUNCTION ',num2str(i) ];
 disp(texto);
 disp('-----------');
 op=input('String identifier = ');
 l=length(op);
 FUN_ID(i,1:l)=op;
 FUNCTIONS(i,5)=l;
 FUNCTIONS(i,1)=NUM_PROC+1;
 FUNCTIONS(i,2)=input('How many processes uses this function=');
 FUNCTIONS(i,3)=input('How many channels for this function=');
 FUNCTIONS(i,4)=input('High(1) or low(0) priority ? ');
 PROCESSES(NUM_PROC+1:FUNCTIONS(i,2)+NUM_PROC,2)=...
 ones(FUNCTIONS(i,2),1)*FUNCTIONS(i,4);
 PROCESSES(NUM_PROC+1:FUNCTIONS(i,2)+NUM_PROC,3)=ones(FUNCTIONS(i,2),1)*i;
 NUM_PROC=NUM_PROC+FUNCTIONS(i,2);
end

CHAN_ID=zeros(NUM_PROC,1);

disp(' ');
NUM_PROCESSORS=input(' How many processors=');
ID_TOPOLOGY=input('Transputers(1), C40s(2),Transputers and C40s(3),other processors(4)=');
ctr=0;
cc4=0;
for p=1:NUM_PROCESSORS
     if ID_TOPOLOGY==1
          FLOP_TIME=FLOP_TIME(1,1);
          EXTERNAL_VAR_COM_TIME=EXTERNAL_VAR_COM_TIME(1,1);
    ctr=ctr+1;
    tr=' ';
          tr=[tr 'trans' num2str(ctr)];
          disp('processor name is', tr);
          tam=length(tr);
          ID_PROCESSOR(p,1:tam)=tr;
          num_links=4;
           PROCESSOR(p,num_links+3)=0;
          PROCESSOR(p,1)=ID_TOPOLOGY;
```

```
        PROCESSOR(p,2)=1;

    end
    if ID_TOPOLOGY==2
    %   FLOP_TIME=FLOP_TIME(2,1);
    %   EXTERNAL_VAR_COM_TIME=EXTERNAL_VAR_COM_TIME(2,1);
    cc4=cc4+1;
    c4=' ';
        c4=[c4 'C40' num2str(cc4)];
    %   disp('processor name is',  c4);
        tam=length(c4);
        ID_PROCESSOR(p,1:tam)=c4;
        num_links=3;
        PROCESSOR(p,num_links+3)=0;
         PROCESSOR(p,1)=ID_TOPOLOGY;
        PROCESSOR(p,2)=2;
    end
    if ID_TOPOLOGY==3
        ip=input('processor name is trans(1) or C40(2)?');
        if ip==1
            ctr=ctr+1;
            tr=' ';
            tr=[tr 'trans' num2str(ctr)];
            tam=length(tr);
            ID_PROCESSOR(p,1:tam)=tr;
        end
        if ip==2
            cc4=cc4+1;
            c4=' ';
            c4=[c4 'C40' num2str(cc4)];
            tam=length(c4);
            ID_PROCESSOR(p,1:tam)=c4;
        end
        num_links=4;
      PROCESSOR(p,num_links+3)=0;
        PROCESSOR(p,1)=ID_TOPOLOGY;
        PROCESSOR(p,2)=ip;

    end
end %for p

fich=fopen('config.m','w+t');
if ID_TOPOLOGY==1
    fprintf(fich, '!Ficheiro de configuracao para transputers=1 \n');
    fprintf(fich, '!Hardware configuration \n');
    fprintf(fich, 'processor host type=pc \n');
    fprintf(fich, 'processor root type=t800 \n');
    for p=1:NUM_PROCESSORS
        tex=['processor root',num2str(p)];
        fprintf(fich, [tex ' type=t800 ' n']);
    end
    fprintf(fich, 'wire ? root[0] host[0]  n');
```

```
    fprintf(fich, 'processor root type=C40 \n');
    if (NUM_PROCESSORS > 1)
        for p=1:NUM_PROCESSORS-1
            tex=['processor wrk',num2str(p)];
            fprintf(fich, [tex ' type=c40 \n']);
    end
  end
    fprintf(fich, 'wire ? root[0]');
end
if ID_TOPOLOGY==3
    fprintf(fich, '!Ficheiro de configuracao para topologias mistas=3 \n');
  fprintf(fich, '!Hardware configuration \n');
  if PROCESSOR(1,2)==1
          fprintf(fich, 'processor host type=pc \n');
          fprintf(fich, 'processor root type=t800 \n');
    else
          fprintf(fich, 'processor root type=C40 \n');
  end
  for p=2:NUM_PROCESSORS
    tex=['processor wrk',num2str(p)];

    if PROCESSOR(p,2)==1
          fprintf(fich, [tex ' type=t800 \n']);
    else
      fprintf(fich, [tex ' type=C40 Kernel="slot?.krn" !please edit the number of slot in ?\n']);

      end
  end
  if PROCESSOR(1,2)==1
   fprintf(fich, 'wire ? root[0] host[0] \n');
  else
    fprintf(fich, 'wire ? root[0] wrk[0] \n');
  end
  for p=2:NUM_PROCESSORS
    tex1=['wrk',num2str(p)];

    if PROCESSOR(p,2)==2
          fprintf(fich, [' wire ? root[2] wrk ' tex1 ' [0] \n']);
    else
      fprintf(fich, [tex ' type=C40 Kernel="slot?.krn" !please edit the number of slot in ?'\n']);

      end
  end
end
fprintf(fich, 'Software configuration \n');
fclose(fich);

for row=1:NUM_FUN
 disp(' ');
 disp(' ');
 fun=FUN_ID(row,1:FUNCTIONS(row,5));
 disp(['Processes which execute the code ' fun])
 disp(['-----------------------------------------']);

 i=FUNCTIONS(row,1);
 for col=1:FUNCTIONS(row,2)
```

```
disp(' ');
funcomp=[fun '(' num2str(col) ') '];
texto=['In which processor is ' funcomp 'executing ? '];
PROCESSES(i,1)=input(texto);
PRIORITARIO=[];
linha=0;
for p=1:NUM_PROCESSORS
 linha=linha+1;
 AUX=find(PROCESSES(:,1)==p & PROCESSES(:,2)==1);
 lin=length(AUX);
 PRIORITARIO(1:lin,linha)=AUX;
end
disp(' ');
disp(['Channels associated with ' funcomp])
disp('--------------------------------');
chanto=[''];
chanfrom=[''];
ex=0;
for j=1:CHAN_ID(i,1)
 chan=CHAN_ID(i,j+1);
 if CHANNEL(chan,1)==i
  % processor i is the source
  proc=CHANNEL(chan,2);
  f=PROCESSES(proc,3);
  chanto=[chanto ' ' operation(proc) '(' num2str(proc+1-FUNCTIONS(f,1)) ') '];
  ex=ex+1;
 else
  % processor i is the destination
  proc=CHANNEL(chan,1);
  f=PROCESSES(proc,3);
  chanfrom=[chanfrom ' ' operation(proc) '(' num2str(proc+1-FUNCTIONS(f,1)) ') '];
  ex=ex+1;
 end
end
if length(chanto)>0
 texto=[funcomp ' is already sending to ' chanto];
 disp(texto);
end
if length(chanfrom)>0
 texto=[funcomp ' is already receiving from ' chanfrom];
 disp(texto);
end

more=FUNCTIONS(row,3)-ex;

for j=1:more
 NUM_CHANNELS=NUM_CHANNELS+1;
 CHAN_ID(i,1)=CHAN_ID(i,1)+1;
 CHAN_ID(i,CHAN_ID(i,1)+1)=NUM_CHANNELS;
 texto=['Is ' funcomp 'the source(1) or the dest.(0) '];
 texto=[texto 'of its channel numb. ' num2str(j+ex) ' ? '];
 from=input(texto);
 if from==1
  CHANNEL(NUM_CHANNELS,1)=i;
  f=input('To which code ? ');
  if FUNCTIONS(f,2)>1
```

```
      texto=['To which of the ' operation(FUNCTIONS(f,1)) ' codes ? '];
      num=input(texto);
     else
      num=1;
     end
     other=FUNCTIONS(f,1)-1+num;
     CHANNEL(NUM_CHANNELS,2)=other;
     CHAN_ID(other,1)=CHAN_ID(other,1)+1;
     CHAN_ID(other,CHAN_ID(other,1)+1)=NUM_CHANNELS;
    else
     CHANNEL(NUM_CHANNELS,2)=i;
     f=input('From which code ? ');
     if FUNCTIONS(f,2)>1
      texto=['From which of the ' operation(FUNCTIONS(f,1)) ' codes ? '];
      num=input(texto);
     else
      num=1;
     end
     other=FUNCTIONS(f,1)-1+num;
     CHANNEL(NUM_CHANNELS,1)=other;
     CHAN_ID(other,1)=CHAN_ID(other,1)+1;
     CHAN_ID(other,CHAN_ID(other,1)+1)=NUM_CHANNELS;
    end
    CHANNEL(NUM_CHANNELS,3)=0;
    CHANNEL(NUM_CHANNELS,4)=0;
    CHANNEL(NUM_CHANNELS,5)=0;
   end
   i=i+1;
  end
end

[M,N]=size(PRIORITARIO);
clear par
for i=1:N-1
  for j=i+1:N
   if (i~=j)
    for k=1:M
     for l=1:M
      par(1)=PRIORITARIO(k,i);
      par(2)=PRIORITARIO(l,j);
      par
      if ID_TOPOLOGY==1
       for CONT=1:NUM_CHANNELS
        if (CHANNEL(CONT,1)==par(1) & CHANNEL(CONT,2)==par(2))
          B=find(CHANNEL(:,1)==par(2) & CHANNEL(:,2)==par(1));
          if ~isempty(B)
           CHANNEL(B,5)=CHANNEL(B,5)+1;
          end
          if CHANNEL(CONT,5)==0
            D=find(PRIORITARIO(1,:)==par(1));
            E=find(PRIORITARIO(1,:)==par(2));
           if (~isempty(D) & ~isempty(E))
            PROCESSOR(D,3)=PROCESSOR(D,3)+1;
            PROCESSOR(E,3)=PROCESSOR(E,3)+1;
            PROCESSOR(D,PROCESSOR(D,3)+3)=E;
            PROCESSOR(E,PROCESSOR(E,3)+3)=D;
```

```
      end
    end

    if CHANNEL(CONT,5)==1
      CONT=CONT+1;
    end
  end
  if (CHANNEL(CONT,1)==par(2) & CHANNEL(CONT,2)==par(1))
    C=find(CHANNEL(:,1)==par(1) & CHANNEL(:,2)==par(2));
    if ~isempty(C)
     CHANNEL(C,5)=CHANNEL(C,5)+1;
    end
    if CHANNEL(CONT,5)==0

      D=find(PRIORITARIO(1,:)==par(1));
      E=find(PRIORITARIO(1,:)==par(2));

     if (~isempty(D) & ~isempty(E))
      PROCESSOR(D,3)=PROCESSOR(D,3)+1;
      PROCESSOR(E,3)=PROCESSOR(E,3)+1;
      PROCESSOR(D,PROCESSOR(D,3)+3)=E;
      PROCESSOR(E,PROCESSOR(E,3)+3)=D;
     end
    end

    if CHANNEL(CONT,5)==1
      CONT=CONT+1;
    end
  end

 end %for CONT

end %if
if ID_TOPOLOGY==2

 for CONT=1:NUM_CHANNELS
  if (CHANNEL(CONT,1)==par(1) & CHANNEL(CONT,2)==par(2))
    B=find(CHANNEL(:,1)==par(2) & CHANNEL(:,2)==par(1));
    if ~isempty(B)
     CHANNEL(B,5)=CHANNEL(B,5)+1;
    end
    if CHANNEL(CONT,5)==0
      D=find(PRIORITARIO(1,:)==par(1));
      E=find(PRIORITARIO(1,:)==par(2));
     if (~isempty(D) & ~isempty(E))
      PROCESSOR(D,3)=PROCESSOR(D,3)+1;
      PROCESSOR(E,3)=PROCESSOR(E,3)+1;
      PROCESSOR(D,PROCESSOR(D,3)+3)=E;
      PROCESSOR(E,PROCESSOR(E,3)+5)=D;
     end
    end

    if CHANNEL(CONT,5)==1
      CONT=CONT+1;
    end
  end
```

```
 if (CHANNEL(CONT,1)==par(2) & CHANNEL(CONT,2)==par(1))
    C=find(CHANNEL(:,1)==par(1) & CHANNEL(:,2)==par(2));
    if ~isempty(C)
     CHANNEL(C,5)=CHANNEL(C,5)+1;
    end
    if CHANNEL(CONT,5)==0

      D=find(PRIORITARIO(1,:)==par(1));
      E=find(PRIORITARIO(1,:)==par(2));
      if (~isempty(D) & ~isempty(E))
       PROCESSOR(D,3)=PROCESSOR(D,3)+1;
       PROCESSOR(E,3)=PROCESSOR(E,3)+1;
       PROCESSOR(D,PROCESSOR(D,3)+4)=E;
       PROCESSOR(E,PROCESSOR(E,3)+4)=D;

      end
    end

    if CHANNEL(CONT,5)==1
     CONT=CONT+1;

    end
 end

 end %for CONT

end %if
if ID_TOPOLOGY==3

 for CONT=1:NUM_CHANNELS
  if (CHANNEL(CONT,1)==par(1) & CHANNEL(CONT,2)==par(2))
    B=find(CHANNEL(:,1)==par(2) & CHANNEL(:,2)==par(1));
    if ~isempty(B)
     CHANNEL(B,5)=CHANNEL(B,5)+1;
    end
    if CHANNEL(CONT,5)==0
      D=find(PRIORITARIO(1,:)==par(1));
      E=find(PRIORITARIO(1,:)==par(2));
     if (~isempty(D) & ~isempty(E))
      PROCESSOR(D,3)=PROCESSOR(D,3)+1;
      PROCESSOR(E,3)=PROCESSOR(E,3)+1;
      PROCESSOR(D,PROCESSOR(D,3)+3)=E;
      PROCESSOR(E,PROCESSOR(E,3)+5)=D;
     end
    end

    if CHANNEL(CONT,5)==1
      CONT=CONT+1;
    end
 end

 if (CHANNEL(CONT,1)==par(2) & CHANNEL(CONT,2)==par(1))
    C=find(CHANNEL(:,1)==par(1) & CHANNEL(:,2)==par(2));
  . if ~isempty(C)
     CHANNEL(C,5)=CHANNEL(C,5)+1;
```

```
          end
          if CHANNEL(CONT,5)==0

            D=find(PRIORITARIO(1,:)==par(1));
            E=find(PRIORITARIO(1,:)==par(2));
            if (~isempty(D) & ~isempty(E))
             PROCESSOR(D,3)=PROCESSOR(D,3)+1;
             PROCESSOR(E,3)=PROCESSOR(E,3)+1;
             PROCESSOR(D,PROCESSOR(D,3)+4)=E;
             PROCESSOR(E,PROCESSOR(E,3)+4)=D;

             end
          end

          if CHANNEL(CONT,5)==1
           CONT=CONT+1;

          end
        end

    end %for CONT

      end %if
     end %forl
    end %for k
   end %if
  end %for j
end %for i


NUM_LOCAL_VAR=0;
NUM_LOCAL_VAR_INI=0;
VAR_ID=[''];

ln1=0;
for row=1:NUM_FUN
 ln1=ln1+1;
 disp(' ');
 disp(' ');
 fun=FUN_ID(row,1:FUNCTIONS(row,5));
 disp(['Local var associated with ' fun]);
 %fich=input('Filename to open for read local variables  - ');
 fil=[fun '.m'];
 setstr(fil);
 fid=fopen( fil,'rt');
 k=0;
 q=0;
  if fid > 0
    while (~(feof(fid)) & (~k))
      [A,COUNT]=fscanf(fid, '%s%b');
      k=strcmp(A, '%-');
      if (~k)
        pos0=ftell(fid);
      end;
    end;
    fclose(fid);
```

```
fid=fopen(fil,'rt');
fseek(fid, pos0, 'bof');
[A,COUNT]=fscanf(fid, '%*s%*b');

pos1=ftell(fid);
fclose(fid);
fid=fopen(fil,'rt');
while (~(feof(fid)) & (~q))
 [A,COUNT]=fscanf(fid, '%s%b');
 q=strcmp(A, '-%');
 if (q==0)
   pos2=ftell(fid);
 end
 if (q==1)
   pos3=ftell(fid);
 end
end
fclose(fid);

fid=fopen(fil,'rt');
fseek(fid, pos1, 'bof');
pos=pos1;
max_var=0;
for var = 1:20
 [A, COUNT]=fscanf(fid, '%s%b');
 pos=ftell(fid);
 q=strcmp(A, '-%');
 if (q==0) & (pos<=pos2)
   A=setstr(A);
   max_var=max_var+strcount(A);
   FUNCTIONS(row,6)=max_var;
 end %if


 if (pos == pos2)
    [A, COUNT]=fscanf(fid, '%*s%b');
     break;
 end

 if feof(fid)
      break;
 end

end %for var
fclose(fid);

fid=fopen(fil,'rt');
fseek(fid, pos1, 'bof');
pos=pos1;
x='';
A='';
posicao=0;

for v= 1:FUNCTIONS(row,6)
 NUM_LOCAL_VAR=NUM_LOCAL_VAR+1;
```

```
posicao=posicao+1;
[AUX, COUNT]=fscanf(fid, '%s%b');
len=length(AUX);
LOCAL_VAR(NUM_LOCAL_VAR,1)=len;
A(length(A)+1 :  length(A)+LOCAL_VAR(NUM_LOCAL_VAR,1)) = AUX;
A=setstr(A);
pos=ftell(fid);
q=strcmp(A, '-%');
if (q==0) & (pos<=pos2)

  for j=1:length(A)
       x=A;
       le=length(x);
       LOCAL_VAR(NUM_LOCAL_VAR,2)=le;
       LOCAL_VAR(NUM_LOCAL_VAR,3)=posicao;
       LOCAL_VAR(NUM_LOCAL_VAR,4)=ln1;
       LOCAL_VAR(NUM_LOCAL_VAR,5)=FUNCTIONS(row,2);
       VAR_ID(ln1,1:LOCAL_VAR(NUM_LOCAL_VAR,2))=x;


  end %for

end %if

if (pos == pos2)
   [A, COUNT]=fscanf(fid, '%*s%b');
    break;
end

if feof(fid)
     break;
end


end %for v
fclose(fid)
fid=fopen(fil,'rt');
ki=0;
qi=0;
while (~(feof(fid)) & (~ki))
  [A,COUNT]=fscanf(fid, '%s%b');
  ki=strcmp(A, '%+');
  if (~ki)
    pos0i=ftell(fid);
  end;
end;
fclose(fid);

fid=fopen(fil,'rt');
fseek(fid, pos0i, 'bof');
[A,COUNT]=fscanf(fid, '%*s%*b');

pos1i=ftell(fid);

fclose(fid);
fid=fopen(fil,'rt');
```

```
while (~(feof(fid)) & (~qi))
 [A,COUNT]=fscanf(fid, '%s%b');
 qi=strcmp(A, '+%');

 if (qi==0)
   pos2i=ftell(fid);
 end
 if (qi==1)
   pos3i=ftell(fid);
 end
end
fclose(fid);
fid=fopen(fi1,'rt');
fseek(fid, pos1i, 'bof');
posi=pos1i;
max_var_i=0;
Ai='';
xi='';
for vari = 1:20
 NUM_LOCAL_VAR_INI=NUM_LOCAL_VAR_INI+1;
 [AUX1, COUNT]=fscanf(fid, '%s%b');
 leni=length(AUX1);
 LOCAL_VAR_INI(NUM_LOCAL_VAR_INI,1)=leni;
 Ai(length(Ai)+1:length(Ai)+LOCAL_VAR_INI(NUM_LOCAL_VAR_INI,1))=AUX1;
 posi=ftell(fid);
 qi=strcmp(Ai,'+%');
 if (qi==0) & (posi<=pos2i)
   Ai=setstr(Ai);
   max_var_i=max_var_i+strcount(Ai);
   FUNCTIONS(row,7)=max_var_i;
   for j=1:length(Ai)
     xi=Ai;
     lei=length(xi);
     LOCAL_VAR_INI(NUM_LOCAL_VAR_INI,2)=lei;
     VAR_ID_INI(ln1,1:LOCAL_VAR_INI(NUM_LOCAL_VAR_INI,2))=xi;
   end %for
 end %if


 if (posi == pos2i)
    [Ai, COUNT]=fscanf(fid, '%*s%b');
     break;
 end

 if feof(fid)
      break;
 end

end %for var
fclose(fid);

end %if fid

end %for row
fclose('all')
```

```
HAS_REC_FROM=zeros(NUM_PROC,1);
EVENTS(1:NUM_PROC,1)=zeros(NUM_PROC,1);
PROCESSOR_TIME=zeros(NUM_PROCESSORS,1);

READY_HPP=zeros(NUM_PROCESSORS,1);
READY_LPP=zeros(NUM_PROCESSORS,1);

TIME=zeros(NUM_PROC,1);

%
%end %bypass


filename=input('Filename for the topology of the system - ');
cont=1;
while cont
 f=[filename '.top'];
 setstr(f);
 if exist(f)==2
  rm=input('The filename already exists in disk. Remove it (yes-1/no-0) ? ');
  if rm
   eval(['delete ' f]);
   eval(['save ' f]);
   cont=0;
  else
   filename=input('Filename for the topology of the system - ');
  end
 else
  eval(['save ' f]);
  cont=0;
 end
end


fn=input('Filename for the initialization - ');
cont=1;
while cont
 fn=[fn '_ini.m'];
 setstr(fn);
 if exist(fn)==2
  rm=input('The filename already exists in disk. Remove it (yes-1/no-0) ? ');
  if rm
   eval(['delete ' fn]);
   cont=0;
  else
   fn=input('Filename for the initialization - ');
  end
 else
  cont=0;
 end
end
```

```
fprintf(fn,'%%  This file initializes the simulation of the transputer,C40 or mixed systems\n');
fprintf(fn,'%%\n');
fprintf(fn,'%%  For each process the logical and physical channels are automatically initialized.\n');
fprintf(fn,'%%  In each process you will have to set them according to the specifications given by the
system\n');
fprintf(fn,'%%  For each process, the variable after is used to control the flow of the process\n');

fprintf(fn,'%%\n');
fprintf(fn, 'sim_glob; \n');
fprintf(fn,['load ' f ' -mat;\n']);
num_working=NUM_PROCESSORS;
fprintf(fn,['num_working =' num2str(num_working) ';\n']);
num_lines=input('How many lines=');
fprintf(fn,['num_lines =' num2str(num_lines) ';\n']);
cont=1;
while cont
 num_col=input('How many columns=');
 if rem(num_col,num_working)~=0
  disp('The number of columns must be multiple of the number of workers!');
 else
  cont=0;
 end
end
fprintf(fn,['num_col =' num2str(num_col) ';\n']);
num_col_proc=num_col/num_working;
fprintf(fn,['num_col_proc =' num2str(num_col_proc) ';\n']);
R=rand(num_lines,num_col);

%end %bypass
contagem=0;
for a=1:NUM_FUN
    nvi=VAR_ID_INI(a, 1:length(VAR_ID_INI));
    nvii=sscanf(nvi,'%s');
    la=length(nvii);
    co=0;
    q=0;
    stri='';
    for v1=1: FUNCTIONS(a,7)
     contagem=contagem+1;

     va2=nvii(LOCAL_VAR_INI(contagem,2)-
LOCAL_VAR_INI(contagem,1)+1:LOCAL_VAR_INI(contagem,2));

     leng=length(va2);
     q=findstr(va2,',');
     if q~=[]
       str=strcut(va2,',');
     else
       str=va2;
     end
     co=co+strcount(str);
     stri(co,1:length(str))=str;


     fprintf(fn,[str ';\n']);
```

```
    end

end

i=0;
range_i=[];
pass=1;
while pass<=2
 i=i+1;
 if i>NUM_FUN
  pass=pass+1;
  i=1;
 end

 if (FUNCTIONS(i,4)==1)&(pass==1)
  range_i=[range_i i];
 elseif (FUNCTIONS(i,4)~=1)&(pass==2)
  range_i=[range_i i];
 end
end


for i=range_i

 proc=FUNCTIONS(i,1);
 fprintf(fn,'%%');
 for j=1:FUNCTIONS(i,2)
  chan=1;
  fprintf(fn,'%%\n');
  fun_name=[operation(proc) '(' num2str(proc-FUNCTIONS(i,1)+1) ')'];
  fprintf(fn,['%%  Initializations for process ' fun_name '\n']);
  fprintf(fn,'%%\n');
  for k=1:NUM_FUN
   for l=1:FUNCTIONS(k,2)
    other=FUNCTIONS(k,1)+l-1;
    if other~=proc
     % see if there is a channel from this proc to the other
     p=find(CHANNEL(:,1)==proc);
     q=find(CHANNEL(p,2)==other);
     if length(q)==1
      % there is
      texto=['chan' num2str(chan) '=' num2str(p(q)) ';\n'];
      tex=['%% chan ' num2str(chan) ' has destination ' operation(other) '(' num2str(other-
FUNCTIONS(k,1)+1) ')\n'];
      fprintf(fn,tex);
      fprintf(fn,texto);
      chan=chan+1;
     end
    end
   end
  end

  for k=1:NUM_FUN
   for l=1:FUNCTIONS(k,2)
    other=FUNCTIONS(k,1)+l-1;
```

```
      if other~=proc
       % see if there is a channel from this proc to the other
       p=find(CHANNEL(:,2)==proc);
       q=find(CHANNEL(p,1)==other);
       if length(q)==1
        % there is
        texto=['chan' num2str(chan) '=' num2str(p(q)) ';\n'];
        tex=['%% chan ' num2str(chan) ' comes from ' operation(other) '(' num2str(other-
FUNCTIONS(k,1)+1) ')\n'];
        fprintf(fn,tex);
        fprintf(fn,texto);
        chan=chan+1;
       end
      end
     end
    end



    fprintf(fn,'%%\n');
    fprintf(fn,'after=0;\n');
    fprintf(fn,'%%  Here other initializations\n');
    fprintf(fn,'%%...\n');

    c='';
    for ll=1:chan-1
     c=[c 'chan' num2str(ll) ','];
    end

    nv=VAR_ID(i, 1:length(VAR_ID));
    n=sscanf(nv,'%s');
    %la=length(n);

    fprintf(fn,'%%\n');
    fprintf(fn,['save_workspace('  num2str(proc) ',' c 'after,' n ');\n']);
    fprintf(fn,['insert_event(EVENT_START_EXE,',num2str(proc) ',0);\n']);
    fprintf(fn,[operation(proc) '(' num2str(proc) ');\n']);
    proc=proc+1;
    fprintf(fn,'%%\n');
  end %forj
end %for i

for i=range_i

  proc=FUNCTIONS(i,1);
  for j=1:FUNCTIONS(i,2)
    chan=1;
    for k=1:NUM_FUN
      for l=1:FUNCTIONS(k,2)
        other=FUNCTIONS(k,1)+l-1;
        if other~=proc
         % see if there is a channel from this proc to the other
         p=find(CHANNEL(:,1)==proc);
         q=find(CHANNEL(p,2)==other);
         if length(q)==1
```

```
        % there is
        chan=chan+1;
       end
      end
     end
    end

    for k=1:NUM_FUN
     for l=1:FUNCTIONS(k,2)
      other=FUNCTIONS(k,1)+l-1;
      if other~=proc
       % see if there is a channel from this proc to the other
       p=find(CHANNEL(:,2)==proc);
       q=find(CHANNEL(p,1)==other);
       if length(q)==1
        % there is

        chan=chan+1;
       end
      end
     end
    end
    c='';
    for ll=1:chan-1
     c=[c 'chan' num2str(ll) ','];
    end

    nv=VAR_ID(i, 1:length(VAR_ID));
    n=sscanf(nv,'%s');
    proc=proc+1;
   end %for j
    fun=FUN_ID(i,1:FUNCTIONS(i,5));
    disp(['Restore local variables ' fun]);

    fi=[fun '.m'];
    setstr(fi)
    fis=fopen( fi,'r');

    fis2=fopen( 'auxiliar.m','w+t');

    k=0;
    o=0;

    if fis > 0
     while (~(feof(fis)) & (~k))
      [A,COUNT]=fscanf(fis, '%s%b'); %le cada str e branco a encontrar perc
      k=strcmp(A, '-%');
      if (~k)
         pos0=ftell(fis);
      end;

     end;
     fseek(fis, 0, 'bof');
     A='';
     contline0=0;
     contline1=0;
```

```
contline2=0;
posa=[];
while (~feof(fis))
 contline1=contline1+1;
 line2=fgetl(fis);
 posaux1=ftell(fis);
 posa(contline1,1:length(posaux1))=posaux1;
 a=findstr(line2,'wait');
 A(contline1,1:length(line2))=line2;
 if~isempty(a)
    contline0=contline0+1;
    contline2=contline2+1;
    if contline2==1
     posl1a=ftell(fis);
     pose1a=posa(contline1-1,1);
     break;
    end
 end
end
fseek(fis, posl1a, 'bof');
encend=[];
contend=0;
while (~feof(fis))
 lineend=fgetl(fis);
 encend=findstr(lineend,'end');
 if ~isempty(encend)
    contend=contend+1;
    if contend==1
     posend=ftell(fis);
     break;
    end
 end
end

fseek(fis, pose1a, 'bof');
contline4=0;
contline5=0;
u=0;
f=0;
d=0;
while (~(feof(fis)) & (~u))
[A,COUNT]=fscanf(fis, '%c%');
 u=strcmp(A, ';');
 f=strcmp(A, ',');
 d=strcmp(A, ')');
 if f==1
  contline4=contline4+1;
  if contline4==1
   posl1c=ftell(fis);
  end
  if contline4==2
   posl2c=ftell(fis);
  end
  if contline4==3
   posl3c=ftell(fis);
  end
```

```
  end
  if d==1
   contline5=contline5+1;
   if contline5==1
    posl4c=ftell(fis);
   end
  end
  if u==1
   posu=ftell(fis);
   break;
  end;
end;


fseek(fis, 0, 'bof');
contline1=0;
contline3=0;
A='';
posb=[];
while (~feof(fis))
 contline1=contline1+1;
 line1=fgetl(fis);
 posaux2=ftell(fis);
 posb(contline1,1:length(posaux2))=posaux2;
 b=findstr(line1,'send');
 A(contline1,1:length(line1))=line1;

 if ~isempty(b)
     contline0=contline0+1;
     contline3=contline3+1;
     if contline3==1
      posl1b=ftell(fis);
      pose1b=posb(contline1-1,1);
     end
     if contline3==2
      posl2b=ftell(fis);
      pose2b=posb(contline1-1,1);
     end
     if contline3==3
      posl3b=ftell(fis);
      pose3b=posb(contline1-1,1);
     end
     if contline3==4
      posl4b=ftell(fis);
      pose4b=posb(contline1-1,1);
     end

 end
end

fseek(fis, pose1b, 'bof');
e=0;
contline6=0;
while (~(feof(fis)) & (~e))
  [A,COUNT]=fscanf(fis, '%c');
   e=strcmp(A,')');
```

```
    if e==1
     contline6=contline6+1;
     if contline6==1
       pose1=ftell(fis);
     end
    end

end


fseek(fis, pos0, 'bof');
[A,COUNT]=fscanf(fis, '%s%b');%le apenas a percent.
pos10=ftell(fis);
fseek(fis, 0, 'bof');%posiciona-se inicio
A='';
contline1=0;
contline9=0;
posc=[];
while (~feof(fis))
 contline1=contline1+1;
 line5=fgetl(fis);
 poscux1=ftell(fis);
 posc(contline1,1:length(poscux1))=poscux1;
 aa=findstr(line5,'while');
 A(contline1,1:length(line5))=line5;
 if ~isempty(aa)
    contline9=contline9+1:

    if contline9==1
     posl1aa=ftell(fis);
     pose1aa=posc(contline1-1,1);
     break;
    end
 end
end
fseek(fis, 0, 'bof');%posiciona-se inicio
A='';
contline1=0;
contline10=0;
posd=[];
while (~feof(fis))
 line6=fscanf(fis, '%s%b');
 posdux1=ftell(fis);

 bb=strcmp(line6,'end');

 if bb==1
    contline10=contline10+1;
    if contline10==1
     posl1bb=ftell(fis);
    end
    if contline10==2
     posl1bbb=ftell(fis);
    end
    if contline10==3
     posl1bbbb=ftell(fis);
```

```
    end
  end

end
fseek(fis, 0, 'bof');
[A,COUNT]=fread(fis, pos10, 'char');
fprintf(fis2, '%s', A);
fprintf(fis2, '\n');
fprintf(fis2,['[' c  'after,'  n ']=restore_workspace(proc_numb);']);
fprintf(fis2, '\n');
fclose(fis2);

fis2=fopen( 'auxiliar.m','r');
r=[];
contline=0;
while (~feof(fis2) & isempty(r))
  contline=contline+1;
  line3=fgetl(fis2);
  r=findstr(line3,'[');
end
[x,line3]=strcut(line3,'[');
[x,line3]=strcut(line3,']');
[X,line2]=strcut(line2,'(');
[X,line2]=strcut(line2,')');
[line2,X]=strcut(X,',');
[line2,X]=strcut(X,',');
if line2==[]
  line2=X;
end
nvr=0;
y2='';
for nrestore=1:FUNCTIONS(i,3)+1+FUNCTIONS(i,6)
  s=findstr(x,',');
  [y,x]=strcut(x,',');

  nvr=nvr+strcount(y)

  W=strcmp(line2,y);
  if (W==1)
    disp(nvr);
    break;
  end


end

fclose(fis2);
fis2=fopen( 'auxiliar.m','a+t');

fprintf(fis2,['nvr=' num2str(nvr) ';\n']);
fprintf(fis2, 'ff=flops;');
if (FUNCTIONS(i,4)==0)

   if (FUNCTIONS(i,3)==1)
    [A,COUNT]=fread(fis,pose1a-pos10,'char');
    fprintf(fis2, '%c', A);
```

```
 fprintf(fis2, '  ');
 fprintf(fis2,['if after==0 \n']);
 fprintf(fis2, '      ');
 fprintf(fis2, ['after=1; \n']);
 fprintf(fis2, '  ');
 fprintf(fis2, ['ffafter=flops; \n']);
 fprintf(fis2, '      ');
 fprintf(fis2,['save_workspace(proc_numb,' c 'after.' n ');\n']);
 [A,COUNT]=fread(fis,posl1c-pose1a,'char');
 fprintf(fis2, '%c', A);
 [A,COUNT]=fread(fis,posl4c-posl1c,'char');
 fprintf(fis2, [',ffafter-ff,nvr)']);
 [A,COUNT]=fread(fis,posl1a-posl4c,'char');
 fprintf(fis2, '%c', A);
 fprintf(fis2, '      ');
 fprintf(fis2, 'return; \n');
 fprintf(fis2, '  ');
 fprintf(fis2, ['else \n']);
 fprintf(fis2, '    ');
 fprintf(fis2, ['after=0; \n']);
 [A,COUNT]=fread(fis,posl1bb-3-posl1a,'char');
 fprintf(fis2, '%c', A);
 fprintf(fis2,'\n');
 fprintf(fis2, '  ');
 fprintf(fis2, ['end \n']);
 pis=posl1bb-3;
 [A,COUNT]=fread(fis,posl1bb-pis,'char');
 fprintf(fis2, '%c', A);
 fprintf(fis2,'\n');
 fprintf(fis2, ['ffafter=flops; \n']);
 fprintf(fis2,['save_workspace(proc_numb,' c 'after.' n ');\n']);
 fprintf(fis2,['finish(proc_numb,flops-ff);\n']);
end

if (FUNCTIONS(i,3)==2)
 [A,COUNT]=fread(fis,posl1aa-pos10,'char');
 fprintf(fis2, '%c', A);
 fprintf(fis2, '  ');
 fprintf(fis2,['if after==0 \n']);
 [A,COUNT]=fread(fis,pose1b-posl1aa,'char');
 fprintf(fis2, '%c', A);
 fprintf(fis2, '      ');
 fprintf(fis2, ['after=1; \n']);
 fprintf(fis2, '      ');
 fprintf(fis2, ['ffafter=flops; \n']);
 fprintf(fis2, '      ');
 fprintf(fis2,['save_workspace(proc_numb,' c 'after.' n ');\n']);
 [A,COUNT]=fread(fis,pose1-1-pose1b,'char')
 fprintf(fis2, '%c', A);
 fprintf(fis2, [',ffafter-ff); \n']);
 [A,COUNT]=fread(fis,posl1b-pose1,'char')
 fprintf(fis2, '      ');
 fprintf(fis2, 'return \n');
 [A,COUNT]=fread(fis,pose1a-posl1b-1,'char')
 fprintf(fis2, '%c', A);
 fprintf(fis2, ['after=1; \n']);
```

```
  fprintf(fis2, '     ');
  fprintf(fis2, ['ffafter=flops; \n']);
  fprintf(fis2, '     ');
  fprintf(fis2,['save_workspace(proc_numb,' c 'after,' n ');\n']);
  [A,COUNT]=fread(fis,posl1c-pose1a+1,'char')
  fprintf(fis2, '%c', A);
  fprintf(fis2, [',ffafter-ff,nvr']);
  [A,COUNT]=fread(fis,posl4c-posl1c,'char')
  [A,COUNT]=fread(fis,posl1a-posl4c,'char')
  fprintf(fis2, '%c', A);
  fprintf(fis2, '\n');
  fprintf(fis2, '     ');
  fprintf(fis2, 'return \n');
  [A,COUNT]=fread(fis,posend-posl1a,'char');
  fprintf(fis2, '%c', A);
  fprintf(fis2, '\n');
  fprintf(fis2, '  ');
  fprintf(fis2,['end \n']);
  fprintf(fis2, '  ');
  fprintf(fis2,['after=0; \n']);
  [A,COUNT]=fread(fis,'char')
  fprintf(fis2, '%c', A);
  fprintf(fis2, ['ffafter=flops; \n']);
  fprintf(fis2, '     ');
  fprintf(fis2,['save_workspace(proc_numb,' c 'after,' n ');\n']);
  fprintf(fis2,['finish(proc_numb,flops-ff);\n']);
 end
end


if (FUNCTIONS(i,4)==1)
 if (FUNCTIONS(i,3)==2)
  [A,COUNT]=fread(fis,posicao1-pos10,'char');
  fprintf(fis2, '%c', A);
  fprintf(fis2, '\t');
  fprintf(fis2,['if after==0']);
  fprintf(fis2, '\n');
  fprintf(fis2, ' t');
  fprintf(fis2, '\t');
  fprintf(fis2, ['after=1;']);
  fprintf(fis2, '\n');
  fprintf(fis2, '\t');
  fprintf(fis2, ' t');
  fprintf(fis2,['save_workspace(proc_numb,' c 'after,' n ');\n']);
  fprintf(fis2, ' t');
  fprintf(fis2, '\t');
  [A,COUNT]=fread(fis,posu-posicao1,'char')
  fprintf(fis2, '%c', A);
  fprintf(fis2, [',flops-ff);']);
  fprintf(fis2, ' n');
  fprintf(fis2, ' t');
  fprintf(fis2, ' t');
  [A,COUNT]=fread(fis,posaux1-posu,'char')
  fprintf(fis2, 'return');
  fprintf(fis2, ' n');
  fprintf(fis2, ' t');
```

```
 fprintf(fis2, 'end \n');
 fprintf(fis2, '\t');
 fprintf(fis2,['if after==1']);
 fprintf(fis2, '\n');
 fprintf(fis2, '\t');
 fprintf(fis2, '\t');
 fprintf(fis2, ['after=2;']);
 fprintf(fis2, '\n');
 fprintf(fis2, '\t');
 fprintf(fis2, '\t');
 fprintf(fis2,['save_workspace(proc_numb,' c 'after,' n ');\n']);
 fprintf(fis2, '\t');
 fprintf(fis2, '\t');
 clear A;
 [A,COUNT]=fread(fis,post-posicao2,'char')
 fprintf(fis2, '%c', A);
 [A,COUNT]=fread(fis,posaux2-post,'char')
 fprintf(fis2, [',flops-ff,nvr);']);
 fprintf(fis2, '\n');
 fprintf(fis2, '\t');
 fprintf(fis2, '\t');
 fprintf(fis2, 'return');
 fprintf(fis2, '\n');
 fprintf(fis2, '\t');
 fprintf(fis2, 'end \n');
 fprintf(fis2, 'end \n');
 fprintf(fis2, 'end \n');
 fprintf(fis2,['finish(proc_numb,flops-ff);\n']);
end

if (FUNCTIONS(i,3)==3)
 [A,COUNT]=fread(fis,pose1a-pos10,'char');
 fprintf(fis2, '%c', A);
 fprintf(fis2, ' ');
 fprintf(fis2,['if (after==0) | (after==4) \n']);
 fprintf(fis2, '    ');
 fprintf(fis2, ['after=1; \n']);
 fprintf(fis2, '    ');
 fprintf(fis2,['save_workspace(proc_numb,' c 'after,' n ');\n']);
 [A,COUNT]=fread(fis,posl1c-pose1a,'char')
 fprintf(fis2, '%c', A);
 fprintf(fis2, ['flops-ff,nvr)']);
 [A,COUNT]=fread(fis,posl4c-posl1c,'char');
 [A,COUNT]=fread(fis,posl1a-posl4c,'char')
 fprintf(fis2, '%c', A);
 fprintf(fis2, '    ');
 fprintf(fis2, 'return\n');
 fprintf(fis2, ' ');
 fprintf(fis2, 'end\n');
 fprintf(fis2, ' ');
 fprintf(fis2,['if after==1 \n']);
 [A,COUNT]=fread(fis,pose1b-posl1a,'char');
 fprintf(fis2, '%c', A);
 fprintf(fis2, '    ');
 fprintf(fis2,['after=2; \n']);
 fprintf(fis2, '    ');
```

```
fprintf(fis2,['save_workspace(proc_numb,' c 'after,' n ');\n']);
[A,COUNT]=fread(fis,pose1-1-pose1b,'char')
fprintf(fis2, '%c', A);
fprintf(fis2, [',flops-ff); \n']);
[A,COUNT]=fread(fis,posl1b-pose1,'char')
fprintf(fis2, '    ');
fprintf(fis2, 'return \n');
fprintf(fis2, '  ');
fprintf(fis2, 'end\n');
fprintf(fis2, '  ');
fprintf(fis2,['if after==2 \n']);
fprintf(fis2, '    ');
fprintf(fis2,['after=3; \n']);
[A,COUNT]=fread(fis,pose2b-posl1b,'char')
fprintf(fis2, '%c', A);
fprintf(fis2,'\n');
fprintf(fis2, '       ');
fprintf(fis2,['save_workspace(proc_numb,' c 'after,' n ');\n']);
fseek(fis, pose2b, 'bof');
g=0;
contline8=0;
while (~(feof(fis)) & (~g))
  [A,COUNT]=fscanf(fis, '%c%');
  g=strcmp(A,')');
  if g==1
    contline8=contline8+1;
    if contline8==1
      posg1=ftell(fis);
    end
  end

end
fseek(fis, pose2b, 'bof');
[A,COUNT]=fread(fis,posg1-1-pose2b,'char')
fprintf(fis2, '%c', A);
fprintf(fis2, [',flops-ff); \n']);
[A,COUNT]=fread(fis,posl2b-posg1,'char')
fprintf(fis2, '       ');
fprintf(fis2, 'return \n');
[A,COUNT]=fread(fis,posl1bb-posl2b,'char')
fprintf(fis2, '%c', A);
[A,COUNT]=fread(fis,posl1bbb+1-posl1bb,'char')
fprintf(fis2, '%c', A);
fprintf(fis2,'\n');
fprintf(fis2, '  ');
fprintf(fis2, 'end \n');
fprintf(fis2, '  ');
fprintf(fis2, 'if after==3 \n');
fprintf(fis2, '    ');
fprintf(fis2, 'after==4; \n');
[A,COUNT]=fread(fis,posl1bbbb-3-posl1bbb,'char')
fprintf(fis2, '%c', A);
fprintf(fis2, '\n');
fprintf(fis2, '  ');
fprintf(fis2, 'end \n');
[A,COUNT]=fread(fis,'char')
```

```
 fprintf(fis2, '%c', A);
 fprintf(fis2,'\n');
 fprintf(fis2,['finish(proc_numb,flops-ff);\n']);
end

if (FUNCTIONS(i,3)==4)
 [A,COUNT]=fread(fis,pose1a-pos10,'char');
 fprintf(fis2, '%c', A);
 fprintf(fis2, '  ');
 fprintf(fis2,['if (after==0) | (after==6) \n']);
 fprintf(fis2, '    ');
 fprintf(fis2, ['after=1; \n']);
 fprintf(fis2, '    ');
 fprintf(fis2,['save_workspace(proc_numb,' c 'after,' n ');\n']);
 [A,COUNT]=fread(fis,posl1c-pose1a,'char')
 fprintf(fis2, '%c', A);
 fprintf(fis2, ['nvr,']);

 [A,COUNT]=fread(fis,posl2c-posl1c,'char')
 [A,COUNT]=fread(fis,posl3c-posl2c,'char')
 fprintf(fis2, '%c', A);
 fprintf(fis2, ['nvr)']);
 [A,COUNT]=fread(fis,posl4c-posl3c,'char')
 [A,COUNT]=fread(fis,posl1a-posl4c,'char')
 fprintf(fis2, '%c', A);
 fprintf(fis2, '    ');
 fprintf(fis2, 'return \n');
 fprintf(fis2, '  ');
 fprintf(fis2, 'end \n');
 fprintf(fis2, '  ');
 fprintf(fis2,['if after==1 \n']);
 fprintf(fis2, '    ');
 fprintf(fis2, ['[dest.source]=processes_id(chan4); \n']);
 [A,COUNT]=fread(fis,pose1b-posl1a,'char')
 fprintf(fis2, '%c', A);
 fprintf(fis2, '      ');
 fprintf(fis2,['after=2; \n']);
 fprintf(fis2, '      ');
 fprintf(fis2,['save_workspace(proc_numb,' c 'after,' n ');\n']);
 [A,COUNT]=fread(fis,pose1-1-pose1b,'char')
 fprintf(fis2, '%c', A);
 fprintf(fis2, [',0); \n']);
 [A,COUNT]=fread(fis,posl1b-pose1,'char')
 fprintf(fis2, '      ');
 fprintf(fis2, 'return  n');
 [A,COUNT]=fread(fis,pose2b-posl1b,'char')
 fprintf(fis2, '%c', A);
 fprintf(fis2,'\n');
 fprintf(fis2, '      ');
 fprintf(fis2, ['after=3; \n']);
 fprintf(fis2, '    ');
 fprintf(fis2, 'end \n');
 fprintf(fis2, '  ');
 fprintf(fis2, 'end \n');
 fprintf(fis2, '  ');
 fprintf(fis2, 'if after==3 \n');
```

```
fprintf(fis2, '    ');
fprintf(fis2, 'after=4; \n');
fprintf(fis2, '    ');
fprintf(fis2,['save_workspace(proc_numb,' c 'after,' n ');\n']);
fseek(fis, pose2b, 'bof');
g=0;
contline8=0;
while (~(feof(fis)) & (~g))
  [A,COUNT]=fscanf(fis, '%c%');
  g=strcmp(A.',');

  if g==1
    contline8=contline8+1;
    if contline8==1
      posg1=ftell(fis);
    end
  end

end
fseek(fis, pose2b, 'bof');
[A,COUNT]=fread(fis,posg1-pose2b,'char')
fprintf(fis2, '%c', A);
fseek(fis, posg1, 'bof');
i=0;
contline10=0;
while (~(feof(fis)) & (~i))
  [A,COUNT]=fscanf(fis, '%c%');
  i=strcmp(A.')');

  if i==1
    contline10=contline10+1;
    if contline10==1
      posi1=ftell(fis);
    end
  end

end
fseek(fis, posg1, 'bof');
[A,COUNT]=fread(fis,posi1-1-posg1,'char')
fprintf(fis2, '%c', A);
fprintf(fis2, [',0); \n']);
[A,COUNT]=fread(fis,posl2b-posi1,'char')
fprintf(fis2, '    ');
fprintf(fis2, 'return \n');
fprintf(fis2, '  ');
fprintf(fis2, 'end \n');
fprintf(fis2, '  ');
fprintf(fis2, 'if after==4 \n');
fprintf(fis2, '    ');
fprintf(fis2, 'after=5; \n');
[A,COUNT]=fread(fis,pose3b-posl2b,'char')
fprintf(fis2, '%c', A);
fprintf(fis2,'\n');
fprintf(fis2, '            ');
fprintf(fis2,['save_workspace(proc_numb,' c 'after,' n ');\n']);
```

```
        fseek(fis, pose3b, 'bof');
        h=0;
        contline9=0;
        while (~(feof(fis)) & (~h))
          [A,COUNT]=fscanf(fis, '%c%');
          h=strcmp(A,')');
          if h==1
            contline9=contline9+1;
            if contline9==1
              posh1=ftell(fis);
            end
          end

        end
        fseek(fis, pose3b, 'bof');
        [A,COUNT]=fread(fis,posh1-1-pose3b,'char')
        fprintf(fis2, '%c', A);
        fprintf(fis2, [',0); \n']);
        [A,COUNT]=fread(fis,posl3b-posh1,'char')
        fprintf(fis2, '          ');
        fprintf(fis2, 'return \n');
        fprintf(fis2, '    ');
        fprintf(fis2, 'end \n');
        fprintf(fis2, '  ');
        fprintf(fis2, 'end \n');
        fprintf(fis2, '  ');
        fprintf(fis2,['if (after==5) | (after==2)  n']);
        fprintf(fis2, '    ');
        fprintf(fis2, ['i=i+1; \n']);
        fprintf(fis2, '    ');
        fprintf(fis2, ['after=6; \n']);
        fprintf(fis2, '  ');
        fprintf(fis2, 'end \n ');
        fprintf(fis2, 'end \n ');
        fprintf(fis2,['finish(proc_numb,0);\n']):
      end
     end
    fclose(fis);
    fclose(fis2);

  end

  fis2=fopen('auxiliar.m','r');

  while ~(feof(fis2))
   [A,COUNT]=fread(fis2, 'char');
  end

  fclose(fis2);
  fis=fopen(fi, 'w');
  COUNT=fprintf(fis, '%s', A );

  fclose(fis);

end %for i range i
```

ANEXO B

Programa Sim

Programa Sim_Global

É apresentado neste anexo o código do programa **Sim** de simulação do sistema constituído por *Transputers*, *C40's* ou misto seguindo-se o código do programa **Sim_Global** no qual se encontram definidas as variáveis globais presentes no sistema a simular:

```
%   This file Sim, starts the simulation of a transputer system

sim_global

filename=input('Filename for the topology of the system - ');
cont=1;
while cont
 f=[filename '.top'];
 setstr(f);
 if exist(f)~=2
   disp('The filename does not  exist in disk! ');
   filename=input('Filename for the topology of the system - ');
 else
   eval(['load ' f' -mat']);
   cont=0;
 end
end
filename=input('Filename for the initialization - ');
cont=1;
while cont
 ff=[filename '_ini'];
 f=[ff '.m'];
 setstr(f);
 if exist(f)~=2
   disp('The filename does not  exist in disk! ');
   filename=input('Filename for the topology of the system - ');
 else
   cont=0;
 end
end
eval(ff);
[op.pn]=retr_syst_events
while pn>0
 feval(op,pn);
 [op.pn]=retr_syst_events
end
quer=input('Do you want to save the data (1-yes/0-no) ? ');
if quer
 filename=input('filename ');
 eval(['save '  filename '.res']);
end
quer=input('Do you want to know the times (1-yes/0-no) ? ');
if quer
 show_t(A);
end
```

```
%   This file Sim_Global, takes care of the assignments of the global variables for
%   the transputer simulation

global NUM_PROC CHAN_ID NUM_CHANNELS CHANNEL NUM_ALT_WAIT ALT_WAIT
global NUM_ALT_SEND ALT_SEND HAS_REC_FROM
global EVENT_START_SEND EVENT_END_SEND EVENT_START_REC EVENT_END_REC
global EVENT_WANTS_TO_SEND EVENT_WANTS_TO_REC EVENT_END_WORKING
global EVENT_START_EXE
global EVENTS TIME WORKSPACE FLOP_TIME NUM_FUN FUNCTIONS FUN_ID
global INTERNAL_FIXED_COM_TIME EXTERNAL_FIXED_COM_TIME
global INTERNAL_VAR_COM_TIME
global EXTERNAL_VAR_COM_TIME NUM_PROCESSORS PROCESSES
global READY_HPP READY_LPP PROCESSOR_TIME PROCESSOR ID_TOPOLOGY
global  ID_PROCESSOR


NUM_ALT_WAIT=0;
NUM_ALT_SEND=0;
EVENT_START_SEND=1;
EVENT_START_REC=2;
EVENT_END_SEND=3;
EVENT_END_REC=4;
EVENT_WANTS_TO_SEND=5;
EVENT_WANTS_TO_REC=6;
EVENT_START_EXE=7;
EVENT_END_WORKING=10;


INTERNAL_FIXED_COM_TIME=0;
INTERNAL_VAR_COM_TIME=0;

% For transputer systems ( matrix : 100*50 )
%  FLOP_TIME=1.8e-6;
%  EXTERNAL_VAR_COM_TIME=2.3e-6;
%  EXTERNAL_FIXED_COM_TIME=0;

% For C40 systems ( matrix: 44*44 )
% FLOP_TIME=1.6e-6;
% EXTERNAL_VAR_COM_TIME=2.35e-7;
% EXTERNAL_FIXED_COM_TIME=2.6e-4;

% For T8/C40 systems (matrix: 30*20 )
% FLOP_TIME=7.4e-6; % For Transputer
% FLOP_TIME=1.6e-6; % For C40
% EXTERNAL_VAR_COM_TIME=9.4e-8;
% EXTERNAL_FIXED_COM_TIME=0;

FLOP_TIME=[7.4e-6; 1.6e-6];
EXTERNAL_VAR_COM_TIME=[2.3e-6; 2.35e-7; 9.40e-8];
EXTERNAL_FIXED_COM_TIME=[0; 2.6e-4; 0];
```

ANEXO C

Função alt_wait

Função change_workspace

Função choose_plot

Função cpu_time

Função finish

Função get_events

Função get_time

Função insert_event

Função insert_syst_events

Função operation

Função processes_id

Função put_u

Função rendez_vous

Função restore_workspace

Função retr_syst_events

Função save_workspace

Função send

Função show_t

Função sig_w_re

Função sig_w_se

Função spet

Função strcat

Função strcount

Função strcut

Função strmat

Função test_alt

Função test_alts

Função wait

Função wait_rec

Função wait_sen

Neste anexo estão incluídas todas as listagens de programas que implementam as **funções internas** e **externas** da *Toolbox:*

```
function alt_wait(ch_1,n_v_1,ch_2,n_v_2,ch_3,n_v_3,ch_4,n_v_4,ch_5,n_v_5)

%   This function implements an ALT PRI WAIT, on channels ch_i, ch_2, ch_3
%   The received input will be stored in n_v_i in the workspace of the ith
%   process

global HAS_REC_FROM CHANNEL NUM_ALT_WAIT ALT_WAIT

%   first test if there was a message sent
has_sent=0;
if wait_sen(ch_1)
 has_sent=1;
 chan=ch_1;
 nv=n_v_1;
elseif wait_sen(ch_2)
 has_sent=1;
 chan=ch_2;
 nv=n_v_2;
elseif nargin>=6
 if wait_sen(ch_3)
  has_sent=1;
  chan=ch_3;
  nv=n_v_3;
 end
elseif nargin>=8
 if wait_sen(ch_4)
  has_sent=1;
  chan=ch_4;
  nv=n_v_4;
 end
elseif nargin>=10
 if wait_sen(ch_5)
  has_sent=1;
  chan=ch_5;
  nv=n_v_5;
 end
end
if has_sent==1
 % the source process associated with channel wants to send
 HAS_REC_FROM(CHANNEL(chan,2))=CHANNEL(chan,1);
 wait(chan,0,nv);
else
 % any of the source processes are not waiting to send
 % insert the alt_wait in the list of ALT_WAITs
 [dest,source1]=processes_id(ch_1);
 NUM_ALT_WAIT=NUM_ALT_WAIT+1;
 ALT_WAIT(NUM_ALT_WAIT,1)=dest;
 ALT_WAIT(NUM_ALT_WAIT,2)=nargin/2;
 if nargin>=6
  ALT_WAIT(NUM_ALT_WAIT,7)=ch_3;
  ALT_WAIT(NUM_ALT_WAIT,8)=n_v_3;
```

```
 end
 if nargin>=8
  ALT_WAIT(NUM_ALT_WAIT,9)=ch_4;
  ALT_WAIT(NUM_ALT_WAIT,10)=n_v_4;
 end
 if nargin==10
  ALT_WAIT(NUM_ALT_WAIT,11)=ch_5;
  ALT_WAIT(NUM_ALT_WAIT,8)=n_v_5;
 end
 ALT_WAIT(NUM_ALT_WAIT,3)=ch_1;
 ALT_WAIT(NUM_ALT_WAIT,4)=n_v_1;
 ALT_WAIT(NUM_ALT_WAIT,5)=ch_2;
 ALT_WAIT(NUM_ALT_WAIT,6)=n_v_2;
 % put all the possible destination processes in a wait state
 wait(ch_1,0,n_v_1);
 wait(ch_2,0,n_v_2);
 if nargin>=6
  wait(ch_3,0,n_v_3);
 end
 if nargin>=8
  wait(ch_4,0,n_v_4);
 end
 if nargin==10
  wait(ch_5,0,n_v_5);
 end
end


function change_workspace(var,proc_number,number_of_variable)

%   This function changes the var number_of_variable in the
%   workspace related with proc proc_numb with variable var

global WORKSPACE

size_u=size(var);
u=var(:);
tamb=0;
tama=0;
k=2;
for i=1:WORKSPACE(proc_number,1)
 if i<number_of_variable
  p=WORKSPACE(proc_number,k)*WORKSPACE(proc_number,k+1);
  tamb=tamb+p+2;
  k=k-p+2;
 end
 if i==number_of_variable
  pnt=k;
  p=WORKSPACE(proc_number,k)*WORKSPACE(proc_number,k+1);
  tam=p+2;
  k=k-p+2;
 end
 if i>number_of_variable
  p=WORKSPACE(proc_number,k)*WORKSPACE(proc_number,k+1);
  tama=tama+p+2;
```

```
   k=k+p+2;
  end
end
prod_u=size_u(1)*size_u(2);
temp=WORKSPACE(proc_number,tamb+tam+2:tamb+tam+tama+1);
if length(temp)~=0
  WORKSPACE(proc_number,tamb+4+prod_u:tamb+3+prod_u+tama)=temp;
end
WORKSPACE(proc_number,tamb+4:tamb+prod_u+3)=u';
WORKSPACE(proc_number,tamb+2:tamb+3)=size_u;



function choose_plot(how_many,i)

if how_many>2
  if i==1
    subplot(221)
  elseif i==2
    subplot(222)
  elseif i==3
    subplot(223)
  else
    subplot(224)
  end
elseif how_many==2
  if i==1
    subplot(211)
  else
    subplot(212)
  end
end



function [t_on,t_off,x,y]=cpu_time(p,op,ini_time.end_time)

global NUM_PROC PROCESSES EVENTS EVENT_START_EXE

process=[];
for j=1:NUM_PROC
  if PROCESSES(j,1)==p
    process=[process j];
  end
end
% determine the on and off time of all the processes that execute in p1
for j=1:length(process)
  last=0;
  m=0;
  proc=process(j);
  for k=1:EVENTS(proc,1)
    if (last==1)
      if m~=0
        if work(j.m+1)~=EVENTS(proc,2*k+1)
          m=m+1;
          work(j.m+1)=EVENTS(proc,2*k+1);
```

```
        last=0;
       else
        m=m-1;
        last=0;
       end
      else
       m=m+1;
       work(j,m+1)=EVENTS(proc,2*k+1);
       last=0;
      end
    elseif (last==0)&(EVENTS(proc,2*k)==EVENT_START_EXE)
     m=m+1;
     work(j,m+1)=EVENTS(proc,2*k+1);
     last=1;
    end
  end
  if m~=0
   m=m+1;
   work(j,m+1)=EVENTS(proc,2*EVENTS(proc,1)+1);
   work(j,1)=m;
  end
end
% determine the on times of the processor
numb=0;
pro=[];
w=[];
for j=1:length(process)
 if work(j,1)~=0
  pro=[pro process(j)];
  w(length(pro),:)=work(j,:);
 end
end
process=pro;
work=w;
clear w pro
if length(process)>1
 % there are more than 1 working process
 num_proc=length(process);
 on_time=work(1:num_proc,1);
 pnt=2*ones(num_proc,1);
 number=1;
 w(2)=min(on_time);
 while min(on_time)~=1e90
  % find the next on_time
  pos=find(min(on_time)==on_time);
  on=on_time(pos);
  off=work(pos,2:work(pos,1)+1);
  keyboard
  if length(find(off==on_time))==0
   % the next on_event is not at the same time as the off-time
   number=number+1;
   w(number-1)=on;
   number=number+1;
   w(number-1)=off;
   if pnt(pos)>work(pos,1)
    pnt(pos)=pnt(pos)+2;
```

```
      on_time(pos)=work(pos.pnt(pos));
     else
      on_time(pos)=1e90;
     end
    else
     % the next on_event is at the same time as the off_time
     pos=find(off==on_time):
     if pnt(pos)>work(pos,1)
      pnt(pos)=pnt(pos)+2;
      on_time(pos)=work(pos.pnt(pos));
     else
      on_time(pos)=1e90;
     end
    end
  end
  w(1)=number;
  work=w;
  clear w
end
if op==1
  last=work(2);
  on=1;
  x=[]:
  y=[]:
  for j=2:work(1)
    x1=last+eps;
    x2=work(j+1);
    if (x1>=ini_time)
      if (x2<=end_time)
        if x2>x1
          x=[x x1 x2];
          y=[y on on];
        end
      elseif x1<=end_time
        x=[x x1 end_time];
        y=[y on on]:
      end
    end
    last=x2:
    on=-on:
  end
end
if(work)==0
  work(1)=0;
end
last=0:
t_on=0:
t_off=0:
on=-1:
for j=2:work(1)+1
  time=work(j)-last;
  if on==1
    t_on=t_on+time;
  else
    t_off=t_off+time;
  end
```

```
 last=work(j);
 on=-on;
end


function finish(proc_numb,flo)

global EVENT_END_WORKING TIME

event=EVENT_END_WORKING;
time=TIME(proc_numb)+gettime(flo,0,0,proc_numb);
insert_event(event,proc_numb,time);



function [x,y]=get_events(proc,ini,fim)

%   This function creates the time axis and the event axis for
%   process proc

global EVENTS EVENT_END_WORKING EVENT_START_SEND EVENT_START_REC
global EVENT_WANTS_TO_SEND EVENT_WANTS_TO_REC EVENT_START_EXE
% create
beg=1;
for j=1:EVENTS(proc,1)
 time=EVENTS(proc,1+j*2);
 if (time>=ini)&(time<=fim)
   en=j;
 elseif time<ini
   beg=j;
 end
end
num_events=en-beg+1;
acont=EVENTS(proc,beg*2:en*2+1);
lastacont=acont(num_events*2-1);
if lastacont~=EVENT_END_WORKING
 num_events=num_events+1;
 acont(num_events*2-1)=EVENT_END_WORKING;
 acont(num_events*2)=fim;
end
x=[];
y=[];
last=acont(2);
for j=1:num_events-1;
 x1=last+eps;
 x2=acont((j+1)*2);
 event=acont(j*2-1);
 if event==EVENT_START_SEND;
   eve=-1;
 elseif event==EVENT_START_REC;
   eve=1;
 elseif event==EVENT_WANTS_TO_SEND
   eve=-2;
 elseif event==EVENT_WANTS_TO_REC
   eve=2;
```

```
 elseif event==EVENT_START_EXE
  eve=0;
 else
  nele=0;
 end
 if x2>x1
  x=[x x1 x2];
  y=[y eve eve];
 end
 last=x2;
end


function time_lapse=getTime(quant,flo_com,chan,proc_numb)

%   This function returns the time spent performing quant floating point
%   operations (if flo/com=0) or communicating a real32  vector with
%   quant elements, by channel chan. The use of chan allows to determinate
%   if it is an internal or external channel.

global PROCESSOR ID_TOPOLOGY
global FLOP_TIME PROCESSES CHANNEL INTERNAL_FIXED_COM_TIME
global INTERNAL_VAR_COM_TIME EXTERNAL_FIXED_COM_TIME
global EXTERNAL_VAR_COM_TIME

processador=PROCESSES(proc_numb,1);
p=PROCESSOR(processador,2);
 if flo_com==0
   % it is a floating point operation
time_lapse=quant*FLOP_TIME(p,1);
 else
   % it is a communication
  if PROCESSES(CHANNEL(chan,1),1)==PROCESSES(CHANNEL(chan,2),1)
    % it is an internal communication
    time_lapse=INTERNAL_FIXED_COM_TIME+quant*INTERNAL_VAR_COM_TIME;
  else
   if ID_TOPOLOGY==1
     time_lapse=EXTERNAL_FIXED_COM_TIME(1,1)+quant*EXTERNAL_VAR_COM_TIME(1,1);
   end
   if ID_TOPOLOGY==2
     time_lapse=EXTERNAL_FIXED_COM_TIME(2,1)+quant*EXTERNAL_VAR_COM_TIME(2,1);
   end
   if ID_TOPOLOGY==3
     time_lapse=EXTERNAL_FIXED_COM_TIME(3,1)+quant*EXTERNAL_VAR_COM_TIME(3,1);
   end
  end
 end


function insert_e(event,proc_numb,time)

%   This function stores the events associated with each process in a list
%   to keep the time history of the process
```

```
global EVENTS PROCESSES PROCESSOR_TIME

EVENTS(proc_numb,1)=EVENTS(proc_numb,1)+1;
pnt=2*EVENTS(proc_numb,1);
EVENTS(proc_numb,pnt)=event;
EVENTS(proc_numb,pnt+1)=time;
if PROCESSES(proc_numb,2)==0
 PROCESSOR_TIME(PROCESSES(proc_numb,1))=time;
end



function insert_syst_events(process,time)

%  This function  is responsible for the insertion of ready processes in the
%  appropriate queue. The queues are ordered w.r.t. time, and if there is
%  in the same queue a process with the same time, the new process will be
%  insered after that process

global PROCESSES READY_HPP READY_LPP

processor=PROCESSES(process,1);
if PROCESSES(process,2)==1
 % it is a high priority process
 if READY_HPP(processor,1)==0
  % there are no ready processes
  READY_HPP(processor,1)=1;
  READY_HPP(processor,2)=time;
  READY_HPP(processor,3)=process;
 else
  % there are ready process; pnt will denote the first process with  a greater
  % time to begin executing
  pnt=1;
  cont=1;
  while cont
   if READY_HPP(processor,pnt*2)>time
    cont=0;
   else
    pnt=pnt+1;
    if pnt>READY_HPP(processor,1)
     cont=0;
    end
   end
  end
  % insert the new record
  if pnt<=READY_HPP(processor,1)
   READY_HPP(processor,(pnt+1)*2:(2*(READY_HPP(processor,1)+1)+1))=...
   READY_HPP(processor,pnt*2:(2*(READY_HPP(processor,1))+1));
  end
  READY_HPP(processor,2*pnt)=time;
  READY_HPP(processor,(2*pnt+1))=process;
  READY_HPP(processor,1)=READY_HPP(processor,1)+1;
 end
else
 % it is a low priority process
 if READY_LPP(processor,1)==0
```

```
    % there are no ready processes
    READY_LPP(processor,1)=1;
    READY_LPP(processor,2)=time;
    READY_LPP(processor,3)=process;
  else
    % there are ready process; pnt will denote the first process with  a greater
    % time to begin executing
    pnt=1;
    cont=1;
    while cont
     if READY_LPP(processor,pnt*2)>time
      cont=0;
     else
      pnt=pnt+1;
      if pnt>READY_LPP(processor,1)
       cont=0;
      end
     end
    end
    % insert the new record
    if pnt<=READY_LPP(processor,1)
     READY_LPP(processor,(pnt+1)*2:(2*(READY_LPP(processor,1)+1)+1))=...
     READY_LPP(processor,pnt*2:(2*(READY_LPP(processor,1))+1));
    end
    READY_LPP(processor,2*pnt)=time;
    READY_LPP(processor,(2*pnt+1))=process;
    READY_LPP(processor,1)=READY_LPP(processor,1)+1;
  end
end



function op=operation(proc_numb)

%   This function is used  to return the identifier of the function

global PROCESSES FUNCTIONS FUN_ID

fun=PROCESSES(proc_numb,3);
l=FUNCTIONS(fun,5);
op=FUN_ID(fun,1:l);



function [dest_iden,source_iden,number_of_var]=processes_id(chan);

%   This function returns the identifiers of the source and destination
%   processes related with chan

global CHANNEL

dest_iden=CHANNEL(chan,2);
source_iden=CHANNEL(chan,1);
if nargout==3
  number_of_var=CHANNEL(chan,5);
end
```

```
function len=put_u(chan,nv);

%   This function puts the variable u in the workspace of the destination
%   process

global CHANNEL

size_u=CHANNEL(chan,6:7);
len=prod(size_u);
u=CHANNEL(chan,8:7+size_u);
k=1;
for j=1:size_u(2)
  var(1:size_u(1),j)=u(k:k+size_u(1)-1)';
  k=k+size_u(1);
end
dest=CHANNEL(chan,2);
change_workspace(var,dest,nv);



function rendez_vous(chan)

%   This function stores the indication that communication has been
%   made

global CHANNEL

CHANNEL(chan,3)=0;
CHANNEL(chan,4)=0;



function
[o1,o2,o3,o4,o5,o6,o7,o8,o9,o10,o11,o12,o13,o14,o15,o16,o17,o18,o19,o20,o21]=restore_workspace(pro
c_numb)

%   This function is used to restore the workspace of process proc_numb.
%   WORKSPACE is a global variable

global WORKSPACE

if nargin>21
  disp('Error in restore_workspace! Number of variables limited to 21');
end
k=2;
for l=1:nargout
  s=WORKSPACE(proc_numb,k:k+1);
  k=k+2;
  for j=1:s(2)
    eval(['o',num2str(l),'(1:',num2str(s(1)),',',num2str(j),')=WORKSPACE(' ...
num2str(proc_numb) ',k:',num2str(s(1)),'+k-1)';']);
    k=k+s(1);
  end
end
```

```
function [op,proc_numb]=retr_syst_events

%   This function is responsible for the retieval of events
%   from the ready queues. It returns the string denoting
%   the name of the function and the process number. Also
%   it inserts in the list of events of the process a confirmation
%   of the execution

global NUM_PROCESSORS READY_HPP PROCESSES EVENT_START_EXE READY_LPP
global PROCESSOR_TIME TIME

time=1e90;
%   find, if possible the earliest high priority process
prio=0;
for i=1:NUM_PROCESSORS
 if READY_HPP(i,1)>0
  if READY_HPP(i,2)<time
   time=READY_HPP(i,2);
   prio=1;
   proc_numb=READY_HPP(i,3);
  end
 end
end
if prio==1
 % a priority process has been found
 % take the process out of its list
 processor=PROCESSES(proc_numb,1);
 numele=READY_HPP(processor,1);
 if numele==1
  READY_HPP(processor,1)=0;
 else
  READY_HPP(processor,2:((numele-1)*2+1))=READY_HPP(processor,4:(numele*2+1));
  READY_HPP(processor,1)=READY_HPP(processor,1)-1;
 end
 op=operation(proc_numb);
 % insert the confirmation of execution
 insert_event(EVENT_START_EXE,proc_numb,time);
else
 % there are no priority PROCESSES
 % find, if possible the earliest low priority process
 normal=0;
 for i=1:NUM_PROCESSORS
  if READY_LPP(i,1)>0
   if READY_LPP(i,2)<time
    time=READY_LPP(i,2);
    normal=1;
    proc_numb=READY_LPP(i,3);
   end
  end
 end
 if normal==1
  % a priority process has been found
  % take the process out of its list
  processor=PROCESSES(proc_numb,1);
  numele=READY_LPP(processor,1);
  if numele==1
```

```
    READY_LPP(processor,1)=0;
   else
    READY_LPP(processor,2:((numele-1)*2+1))=READY_LPP(processor,4:(numele*2+1));
    READY_LPP(processor,1)=READY_LPP(processor,1)-1;
   end
   op=operation(proc_numb);
   if TIME(proc_numb)>PROCESSOR_TIME(processor)
    PROCESSOR_TIME(processor)=TIME(proc_numb);
   else
    TIME(proc_numb)=PROCESSOR_TIME(processor);
   end
   insert_event(EVENT_START_EXE,proc_numb,TIME(proc_numb));
  else
   % it was not possible to find one process ready; simulation is over
   op=[];
   proc_numb=0;
  end
end



function
save_workspace(proc_numb,i1,i2,i3,i4,i5,i6,i7,i8,i9,i10,i11,i12,i13,i14,i15,i16,i17,i18,i19,i20,i21)

%   This function is used to save the workspace of process proc_numb.
%   WORKSPACE is a global variable

global WORKSPACE

if nargin>22
  disp('Error in save_workspace! Number of variables limited to 21');
end
WORKSPACE(proc_numb,1)=nargin-1;
k=2;
for i=1:nargin-1
  eval(['s=size(i',num2str(i),');']);
  WORKSPACE(proc_numb,k:k+1)=s(:)';
  eval(['WORKSPACE(proc_numb,k+2:k+1+prod(s))=i',num2str(i),'(:)';']);
  k=k+2+prod(s);
end



function send(u,chan,flo)

%   This function implements the OCCAM ! primitive. Its arguments are:
%   u - vector to be sent
%   chan - channel_identifier
%   flo - number of floating point operations since last event
global TIME EVENTS EVENT_WANTS_TO_SEND EVENT_START_SEND EVENT_START_REC
global EVENT_END_SEND EVENT_END_REC

tes_alt(chan);
if wait_rec(chan)
  % the other process is waiting to receive
  rendez_vous(chan);
```

```
  [dest,source,number_of_var]=processes_id(chan);
  change_workspace(u,dest,number_of_var);
  time=TIME(source)+gettime(flo,0,chan,source);
  % see if the other process wanted to receive later
  if time<=TIME(dest)
   EVENTS(dest,1)=EVENTS(dest,1)-1;
   if time<TIME(dest)
    event=EVENT_WANTS_TO_SEND;
    insert_event(event,source,time);
    time=TIME(dest);
   end
  end
  event1=EVENT_START_SEND;
  event2=EVENT_START_REC;
  insert_event(event1,source,time);
  insert_event(event2,dest,time);
  time=time+gettime(length(u),1,chan,source);
  event1=EVENT_END_SEND;
  event2=EVENT_END_REC;
  insert_event(event1,source,time);
  insert_event(event2,dest,time);
  insert_syst_events(dest,time);
  if tes_alts(chan,time)==0
   insert_syst_events(source,time);
   TIME(source)=time;
  end
  TIME(dest)=time;
 else
  %  the other process is not waiting to receive
  sig_w_se(chan);
  store_u(chan,u);
  [dest,source]=processes_id(chan);
  time=TIME(source)+gettime(flo,0,chan,source);
  event=EVENT_WANTS_TO_SEND;
  insert_event(event,source,time);
  TIME(source)=time;
 end


 function show_t(a)

 global NUM_PROC EVENTS NUM_PROCESSORS FUNCTIONS FLOP_TIME

 for i=1:NUM_PROC
  last(i)=EVENTS(i,EVENTS(i,1)*2+1);
 end
 end_t=max(last);
 disp(['The Simulation stops at ' num2str(end_t) ]);
 already_no=0;
 graphs=input('Do you want plots (yes=1/no=0) ? ');
 if graphs
  clg
  figures=1;
  want=1;
  while want
```

```
figure(figures);
disp(' ');
disp(['Plot ' num2str(figures)]);
disp('------');
option=input('Processor (1) or process (2) data ? ');
if option==1
 plot_ef=input('Overall system (1) or specified processor (0) ? ');
 if plot_ef==1
  for j=1:NUM_PROCESSORS
   [t_on,t_off]=cpu_time(j,0);
   if (t_on+t_off)>0
    ef(j)=t_on/(t_on+t_off);
   else
    ef(j)=0;
   end;
   disp(['The efficiency of processor ' num2str(j) ' is ' num2str(ef(j))]);
  end
  %choose_plot(how_many,i);
  bar(1:NUM_PROCESSORS,ef,'w');
  title('Overall System');
  xlabel('Processor Number')
  ylabel('Efficiency');
 else
  %numb=input('1 or 2 graphs ? = ');
  p1=input('Which processor ? ');
  all_time=input('All the simulation (1=yes/0=no) ? ');
  if all_time==0
   ini_time=input('Initial time=');
   end_time=input('End Time (0 for end) =');
   if end_time==0
    end_time=end_t;
   end
  else
   ini_time=0;
   end_time=end_t;
  end
  [t_on1,t_off1,x1,y1]=cpu_time(p1,1,ini_time,end_time);
  ef1=t_on1/(t_on1+t_off1);
  x_max=x1(length(x1));
  disp(['The efficiency of processor ' num2str(p1) ' is ' num2str(ef1)]);
  %if numb==2
   %p2=input('Second processor ? ');
   %all_time=input('All the simulation (1=yes/0=no) ? ');
   %if all_time==0
    %ini_time=input('Initial time=');
    %end_time=input('End Time (0 for end) =');
    %if end_time==0
     %end_time=end_t;
    %end
   %else
    %ini_time=0;
    %end_time=end_t;
   %end
   %[t_on2,t_off2,x2,y2]=cpu_time(p2,1,ini_time,end_time);
   %ef2=t_on2/(t_on2+t_off2);
   %disp(['The efficiency of processor ' num2str(p2) ' is ' num2str(ef2)]);
```

```
      %x_max=max([x_max x2(length(x2))]);
      %x_min=min([x1(1) x2(1)]);
      %delta=(x_max-x_min)/10;
      %ax=[x_min-delta x_max+delta -4 4];
      %spet_plot(x1,y1+2,x2,y2-2,ax,how_many,i)

      %title(['Proc. ' num2str(p1) ' (above)/Proc. ' num2str(p2) ' (below)']);
      %xlabel('Time')
      %ylabel('Working Periods')
      %axis;
    %else
      delta=(x_max-x1(1))/10;
      x_min=x1(1);
      ax=[x_min-delta x_max+delta -1.5 1.5];
      spet_plot(x1,y1,ax)
      title(['Processor ' num2str(p1)'])
      xlabel('Time')
      ylabel('Working Periods')
      axis;
    %end
    pr=input('Efficiency figures in the graphics window (yes=1/no=0) ? ');
    if pr==1
      texto=['eff. for proc. ' num2str(p1) ' - ' num2str(ef1)];
      gtext(texto)
      %if numb==2
        %texto=['eff. for proc. ' num2str(p2) ' - ' num2str(ef2)];
        %gtext(texto)
      %end
    end
  end
else
  % process
  numb=input('1 or 2 graphs ? =');
  fun1=input('Which function=');
  if FUNCTIONS(fun1,2)>1
    num1=input('Which number=');
  else
    num1=1;
  end
  proc1=FUNCTIONS(fun1,1)+num1-1;
  all_time=input('All the simulation (1=yes/0=no) ? ');
  if all_time==0
    ini_time=input('Initial time=');
    end_time=input('End Time (0 for end) =');
    if end_time==0
      end_time=EVENTS(proc1,EVENTS(proc1,1)*2+1);
    end
  else
    ini_time=0;
    end_time=EVENTS(proc1,EVENTS(proc1,1)*2+1);
  end
  [x1,y1]=get_events(proc1,ini_time,end_time);
  x_max=x1(length(x1));
  x_min=x1(1);
  if numb==2
    fun2=input('Second function=');
```

```
  if FUNCTIONS(fun2,2)>1
   num2=input('Which number=');
  else
   num2=1;
  end
  proc2=FUNCTIONS(fun2,1)+num2-1;
  all_time=input('All the simulation (1=yes/0=no) ? ');
  if all_time==0
   ini_time=input('Initial time=');
   end_time=input('End Time=');
   if end_time==0
    end_time=EVENTS(proc2,EVENTS(proc2,1)*2+1);
   end
  else
   ini_time=0;
   end_time=EVENTS(proc2,EVENTS(proc2,1)*2+1);
  end
  [x2,y2]=get_events(proc2,ini_time,end_time);
  x_max=max([x_max x2(length(x2))]);
  x_min=min([x_min x2(1)]);
  delta=(x_max-x_min)/10;
  ax=[x_min-delta x_max+delta -6 6];
  spet_plot(x1,y1+3,x2,y2-3,ax)
  texto=[operation(proc1) '(' num2str(num1) ') (above)/' operation(proc2) '(' num2str(num2) ') (below)'];
  title(texto);
  xlabel('Time')
  ylabel('EVENTS')
  axis;
 else
  % just 1 plot
  delta=(x_max-x_min)/10;
  ax=[x_min-delta x_max+delta -3 3];
  spet_plot(x1,y1,ax);
  texto=[operation(proc1) '(' num2str(num1) ')'];
  title(texto);
  xlabel('Time')
  ylabel('EVENTS')
  axis;
 end
 if already_no==0
  already_no=1;
  quer=input('Do you want to see the conventions (1-yes/0-no) ?');
  if quer
   disp(' ');
   disp('The y axis may have the following values:');
   disp('   2 - waiting to receive');
   disp('   1 - actually receiving');
   disp('   0 - working');
   disp('  -1 - actually sending');
   disp('  -2 - waiting to send');
   disp('If 2 graphs are shown, they are shifted by +3 and -3')
   disp(' ');
  end
 end
end
pr=input('Output to printer (1=yes/0=no) ? ');
```

```
    if pr
      print -dmeta
    end
    figures=figures+1;
    want=input('Do you want another plot (1-yes / 0-no) ');
  end
  tit=input('Do you want a title (1=yes/0=no) ? ');
  if tit
    ti=input('Input Title - ');
    gtext(ti);
  end
end
if nargin==1
  quer=input('Do you want to know the speedup (1=yes/0=no) ? ');
  if quer
    [m,n]=size(a);
    y=rand(m,1);
    [x,ff,ff1]=solveqrr(a,y);
    time_par=0;
    for i=1:NUM_PROC
      if EVENTS(i,EVENTS(i,1)*2+1)>time_par
        time_par=EVENTS(i,EVENTS(i,1)*2+1);
      end
    end
    time_seq=ff1*FLOP_TIME;
    texto1=['Seq. Time=' num2str(time_seq)];
    texto2=['Par. Time=' num2str(time_par)];
    texto3=['Speedup=' num2str(time_seq/time_par)];
    disp([texto1 ' ' texto2 ' ' texto3]);
    if graphs
      pr=input('Figures in the graphic window (1=yes/0=no) ? ');
      if pr
        gtext(texto1);
        gtext(texto2);
        gtext(texto3);
      end
    end
  end
end
```

```
function sig_w_re(chan,number_of_variable)

%   This function stores the indication that the destination process related
%   to chan wants to receive data, and that the destination variable has
%   the position number_of_variable in its workspace

global CHANNEL

CHANNEL(chan,4)=1;
CHANNEL(chan,5)=number_of_variable;
```

```
function sig_w_se(chan)

%   This function stores the indication that the source process related
%   to chan wants to send data

global CHANNEL

CHANNEL(chan,3)=1;


function spet_plot(x1,y1,x2,y2,ax)

if nargin==5
  axi=ax;
else
  axi=x2;
end
clg
%choose_plot(how_many,ii);
plot(x1(1:2),y1(1:2),'w')
hold
k=3:4;
for i=2:length(x1)/2
  %choose_plot(how_many,ii);
  plot(x1(k),y1(k),'w')
  k=k+2;
end
if nargin==5
  k=1:2;
  for i=1:length(x2)/2
    %choose_plot(how_many,ii);
    plot(x2(k),y2(k),'w')
    k=k+2;
  end
end
hold
axis(axi);
axis;


function store_u(chan,u);

%   This function puts the variable u in a temporary storage place
%   associated with channel

global CHANNEL

CHANNEL(chan,6:7)=size(u);
CHANNEL(chan,8:7+prod(size(u)))=u(:)';
```

```
function[x]=strcat(a,b)

x=a;
for i=1:length(b)
   x(length(x)+1)=b(i);
end;
end


function[x]=strcount(a)

x=0;
if length(a)>0
   x=1;
   for i=1:length(a)
      if (a(i)==' ')
         x=x+1;
      end;
   end;
end;
end


function[x,y]=strcut(a,b)

encontrou=0;
x='';
if length(a) > 0
   for i=length(a):-1:1
    if (a(i)==b)
         encontrou=i;
    end;
   end;
   x=a(1:encontrou-1);
   y=a(encontrou+1:length(a));
end;
end


function [s]=strmat(a)

linhas=strcount(a);
s(linhas,10)= ' ';
for i=1:linhas -1
   s(i,:)='          ';
   [temp, a]=strcut (a, ',');
   s(i,1:length(temp))=temp;
end;
   s(linhas, 1:length(a))=a;
end
```

```
function tes_alt(chan)

%   This function tests if this channel is being used by an alt_wait;
%   If  it is, it takes cancels the alt_wait and selects the first
%   channel as the receiving channel

global NUM_ALT_WAIT ALT_WAIT CHANNEL HAS_REC_FROM

[dest,source]=processes_id(chan);
if NUM_ALT_WAIT>0
 pos=find(dest==ALT_WAIT(1:NUM_ALT_WAIT,1));
 if pos~=[]
  % there is an alt_wait
  HAS_REC_FROM(dest)=source;
  % signalizing that the other processes cannot send anymore
  m=3;
  for i=1:ALT_WAIT(pos,2)
   k=ALT_WAIT(pos,m);
   if k~=chan
    CHANNEL(k,4)=0;
   end
   m=m+2;
  end
  % take this alt_wait out of the list
  NUM_ALT_WAIT=NUM_ALT_WAIT-1;
  if NUM_ALT_WAIT>0
   temp=ALT_WAIT(1:pos-1,:);
   temp(pos:NUM_ALT_WAIT,:)=ALT_WAIT(pos+1:NUM_ALT_WAIT+1,:);
   ALT_WAIT=[];
   ALT_WAIT=temp;
  end
 end
end



function has_got=tes_alts(chan,time)

%   This function tests if there is an alt_send associated with a channel.
%   If there is, has_got will be true
%   If there is no alt_send, it_is_now returns false

%   test if there is an alt_send

global NUM_ALT_SEND ALT_SEND TIME

has_got=0;
[dest,source]=processes_id(chan);
if NUM_ALT_SEND>0
 pnt=1;
 for i=1:NUM_ALT_SEND
  if ALT_SEND(i,1)==source
   has_got=1;
   pnt=i;
  end
 end
```

```
end
if has_got
 if ALT_SEND(pnt,3)<(time-TIME(source))
  ALT_SEND(pnt,3)=time-TIME(source);
 end
 if ALT_SEND(pnt,2)==1
  % it is the last message
  TIME(source)=TIME(source)+ALT_SEND(pnt,3);
  insert_syst_events(source,TIME(source));
  %  take this alt_wait out of the list
  NUM_ALT_SEND=NUM_ALT_SEND-1;
  if NUM_ALT_SEND>0
   temp=ALT_SEND(pnt+1:NUM_ALT_SEND+1,:);
   ALT_SEND(pnt:NUM_ALT_SEND,:)=temp;
  end
 else
  % it is not the last message
  %  take the channel out of the list of channels
  for i=3:(ALT_SEND(pnt,2)+3)
   if  ALT_SEND(pnt,i)==chan
    pos=i;
   end
  end
  ALT_SEND(pnt,pos:(ALT_SEND(pnt,2)+2))=ALT_SEND(pnt,(pos+1):ALT_SEND(pnt,2)+3);
  ALT_SEND(pnt,2)=ALT_SEND(pnt,2)-1;
 end
end



function wait(chan,flo,number_of_variable);

%   This function implements the OCCAM ? primitive. Its arguments are:
%   chan -channel identifier
%   flo - number of floating point operations since last event

global TIME EVENTS EVENT_WANTS_TO_REC EVENT_START_SEND EVENT_START_REC
global EVENT_END_SEND EVENT_END_REC

if wait_sen(chan)
 % the other process wants to send
 rendez_vous(chan);
 [dest,source]=processes_id(chan);
 len=put_u(chan,number_of_variable);
  time=TIME(dest)+getTime(flo,0,chan,dest);
 % see if the other process wanted to send later
 if time<=TIME(source)
  EVENTS(source,1)=EVENTS(source,1)-1;
  if time<TIME(source)
   event=EVENT_WANTS_TO_REC;
   insert_event(event,dest,time);
   time=TIME(source);
  end
 end
 event1=EVENT_START_SEND;
 event2=EVENT_START_REC;
```

```
  insert_event(event1,source,time);
  insert_event(event2,dest,time);
    time=time+gettime(len,1,chan,dest);
  event1=EVENT_END_SEND;
  event2=EVENT_END_REC;
  insert_event(event1,source,time);
  insert_event(event2,dest,time);
  insert_syst_events(dest,time);
  if tes_alts(chan,time)==0
    insert_syst_events(source,time);
    TIME(source)=time;
  end
  TIME(dest)=time;
else
  % the other process is not waiting to send
  sig_w_re(chan,number_of_variable);
  [dest,source]=processes_id(chan);
  time=TIME(dest)+gettime(flo,0,chan,dest);
  event=EVENT_WANTS_TO_REC;
  insert_event(event,dest,time);
  TIME(dest)=time;
end


function flag=wait_rec(chan)

%   This function returns a boolean value which indicates if the channel
%   is waiting to receive a message

global CHANNEL

flag=CHANNEL(chan,4)==1;


function flag=wait_sen(chan)

%   This function returns a boolean value which indicates if the channel
%   is waiting to send a message

global CHANNEL

flag=CHANNEL(chan,3)==1;
```

ANEXO D

Programa Sequencial em *Transputer*

O programa sequencial a seguir foi implementado utilizando o compilador 3L Parallel C versão 2.2.4 num Transputer para obtenção do seu tempo de execução.

```
/* sequential file in transputer */

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <thread.h>
#include "matmacro.h"
#include "matcalc.h"
#include "matio.h"
#include <timer.h>
#define linhas 100

float to_ms()
  { if (thread_priority()==0) return 1000;
    else return 15.625;}

float modulo(float n)
{
 if (n < 0) n = -n;
 return(n);
}

float norm(matrix u, int l)
{
  float n=0.0;
  float soma=0.0;
  float soma_r=0.0;
  int i;
  for (i=1;i<=l;i++){
        n=modulo(ELEMENTO(u, i, 1));
        /*getchar();
        printf("n=%f i=%d l=%d \n", n, i, l);*/
        if (n < 0) n = -n;
        soma = soma+n*n;
  }
  soma_r=sqrt(soma);
  return(soma_r);
}
int sign(float u)
{
   if (u>0)
     return 1;
   if (u==0)
     return 0;
   if (u<0)
     return -1;
}
main()
{
int loop,loop2;
float inicio,fim, result;
int dc, dimlin, dimcol, rr,  i;
```

```
float sig, sigg, den, num, consta;
int cont=1, num_working=1, rem, n, num_col_proc;
int  contad=0;
matrix az=NULL;
matrix r=NULL;
matrix s=NULL;
matrix rz=NULL;
matrix ry=NULL;
matrix rw=NULL;
matrix y=NULL;
matrix yz=NULL;
matrix u=NULL;
matrix w=NULL;
matrix wz=NULL;
matrix con=NULL;
matrix ww=NULL;
matrix x=NULL;
matrix b1=NULL;
matrix b2=NULL;
printf("\n LINHAS:\n");
scanf("%d",&dimlin);
while (cont)
{
 printf("\n COLUNAS:\n");
 scanf("%d",&dimcol);
 dc=dimcol;
 n=dimcol/num_working;
 rem=dimcol-n*num_working;
 if (rem !=0)
   printf("aaa\n");
 else
   cont=0;
}
num_col_proc=dimcol/num_working;
u=set_matrix(NULL,dimlin,1);
matrix_zeros(u);
w=set_matrix(NULL,dimlin,1);
wz=set_matrix(NULL,dimlin,1);
az=set_matrix(NULL,dimlin,dimcol-1);
load_matrix(r,"1.2,1.3,1.4,2.1,2.2,3.3,2.1,2.1,2.3,2.3,4.1,3.9,3.7,3.2,4.0,5.5,4.5,6.1,4.1,5.6,7.8,9.9,4,6,7.1,1.
1.1.1,1.1,1.1,1.1,1.1,1.1,1.1,2.2,2.2,2.2,3.3,3.3,3.3,4.4,5.5,5.5,5.5,5.5,5.5,5.5,5.5,1.1,1.1!");
y=set_matrix(NULL,dimlin,1);
yz=set_matrix(NULL,dimlin,1);
load_matrix(y,"1.1,1.2,1.3,2.0,5.2!");
r=set_matrix(NULL,dimlin,dimcol);
s=set_matrix(NULL,dimlin,dimcol);
print_matrix(y);
rz=set_matrix(NULL,dimlin,1);
ry=set_matrix(NULL,dimlin,dimcol-1);
print_matrix_full(ry);
getchar();
rw=set_matrix(NULL,dimlin,1);
con=set_matrix(NULL,1,1);
ww=set_matrix(NULL,1,dimcol-1);
x=set_matrix(NULL,dimcol,1);
```

```
matrix_zeros(x);
b1=set_matrix(NULL,1,1);
b2=set_matrix(NULL,1,1);
print_matrix_full(ry);
getchar();
/*mcopy(a,r);*/
inicio=(float)timer_now();
/*printf("inicio=%d \n", inicio); */
for (rr=0;rr<=dimcol-1;rr++){
  submatrix(r, u, rr+1, rr+1,(int)(NLINHAS(r)), rr+1);
  sig=norm(u,(int)(NLINHAS(u)));
  sigg=-sign(ELEMENTO(u,1,1))*sig;
 /* den=sig*(sig+modulo(ELEMENTO(u,1,1)));*/
 num=1/sqrt(sig*(sig+modulo(ELEMENTO(u,1,1))));
  ELEMENTO(u,1,1)=sign(ELEMENTO(u,1,1))*(sig+modulo(ELEMENTO(u,1,1)));
  /*num=1/sqrt(den);*/
  nummult(num,u,w,0);
  submatrix(y, yz, rr+1, 1,(int)(NLINHAS(y)),1);
  mmult(w,yz,con,1,0);
  consta=ELEMENTO(con, 1,1);
  nummult(consta,w,wz,0);
  MSUB(yz,wz,yz);
  submatrix2(yz, y, 1, 1,(int)(NLINHAS(yz)), 1, rr+1, 1);
  if (rr !=dimcol-1){
/*ww=set_sub_matrix(ww,1,dc-1,1);*/
MATRIX_HEADER(ww,1,dc-1,1);
matrix_zeros(ww);
/*printf("7----------\n");*/
/*ry=set_sub_matrix(ry,dimlin,dc-1,1); */
MATRIX_HEADER(ry,dimlin,dc-1,1);
matrix_zeros(ry);
/*printf("8----------\n");*/
/*az=set_sub_matrix(az,dimlin,dc-1,1);*/
MATRIX_HEADER(az,dimlin,dc-1,1);
matrix_zeros(az);
 submatrix(r, ry, rr+1, rr+2,(int)(NLINHAS(r)),(int)(NCOLUNAS(r)));
   mmult(w,ry,ww,1,0);
   mmult(w,ww,az,0,0);
   MSUB(ry,az,ry);
   submatrix2(ry, r, 1, 1,(int)(NLINHAS(ry)),(int)(NCOLUNAS(ry)), rr+1, rr+2);
  }
  ELEMENTO(r,rr+1,rr+1)=sigg;
  if (rr< dimcol-1){
    for (loop2=1;loop2<=rr+1;loop2++)
     for (loop=loop2+1;loop<=NLINHAS(r);loop++)
       ELEMENTO(r,loop,loop2)=0;
  }
dimlin=dimlin-1;
dc=dc-1;
if (dimlin>0)
/*u=set_sub_matrix(u,dimlin,1,1); */
MATRIX_HEADER(u,dimlin,1,1);
matrix_zeros(u);
/*w=set_sub_matrix(w,dimlin,1,1);  */
MATRIX_HEADER(w,dimlin,1,1);
matrix_zeros(w);
```

```
/*yz=set_sub_matrix(yz,dimlin,1,1);*/
MATRIX_HEADER(yz,dimlin,1,1);
matrix_zeros(yz);
/*wz=set_sub_matrix(wz,dimlin,1,1);*/
MATRIX_HEADER(wz,dimlin,1,1);
matrix_zeros(wz);
/* } */
}
fim=(float)timer_now();
result=(float)(fim-inicio)/ to_ms();
printf("\n %f%s\n",result,"miliseg");
ELEMENTO(x,dimcol,1)=ELEMENTO(y,dimcol,1)/ELEMENTO(r,dimcol,dimcol);
for (i=dimcol-1;i>=1;i--){
  b1=set_matrix(b1,1,dimcol-i);
  matrix_zeros(b1);
  b2=set_matrix(b2,dimcol-i,1);
  matrix_zeros(b2);
  submatrix(r, b1, i, i+1, i, dimcol);
  submatrix(x, b2, i+1, 1,(int)(NLINHAS(x)), 1);
  mmult(b1,b2,con,0,0);
  consta=ELEMENTO(con,1,1);
  ELEMENTO(x,i,1)=(ELEMENTO(y,i,1)-consta)/ELEMENTO(r,i,i);
}
/*fim=(float)timer_now();
result=(float)(fim-inicio)/ to_ms();
printf("\n %f%s\n",result,"miliseg");*/
return(0);
}
```

ANEXO E

Programa Paralelo em *Transputers*

O código do programa paralelo implementado na plataforma paralela real de *Transputers* é apresentado neste anexo.

O ficheiro de configuração da topologia contituída por *Transputers* é o seguinte:

```
! configurer file for parallel transputer system
! Hardware
processor host
processor root
processor root2
processor root3
wire ?  root[0] host[0]
wire ?  root[2] root2[1]
wire ?  root2[2] root3[1]

!Software
task afserver ins=1 outs=1
task filter  ins=2 outs=2 data=10k
task ro1 ins=3 outs=3 urgent  data=400k
task ro2 ins=2 outs=2 urgent  data=400k
task ca1_1 file = "ca1.b4" ins=2 outs=1    data=400k
task ca1_2 file = "ca1.b4" ins=2 outs=1    data=400k

bind input ca1_1[1] value = 3
bind input ca1_2[1] value = 4

place afserver host
place filter root
place ro1 root2
place ca1_1 root2
place ro2 root3
place ca1_2 root3

connect ? afserver[0] filter[0]
connect ? filter[0] afserver[0]
connect ? filter[1] ro1[1]
connect ? ro1[1] filter[1]
connect ? ro1[0] ca1_1[0]
connect ? ca1_1[0] ro1[0]
connect ? ro1[2] ro2[1]
connect ? ro2[1] ro1[2]
connect ? ro2[0] ca1_2[0]
connect ? ca1_2[0] ro2[0]
```

A seguir é apresentado o código do processo *router1* (*ro1*) de alta prioridade situado no *transputer root2*:

```
/* ro1 in parallel transputer system */

#include <chan.h>
#include <stdio.h>
#include <alt.h>
#include <math.h>
#include <stdlib.h>
#include "matmacro.h"
#include "matcalc.h"
#include "matio.h"
#include <timer.h>
#include <thread.h>

float to_ms()
  { if (thread_priority()==0) return 1000;
     else return 15.625;}

main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
  int proc_numb=1;

  float inicio,fim, result=0;
  int lenghta=0, tamu=0, lena=0,  lenu=0, tam=0, tamuu=0, le=0;
  int i=1,  cont=1, rem, n, num_col_proc=0, num_working=2;
  int dimlin, dimcol;

  matrix a=NULL;
  matrix u=NULL;
  matrix b=NULL;

  printf("\n LINHAS:\n");
  scanf("%d",&dimlin);
  while (cont)
  {
   printf("\n COLUNAS:\n");
   scanf("%d",&dimcol);
  printf("%d\n", dimcol);
   n=dimcol/num_working;
   rem=dimcol-n*num_working;
   printf("%d\n", rem);
   if (rem!=0)
    printf("\n");
   else
    cont=0;
   }

  u=set_matrix(NULL,dimlin,1);

  matrix_zeros(v);
  num_col_proc=dimcol/num_working;
printf("1:\n");
```

```
printf("num_col_proc=%d\n", num_col_proc);
getchar();
  a=set_matrix(NULL,dimlin,dimcol);
  matrix_zeros(a);
printf("1:\n");
print_matrix_full(a);
getchar();
  load_matrix(a,"1.2, 1.3, 1.4, 2.9, 2.1,2.2,3.3,3.9,2.1,2.1,2.3,4.9,2.3,4.1,3.9,4.5, 5.5, 4.5, 6.1,4.1,5.6, 7.8,
,9.9, 4.6,7.1,1.1,1.1,1.1,1.1,1.1,1.1,1.1,1.1,1.1,2.2,2.2,2.2,3.3,3.3,4.4, 5.5,5.5,5.5,5.5,5.5,5.5,5.5,1.1,1.1 !");
printf("1:\n");
print_matrix_full(a);
getchar();
   b=set_matrix(NULL,dimlin,num_col_proc);
  matrix_zeros(b);
printf("1:\n");
/*print_matrix_full(b); */
getchar();

  chan_out_word(num_col_proc,out_ports[0]);
printf ("Ro1 : Send to calc1_1 1\n");
  chan_out_word(dimlin,out_ports[0]);
printf("Ro1 : Send to calc1_1 2\n");
  lenghta = dimlin*dimcol;
printf("lenghta %d \n", lenghta);
  lena=lenghta*4;
  chan_out_word(lena, out_ports[0]);
printf("Ro1 :send to calc1_1 3\n");
  submatrix(a,b,1,1,dimlin,dimcol/2);
  chan_out_message(lena,(b+MDATA), out_ports[0]);
printf("Ro1  send to calc1_1 4\n");
print_matrix_full(b);
getchar();

  chan_out_word(dimcol,out_ports[2]);
printf ("Ro1 : Send to ro2 1\n");
  chan_out_word(num_col_proc,out_ports[2]);
printf("Ro1 : Send to ro2 2\n");
  chan_out_word(dimlin,out_ports[2]);
printf("Ro1 : Send to ro2 3\n");
  chan_out_word(lena, out_ports[2]);
printf("Ro1 : Send to ro2 4\n");
  submatrix(a,b,1,dimcol/2+1, dimlin, dimcol);
  chan_out_message(lena,(b+MDATA), out_ports[2]);
printf("Ro1 : Send to ro2 5\n");
/*print_matrix_full(b); */
getchar();

  inicio=(float) timer_now();
  while (i <= (int)(dimcol-num_working+proc_numb))
 /*while (i < dimcol)*/
  {
   int c;
   /*printf("Ro1 i=%d: IN wHILE 1 \n",i);*/
   c = alt_wait(2, in_ports[0], in_ports[2]);
   /*printf("c=%d \n",c);*/
   if (c == 0)
```

```
{

chan_in_word(&lenu, in_ports[0]); /*is lenght u*/
/*printf("ro1 receive from calc1_1 lenu=%d  \n",lenu);*/
 tam=lenu/4;
/*u=set_sub_matrix(u,tam,1,1);*/
 MATRIX_HEADER (u,tam,1,1);

chan_in_message(lenu, (u+MDATA), in_ports[0]); /* is vector u*/
/*printf("ro1 receive from calc1_1 u\n");
print_matrix(u);*/
if (i<dimcol){
 chan_out_word(lenu, out_ports[2]);    /* is lenght  u*/
 /*printf("ro1 send to ro2  lenu=%d\n", lenu); */
 chan_out_message(lenu, (u+MDATA), out_ports[2]);  /* is vector u*/
 /*printf("ro1 send to ro2 u\n");*/
}
}

else
{
 chan_in_word(&lenu, in_ports[2]); /*is lenght u*/
 /*printf("ro1 receive from ro2 lenu=%d  1\n",lenu); */
   tam=(int)lenu/4;
 /*printf("tam=%d\n",tam);*/
/* u=set_sub_matrix(u,tam,1,1);*/
 MATRIX_HEADER (u,tam,1,1);

 chan_in_message(lenu, (u+MDATA), in_ports[2]);  /* is vector u*/
 /*printf("ro1 receive from ro2 vector u\n");*/
 /*print_matrix(u);*/

 if (i<dimcol){
  tamu=tam-1;
  /*printf("tamu=%d\n",tamu);
  getchar();*/
  tamuu=(int)tamu*4;
  /*printf("tamuu=%d\n",tamuu);
  getchar();*/
  chan_out_word(tamuu, out_ports[0]);  /* is lenght u*/
  /*printf("Ro1 send to calc_1 lenght u= %d\n",tamuu);
  getchar();*/
  u++;
   MATRIX_HEADER (u,tamu,1,1);
   chan_out_message(tamuu, (u+MDATA), out_ports[0]); /* is vector u*/
  /*printf("Ro1 send to calc_1 verctor u\n");*/
 }
 }

 i=i+1;

}
 fim=(float) timer_now();
 result=(float)(fim-inicio)/to_ms();
 printf("Ro1: %f %s\n",result,"milisegundos");
}
```

A seguir é apresentado o código do processo *router2* (*ro2*) de alta prioridade situado no *transputer root3*:

```
/* ro2 in parallel transputer system */

#include <chan.h>
#include <alt.h>
#include <math.h>
#include <stdlib.h>
#include "matmacro.h"
#include "matcalc.h"
#include <timer.h>
#include <thread.h>

main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
int proc_numb=2;
  int  tamu, lena=0,  lenu, tam, tamuu;
  int i=1,  cont=1, num_col_proc=0, num_working=2;
  int dimlin, dimcol;

  matrix a=NULL;
  matrix u=NULL;
  matrix b=NULL;
  u=set_matrix(NULL,dimlin,1);
  matrix_zeros(u);

   chan_in_word(&dimcol, in_ports[1]);/*dimension of columns matrix a*/
   chan_in_word(&num_col_proc, in_ports[1]);/*number of columns matrix b*/
  chan_in_word(&dimlin,in_ports[1]);/* dimensions of lines matrix a and b*/
  b=set_matrix(NULL,dimlin,num_col_proc);
  chan_in_word(&lena, in_ports[1]);/* lenght matrix b*/
  chan_in_message(lena,(b+MDATA), in_ports[1]);/*is matrix b*/

/*the same information is send to calc_2*/
  chan_out_word(num_col_proc,out_ports[0]);
  chan_out_word(dimlin,out_ports[0]);
  chan_out_word(lena, out_ports[0]);
  chan_out_message(lena,(b+MDATA), out_ports[0]);


  while (i <= (int)(dimcol-num_working+proc_numb))
 /*while (i < dimcol)*/
  {
  int c;
  c = alt_wait(2, in_ports[0], in_ports[1]);
  if (c == 0)
  {
  lenu=0;
  tam=0;
  chan_in_word(&lenu, in_ports[0]); /* ro2 receive lenght u from calc_2*/
   tam=(int)lenu/4;
```

```
  /*u=set_sub_matrix(u,tam,1,1);*/
  MATRIX_HEADER (u,tam,1,1);
  chan_in_message(lenu, (u+MDATA), in_ports[0]); /* ro2 receive vector u from calc_2*/

  if (i<dimcol){
   chan_out_word(lenu, out_ports[1]); /*ro2 send lenght u to ro1*/
   chan_out_message(lenu, (u+MDATA), out_ports[1]);  /*ro2 send vector u to ro1*/
  }
  }
  else
  {
   chan_in_word(&lenu, in_ports[1]); /* ro2 receive lenght u from ro1*/
   tam=lenu/4;
   /*u=set_sub_matrix(u,tam,1,1); */
   MATRIX_HEADER (u,tam,1,1);
   chan_in_message(lenu, (u+MDATA), in_ports[1]); /* ro2 receive vector u from ro1*/
   if (i<dimcol){
    tamu=tam-1;
    tamuu=tamu*4;
    chan_out_word(tamuu, out_ports[0]); /*ro2 send lenght u to calc_2*/
     u++;
    MATRIX_HEADER (u,tamu,1,1);
    chan_out_message(tamuu, (u+MDATA), out_ports[0]); /*ro2 send vector u to calc_2*/
   }
  }
i=i+1;
}

}
```

É apresentado o código dos processos de cálculo de baixa prioridade *ca1_1* e *ca1_2* situados nos transputers *root2* e *root3* respectivamente:

```
/* ca1 in parallel transputer system */

#include <math.h>
#include <chan.h>
/*#include <stdio.h>*/
#include <alt.h>
#include <stdlib.h>
#include "matmacro.h"
#include "matcalc.h"
/*#include "matio.h"*/
#include <timer.h>
#include <thread.h>

float to_ms()
  { if (thread_priority()==0) return 1000;
      else return 15.625;}

float modulo(float n)
{
 if (n < 0) n = -n;
 return(n);
```

```
}

float norm(matrix u, int l)

{   —
 register float n;
 register float soma=0.0;
/*  float soma_r=0.0;*/
 register int i;


 for (i=1;i<=l;i++){
      n=modulo(ELEMENTO(u, i, 1));
      if (n < 0) n = -n;
      soma = soma+n*n;
 }
return(sqrt(soma));
}

int sign(float u)
{
  if (u>0)
    return 1;
  if (u==0)
    return 0;
  if (u<0)
    return -1;
}

main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
int proc_numb = (int) in_ports[1];
int loop,loop2;
int sweep=1, next_column_in_sweep=1, num_working=2, first_row=1;
float sig, sigg, den, num;
int cont=1, num_col_proc, dimlin;
int   contad=0, lena, lenghtu, tamuu, lenu=0, tamuuu;

matrix az=NULL;
matrix ry=NULL;
matrix u=NULL;
matrix w=NULL;
matrix ww=NULL;
matrix b=NULL;

 chan_in_word(&num_col_proc,in_ports[0]); /*ca1 receive from ro1 or ro2 number columns b*/
 chan_in_word(&dimlin,in_ports[0]); /*ca1 receive from ro1 or ro2 lines of b*/
 chan_in_word(&lena, in_ports[0]); /* ca1 receive from ro1or ro2 lenght b*/

 b=set_matrix(NULL, dimlin,num_col_proc);
 matrix_zeros(b);
 chan_in_message(lena,(b-MDATA),in_ports[0]);/*ca1 receive from ro1 or ro2 matrix b*/
u=set_matrix(NULL,dimlin,1);
matrix_zeros(u);
w=set_matrix(NULL,dimlin,1);
```

```
matrix_zeros(w);
az=set_matrix(NULL,dimlin,num_col_proc);
ry=set_matrix(NULL,dimlin,num_col_proc);
matrix_zeros(ry);
ww=set_matrix(NULL,1,num_col_proc);

while (sweep<=num_col_proc)
{
 if (fmod(next_column_in_sweep, num_working)==fmod((int)(proc_numb-3), num_working))
 /*if (next_column_in_sweep==proc_numb)*/
 {
  submatrix(b, u, first_row, sweep, dimlin, sweep); /*is vector u*/
  lenghtu=(int)NLINHAS(u);/*is lenght u*/
  sig=norm(u,lenghtu);/*norm u*/
  sigg=-sign(ELEMENTO(u,1,1))*sig;
  den=sig*(sig+modulo(ELEMENTO(u,1,1)));
  ELEMENTO(u,1,1)=sign(ELEMENTO(u,1,1))*(sig+modulo(ELEMENTO(u,1,1)));
  num=1/sqrt(den);
  nummult(num,u,u,0);
  ELEMENTO(b,first_row,sweep)=sigg;
  if (first_row< dimlin)
    {
    for (loop2=1;loop2<=sweep;loop2++)
      for (loop=loop2+1;loop<=dimlin;loop++)
        ELEMENTO(b,loop,loop2)=0;
    }
sweep=sweep+1;
lenu=(int)lenghtu*4;
  chan_out_word(lenu,out_ports[0]);/*ca1 send to ro1 or ro2 lenght u*/
  chan_out_message(lenu,(u+MDATA), out_ports[0]);/*ca1 send to ro1 or ro2 vector u*/
 }
 else
 {
  chan_in_word(&tamuu, in_ports[0]); /*calc_1 receive from ro1 lenght u*/
  tamuuu=(int)tamuu/4;
  /*u=set_sub_matrix(u,tamuuu,1,1);*/
  MATRIX_HEADER(u,tamuuu,1,1);
  chan_in_message(tamuu,(u+MDATA),in_ports[0]); /*ca1 receive from ro1 or ro2 vector u*/
 }
 if (sweep <=num_col_proc){
    submatrix(b, ry, first_row, sweep, dimlin, num_col_proc);
    mmult(u,ry,ww,1,0);
    mmult(u,ww,az,0,0);
    MSUB(ry,az,ry);
    submatrix2(ry, b, 1, 1,(int)(NLINHAS(ry)), (int)(NCOLUNAS(ry)), first_row, sweep);
    first_row =first_row+1;
    next_column_in_sweep=next_column_in_sweep+1;
    if (next_column_in_sweep>num_working)
      next_column_in_sweep=1;
 }
}/*end do while*/
}
```

ANEXO F

Programa de comunicações entre *Transputers*

O código do programa de comunicações entre *Transputers* é o apresentado neste anexo.

O ficheiro de configuração da topologia contituída por *Transputers* é o seguinte:

```
! configurer file for determine comunication time between t8/t8
! Hardware
processor host
processor root
processor root2
processor root3
wire ?  root[0] host[0]
wire ?  root[2] root2[1]
wire ?  root2[2] root3[1]

!Software
task afserver ins=1 outs=1
task filter  ins=2 outs=2 data=10k
task master ins=2 outs=2  urgent data=100k
task slave ins=1 outs=1  data=100k

place afserver host
place filter root
place master root2
place slave root3

connect ? afserver[0] filter[0]
connect ? filter[0] afserver[0]
connect ? filter[1] master[1]
connect ? master[1] filter[1]
connect ? master[0] slave[0]
connect ? slave[0] master[0]
```

Os programas a seguir permitem determinar tempos de comunicação na transmissão de um vector de reais entre *Transputers*. O primeiro programa, o *Master* envia para o segundo o *Slave* cada um dos elementos de um vector. Depois do slave receber envia de novo os elementos para o *Master*:

```
/*Master*/
/* Times for transmit flots between parallel transputer system */
#include <stdlib.h>
#include <timer.h>
#include <stdio.h>
#include <chan.h>

main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
  {
   float buffer [100], tempo [100];
   int dimlin=100, m;
   int inicio, fim, i=1;
for (i=dimlin; i>=1; i-=10){
    m=i*4;
```

```
    inicio = timer_now();
    chan_out_message(m,buffer, out_ports[0]);
    chan_in_message(m,buffer, in_ports[0]);
    fim = timer_now();
    tempo[i] = (float)(fim-inicio);
    printf("\nTempo total: %f %s i=%d \n",(float)tempo[i], " us", i);
    /*getchar();*/
   }
  }


/*Slave*/
/* Times for transmit flots between parallel transputer system */
#include <chan.h>

main(int argc, char *argv[], char *envp[],
     CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
  {
   float buffer[100];
   int m, i , dimlin=100;
  for (i=dimlin; i>=1; i-=10){
   m=i*4;
   chan_in_message(m,buffer, in_ports[0]);
   chan_out_message(m,buffer, out_ports[0]);
  }

  }
```

ANEXO G

Programa Sequencial em *C40*

O programa a seguir foi implementado em Parallel C versão 2.0.2 em um C40 para obtenção do tempo de execução sequencial do algoritmo:

```
/* Sequential file in C40 system*/

#include <math.h>
#include <c40tim.h>
#include <stdio.h>
#include <stdlib.h>
#include "matmacro.h"
#include "matcalc.h"
#include "matio.h"
#include <timer.h>
#define linhas 100
#define to_ms timer_rate()*1000

float modulo(float n)
{
 if (n < 0) n = -n;
 return(n);
}

float norm(matrix u, int l)
{
  float n=0.0;
  float soma=0.0;
  float soma_r=0.0;
  int i;
  for (i=1;i<=l;i++){
        n=modulo(ELEMENTO(u, i, 1));
        /*getchar();
        printf("n=%f i=%d l=%d \n", n, i, l);*/
        if (n < 0) n = -n;
        soma = soma+n*n;
  }
  soma_r=sqrt(soma);
  return(soma_r);
}

int sign(float u)
{
   if (u>0)
     return 1;
   if (u==0)
     return 0;
   if (u<0)
     return -1;

}
main()
{
int loop,loop2;
float inicio,fim, result;
```

```
int dc, dimlin, dimcol, rr, ss, tt, vv, i, j;
int f, FF, ff1, ff;
float sig, sigg, den, num, consta;
int cont=1, num_working=1, rem, n, num_col_proc;
int k, m, contad=0;
matrix a=NULL;
matrix az=NULL;
matrix r=NULL;
matrix s=NULL;
matrix rz=NULL;
matrix ry=NULL;
matrix rw=NULL;
matrix y=NULL;
matrix yz=NULL;
matrix u=NULL;
matrix w=NULL;
matrix wz=NULL;
matrix con=NULL;
matrix ww=NULL;
matrix x=NULL;
matrix b1=NULL;
matrix b2=NULL;
printf("\n LINHAS:\n");
scanf("%d",&dimlin);
while (cont)
{
 printf("\n COLUNAS:\n");
 scanf("%d",&dimcol);
 dc=dimcol;
 n=dimcol/num_working;
 rem=dimcol-n*num_working;
 if (rem !=0)
   printf("aaa\n");
 else
   cont=0;
}
num_col_proc=dimcol/num_working;
u=set_matrix(NULL,dimlin,1);
matrix_zeros(u);
w=set_matrix(NULL,dimlin,1);
wz=set_matrix(NULL,dimlin,1);
a=set_matrix(NULL,dimlin,dimcol);
az=set_matrix(NULL,dimlin,dimcol-1);
load_matrix(a,"1.2,1.3,1.4,2.1,2.2,3.3,2.1,2.1,2.3,2.3,4.1,3.9,3.7,3.2,4.0,5.5,4.5,6.1,4.1,5.6,7.8,9.9,4,6,7.1,1.
1,1.1,1.1,1.1,1.1,1.1,1.1,1.1,2.2,2.2,2.2,3.3,3.3,4.4,5.5,5.5,5.5,5.5,5.5,5.5,1.1,1.1!");
y=set_matrix(NULL,dimlin,1);
yz=set_matrix(NULL,dimlin,1);
load_matrix(y,"1.1,1.2,1.3,2.0,5.2!");
r=set_matrix(NULL,dimlin,dimcol);
s=set_matrix(NULL,dimlin,dimcol);
print_matrix(a);
print_matrix(y);
rz=set_matrix(NULL,dimlin,1);
ry=set_matrix(NULL,dimlin,dimcol-1);
print_matrix_full(ry);
getchar();
```

```
rw=set_matrix(NULL,dimlin,1);
con=set_matrix(NULL,1,1);
ww=set_matrix(NULL,1,dimcol-1);
x=set_matrix(NULL,dimcol,1);
matrix_zeros(x);
b1=set_matrix(NULL,1,1);
b2=set_matrix(NULL,1,1);
print_matrix_full(ry);
getchar();
/*mcopy(a,r);*/

inicio=(float) timer_now();
for (rr=0;rr<=dimcol-1;rr++){
    submatrix(r, u, rr+1, rr+1, NLINHAS(r), rr+1);
    sig=norm(u,NLINHAS(u));
  sigg=-sign(ELEMENTO(u,1,1))*sig;
  den=sig*(sig+modulo(ELEMENTO(u,1,1)));
  ELEMENTO(u,1,1)=sign(ELEMENTO(u,1,1))*(sig+modulo(ELEMENTO(u,1,1)));
   num=1/sqrt(den);
   nummult(num,u,w,0);
   submatrix(y, yz, rr+1, 1, NLINHAS(y),1);
   mmult(w,yz,con,1,0);
   consta=ELEMENTO(con, 1,1);
   nummult(consta,w,wz,0);
   MSUB(yz,wz,yz);
   submatrix2(yz, y, 1, 1, NLINHAS(yz), 1, rr+1, 1);
  if (rr !=dimcol-1){
    MATRIX_HEADER(ww,1,dc-1,1);
    matrix_zeros(ww);
    MATRIX_HEADER(ry,dimlin,dc-1,1);
     matrix_zeros(ry);
     MATRIX_HEADER(az,dimlin,dc-1,1);
    matrix_zeros(az);
     submatrix(r, ry, rr+1, rr+2, NLINHAS(r), NCOLUNAS(r));
    mmult(w,ry,ww,1,0);
    mmult(w,ww,az,0,0);
     MSUB(ry,az,ry);
     submatrix2(ry, r, 1, 1, NLINHAS(ry), NCOLUNAS(ry), rr+1, rr+2);
  }
printf(" \n");
   ELEMENTO(r,rr+1,rr+1)=sigg;
  if (rr< dimcol-1){
     for (loop2=1;loop2<=rr+1;loop2++)
       for (loop=loop2+1;loop<=NLINHAS(r);loop++)
         ELEMENTO(r,loop,loop2)=0;
  }

dimlin=dimlin-1;
dc=dc-1;
if (dimlin>0)
{
MATRIX_HEADER(u,dimlin,1,1);
matrix_zeros(u);
MATRIX_HEADER(w,dimlin,1,1);
matrix_zeros(w);
MATRIX_HEADER(yz,dimlin,1,1);
```

```
matrix_zeros(yz);
MATRIX_HEADER(wz,dimlin,1,1);
matrix_zeros(wz);
 }
}
fim=(float)timer_now();
result=(float)(fim-inicio)/ to_ms;
printf("\n %f%s\n",result,"miliseg");

ELEMENTO(x,dimcol,1)=ELEMENTO(y,dimcol,1)/ELEMENTO(r,dimcol,dimcol);
for (i=dimcol-1;i>=1;i--){
  b1=set_matrix(b1,1,dimcol-i);
  matrix_zeros(b1);
  b2=set_matrix(b2,dimcol-i,1);
  matrix_zeros(b2);
  submatrix(r, b1, i, i+1, i, dimcol);
  submatrix(x, b2, i+1, 1, NLINHAS(x), 1);
  mmult(b1,b2,con,0,0);
  consta=ELEMENTO(con,1,1);
  ELEMENTO(x,i,1)=(ELEMENTO(y,i,1)-consta)/ELEMENTO(r,i,i);
}

}
```

ANEXO H

Programa Paralelo em *C40s*

O ficheiro de configuração da topologia contituída por *C40s* é o seguinte:

```
! configurer file for parallel C40 system
! total of 2 processors found
processor ROOT type=TMS320C40
processor P1 type=TMS320C40
wire ?  ROOT[0] P1[3]
wire ?  ROOT[2] P1[5]
!Software configuration
task ro1a ins=2 outs=2 priority=0  stack=20k heap=800k opt=code opt=code:ramblk0 opt=heap:ramblk1
task ro2 ins=2 outs=2 priority=0 stack=20k heap=100k opt=code opt=code:ramblk0  opt=heap:ramblk1
task ca1_1 file = "ca1.tsk" ins=2 outs=1   priority=1 stack=1.1k heap=100k opt=code opt=code:ramblk0
opt=heap:ramblk1


task ca1_2 file = "ca1.tsk" ins=2 outs=1   priority=1 stack=1.1k heap=100k opt=code opt=code:ramblk0
opt=heap:ramblk1


bind input consuma_1[1] value = 3
bind input consuma_2[1] value = 4

place ro1  ROOT
place ro2  P1
place ca1_1 ROOT
place ca1_2 P1

default connect virtual
!UPR MAX = 10000
!UPR BUFFERS = 10

connect ? ro1[0] ca1_1[0]
connect ? ca1_1[0] ro1[0]
connect ? ro1[1] ro2[1]
connect ? ro2[1] ro1[1]
connect ? ro2[0] ca1_2[0]
connect ? ca1_2[0] ro2[0]

!connect ? ro1[1] ca1_2[0]
!connect ? ca1_2[0] ro1[1]
```

A seguir é apresentado o código do processo *router1* (*ro1*) de alta prioridade situado no *transputer ROOT*:

```
/* ro1 in parallel C40 system */

#include <chan.h>
#include <c40tim.h>
#include <stdio.h>
#include <alt.h>
#include <math.h>
#include <stdlib.h>
#include "matcalc.h"
```

```
#include "matio.h"
#include <timer.h>
#define to_ms timer_rate()*1000

struct map{
      CHAN *dataready;
      CHAN *phys_chan;
      CHAN *internal_chan;
};

CHAN   datar1, internal1;
void guard(void *arg)
{
 struct map *s =(struct map *)arg;
 int buf;
 for (;;){
    chan_in_word(&buf,s->dataready);
    chan_in_word(&buf, s->phys_chan);
    chan_out_word(buf, s->internal_chan);
 }
}
main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
int proc_numb=1;
  float inicio,fim, result;
  int lenghta, lenghtu=0,   dim, tamu=0, buf;
  int i=1, x, cont=1, rem, n,num_col_proc, num_working=2, m, c;
  int dimlin, dimcol;
  matrix a=NULL;
  matrix u=NULL;
  matrix b=NULL;
  matrix v=NULL;
  struct map s0, s1;

  s1.phys_chan=in_ports[1];
  s1.internal_chan=&internal1;
  s1.dataready=&datar1;
  chan_init(&internal1);
  chan_init(&datar1);
  thread_new(guard, 1024, &s1);
  chan_out_word(i, &datar1);
  printf("\n LINHAS:\n");
  scanf("%d",&dimlin);
  while (cont)
  {
   printf("\n COLUNAS:\n");
   scanf("%d",&dimcol);
   n=dimcol/num_working;
   rem=dimcol-n*num_working;
  if (rem!=0)
    printf("aaaaaaaaaa\n");
   else
    cont=0;
   }
  u=set_matrix(NULL,dimlin,1);
```

```
  v=set_matrix(NULL,dimlin,1);
  matrix_zeros(v);
  num_col_proc=dimcol/num_working;
  a=set_matrix(NULL,dimlin,dimcol);
  matrix_zeros(a);
  load_matrix(a,"1.2, 1.3, 1.4, 2.9, 2.1,2.2,3.3,3.9,2.1,2.1,2.3,4.9,2.3,4.1,3.9,4.5,
5.5,4.5,6.1,4.1,5.6,7.8,9.9,4,6,7.1,1.1,1.1,1.1,1.1,1.1,1.1,1.1,1.1,2.2,2.2,2.2,3.3,
3.3,4.4,5.5,5.5,5.5,5.5,5.5,5.5,5.5,1.1,1.1!");

  b=set_matrix(NULL,dimlin,num_col_proc);
  matrix_zeros(b);

  chan_out_word(num_col_proc,out_ports[0]);
/*printf("Ro1 send to cal num_col_proc=%d\n", num_col_proc);  */
  chan_out_word(dimlin,out_ports[0]);
/*printf("Ro1 send to ca1 dimlin=%d\n", dimlin);*/
  lenghta = (int)dimlin*num_col_proc;
  chan_out_word(lenghta, out_ports[0]);
/*printf("Ro1 send to ca1 lenghta=%d\n",lenghta);*/
  submatrix(a,b,1,1,dimlin,dimcol/2);
  chan_out_message(lenghta,(b+MDATA), out_ports[0]);
/*printf("Ro1 send to ca1 matrix b\n");  */
/*print_matrix_full(b);
getchar();*/

  chan_out_word(num_col_proc,out_ports[1]);
/*printf("Ro1 send to  Ro2 num_col_proc=%d\n", num_col_proc);*/
  chan_out_word(dimlin,out_ports[1]);
/*printf("Ro1 send to Ro2 dimlin=%d\n", dimlin);*/
  chan_out_word(lenghta, out_ports[1]);
/*printf("Ro1 send to Ro2 lenghta=%d\n", lenghta);  */
  submatrix(a,b,1,dimcol/2+1, dimlin, dimcol);
  chan_out_message(lenghta,(b+MDATA), out_ports[1]);
/*printf("Ro1 send to Ro2 matrix b\n");  */
  chan_out_word(dimcol,out_ports[1]);
/*printf("Ro1 send to Ro2 dimcol=%d\n", dimcol); */

  /* sync */
 /*  chan_in_word(&buf, out_ports[0]); */
  /* chan_in_word(&buf, out_ports[1]); */

inicio=(float) timer_now();
while (i <= (int)(dimcol-num_working+proc_numb))
/*while (i <= dimcol)  */
  {
   int c;
/*printf("ROUTER(1) i=%d entrou no WHILE ",i); */
   c = alt_wait(2, in_ports[0], &internal1);
  /* printf(" tem o c=%d\n",c);*/
   if (c == 0)
   {

  chan_in_word(&lenghtu, in_ports[0]);  /* is lenght u*/
 /*printf("receive from ca1 lenghtu=%d  1\n",lenghtu); */
   MATRIX_HEADER (u,lenghtu,1,1);
  chan_in_message(lenghtu, (u+MDATA), in_ports[0]); /* is vector u*/
```

```
 /* printf("ro1 receive from ca1 lenghtu=%d e o u\n", lenghtu); */
 /*   print_matrix(u);  */

   if(i<dimcol){
/* printf("ro1 send to ro2 lenghtu =%d\n", lenghtu);*/
   chan_out_word(lenghtu, out_ports[1]);  /* comprimento u*/
  /* printf("ro1 send to ro2 lenghtu =%d\n", lenghtu);*/
   chan_out_message(lenghtu, (u+MDATA), out_ports[1]); /* u*/
  /* printf("ro1 send to ro2 lenghtu=%d e o u \n",lenghtu);*/
   }
   }
   else
   {
/*printf("ro1 is in else waiting for lenght u\n");*/

   chan_in_word(&lenghtu, &internal1);   /* lenght u */
 /*   printf("ro1 receive from ro2 lenghtu=%d  1\n",lenghtu);*/
   /* u=set_sub_matrix(u,lenghtu,1,1); */
   MATRIX_HEADER(u,lenghtu,1,1);
   chan_in_message(lenghtu, (u+MDATA), in_ports[1]);  /*  u*/
/*printf("ro1 receive from ro2 u\n");  */
  /*  print_matrix(u);
   getchar();         */
    /* tell alt data ready */
     chan_out_word(i, &datar1);

   if(i<dimcol){
   tamu=lenghtu-1;
   chan_out_word(tamu, out_ports[0]);  /* comprimento u*/
/*printf("ro1 send to ca1 lenght u=%d\n", tamu);*/
   u++;
   MATRIX_HEADER(u,tamu,1,1);
  /*  print_matrix(u);*/
   chan_out_message(tamu, (u+MDATA), out_ports[0]); /* u*/
  /* printf("ro1 send to ca1 lenght=%d and u\n", tamu);*/
   }
   }
   i=i+1;
  }
   fim=(float) timer_now();
   result=(float)(fim-inicio)/to_ms;
   printf("Router: %f %s\n",result,"milisegundos");

}
```

O código a seguir corresponde ao processo *router2* (*ro2*) de alta prioridade situado no *transputer P1*:

```
/* ro2 in parallel C40 system */


#include <chan.h>
#include <c40tim.h>
```

```
#include <stdio.h>
#include <alt.h>
#include <math.h>
#include <stdlib.h>
#include "matcalc.h"
#include "matio.h"
#include <timer.h>
#define to_ms timer_rate()*1000

struct map{
     CHAN *dataready;
     CHAN *phys_chan;
     CHAN *internal_chan;
};

CHAN datar1, internal1;
void guard(void *arg)
{
 struct map *s =(struct map *)arg;
 int buf;
 int buf2[10];
 for (;;){
    chan_in_word(&buf,s->dataready);
    chan_in_message(1,buf2, s->phys_chan);
    chan_out_message(1,buf2, s->internal_chan);
 }
}

main(int argc, char *argv[], char *envp[],
   CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
int proc_numb=2;
  int lenghta, lenghtu=0,  dim, tamu, buf;
  int i=1, x, cont=1, rem, n,num_col_proc, num_working=2, m, c;
  int dimlin, dimcol;

  matrix a=NULL;
  matrix u=NULL;
  matrix b=NULL;
  matrix v=NULL;
  struct map s0,s1;
  s1.phys_chan=in_ports[1];
  s1.internal_chan=&internal1;
  s1.dataready=&datar1;
  chan_init(&internal1);
  chan_init(&datar1);

/*  thread_new(guard, 1024, &s1);

  chan_out_word(1000, &datar1);  */

  u=set_matrix(NULL,dimlin,1);
  v=set_matrix(NULL,dimlin,1);
  matrix_zeros(v);
  b=set_matrix(NULL,dimlin,num_col_proc);
  chan_in_word(&num_col_proc,in_ports[1]);
```

```
  /*printf("Ro2 receive num_col_proc=%d\n", num_col_proc);*/
    chan_in_word(&dimlin,in_ports[1]);
  /*printf("Ro2 receive from ro1 dimlin=%d\n", dimlin);*/
    chan_in_word(&lenghta, in_ports[1]);
  /*printf("Ro2 receive from ro1 lenghta=%d\n", lenghta); */
    chan_in_message(lenghta,(b+MDATA), in_ports[1]);

    chan_in_word(&dimcol,in_ports[1]);
  /*printf("Ro2 receive from ro1 dimcol=%d\n", dimcol);*/

  chan_out_word(num_col_proc,out_ports[0]);
  /*printf("Ro2 send to calc num_col_proc=%d\n",num_col_proc );*/
    chan_out_word(dimlin,out_ports[0]);
  /*printf("Ro2 send to calc dimlin=%d\n",dimlin ); */
    chan_out_word(lenghta, out_ports[0]);
  /*printf("Ro2 send to calc lenghta=%d\n",lenghta );  */
    chan_out_message(lenghta,(b+MDATA), out_ports[0]);
  /*printf("Ro2 send to calc a matrix b\n"); */
  /*print_matrix_full(b);
  getchar();  */
     /*chan_in_word(&lenghtu, &internal1);*/   /* lenght u */
    /* printf("Ro2 receive from Ro1 o lenghtu=%d \n", lenghtu);*/

    /* sync */
   /* chan_in_word(&buf, out_ports[1]);*/
  while (i <= (int)(dimcol-num_working+proc_numb))
  /*while (i <= dimcol)*/
    {
    int c;
  /*printf("ro2 i=%d in WHILE  ",i); */
  /*    c = alt_wait(2, in_ports[0], &internal1);*/
    c = alt_wait(2, in_ports[0], in_ports[1]);
  /*printf(" c=%d\n",c); */
    if (c == 0)
    {

    chan_in_word(&lenghtu, in_ports[0]);
  /* printf("ro2 receive from ca1_2 lenght u=%d  1\n",lenghtu);  */
    if (lenghtu==0) break;
  /* u=set_sub_matrix(u,lenghtu,1,1); */
    MATRIX_HEADER(u,lenghtu,1,1);
    chan_in_message(lenghtu, (u+MDATA), in_ports[0]); /* u*/
  /*printf("ro2 receive from ca1_2 =%d  u\n", lenghtu);
    print_matrix(u); */

    if(i<dimcol){
    chan_out_word(lenghtu, out_ports[1]);
  /* printf("ro2 send to ro1 lenghtu=%d\n", lenghtu);*/
    chan_out_message(lenghtu, (u+MDATA), out_ports[1]); /* u*/
  /*printf("ro2 send to ro1 lenghtu=%d e o u\n", lenghtu);*/
    }
    }
    else
    {
  /* printf("ro2 in else");*/
```

```
/*printf("  waiting for lenghtu \n");  */
  /*  chan_in_word(&lenghtu, &internal1);  */
/*printf(" receive from ro1 lenghtu=%d 1 \n",lenghtu);*/
    chan_in_word(&lenghtu, in_ports[1]);    /* comprimento de u */
     /*u=set_sub_matrix(NULL,lenghtu,1,1); */
    MATRIX_HEADER(u,lenghtu,1,1);
    chan_in_message(lenghtu, (u+MDATA), in_ports[1]);  /*  u*/
/*printf("ro2 receive from ro1 lenghtu=%d e o u\n", lenghtu);
print_matrix(u);
getchar(); */
     /* tell alt data ready */
      /*chan_out_word(i, &datar1);*/

    if(i<dimcol){
    tamu=lenghtu-1;
    if (tamu==-1) tamu=1;
    chan_out_word(tamu, out_ports[0]);  /* lenght u*/
   /* printf("send to calc1 lenght u\n");   */
    u++;
    MATRIX_HEADER(u,tamu,1,1);
  /*  print_matrix(u);*/
    chan_out_message(tamu, (u+MDATA), out_ports[0]); /* u*/
/*printf("ro2 send to ca1_2 tamu=%d e o  u\n", tamu);*/
    }
    }
    i=i+1;
   }
}
```

É apresentado o código dos processos de cálculo de baixa prioridade *ca1_1* e *ca1_2* situados nos *C40s ROOT* e *P1* respectivamente:

```
/* ca1 in parallel transputer system */

#include <math.h>
#include <chan.h>
#include <c40tim.h>
#include <stdio.h>
#include <alt.h>
#include <stdlib.h>
#include "matmacro.h"
#include "matcalc.h"
#include "matio.h"
/*#include "matpar.h"*/
#include <timer.h>
#define linhas 100
#define to_ms timer_rate()*1000

float modulo(float n)
{
 if (n < 0) n = -n;
 return(n);
```

```
}
float norm(matrix u, int l)
{
 float n=0.0;
 float soma=0.0;
 float soma_r=0.0;
 int i;

 for (i=1;i<=l;i++){
      n=modulo(ELEMENTO(u, i, 1));
/*      getchar();*/
      /*printf("n=%f i=%d l=%d \n", n, i, l);*/
      if (n < 0) n = -n;
      soma = soma+n*n;
 }
 soma_r=sqrt(soma);
 return(soma_r);
}
int sign(float u)
{
  if (u>0)
    return 1;
  if (u==0)
    return 0;
  if (u<0)
    return -1;

}
main(int argc, char *argv[], char *envp[], CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
int proc_numb = (int) in_ports[1];
int loop,loop2, i, dim, buf;
int sweep=1, next_column_in_sweep=1, num_working=2, first_row=1;
float sig, sigg, den, num;
int cont=1,  num_col_proc, dimlin;
int k, m, contad=0, lenghta,  lenghtu=0, tamu;

matrix az=NULL;
matrix ry=NULL;
matrix u=NULL;
matrix w=NULL;
matrix ww=NULL;
matrix b=NULL;

/*printf("Proc %d Init \n",proc_numb); */

/*inicio=(float) timer_now();  */
  chan_in_word(&num_col_proc,in_ports[0]);
/*printf("ca1 %d receive from ro1 or ro2 num_col_proc%d \n",proc_numb, num_col_proc);*/
  chan_in_word(&dimlin,in_ports[0]);
/*printf("ca1 %d receive from ro1 or ro2 dimlin=%d \n",proc_numb, dimlin); */
  chan_in_word(&lenghta, in_ports[0]);
/*printf("ca1 %d receive from ro1 or ro2 lenghta=%d \n",proc_numb, lenghta);   */

b=set_matrix(NULL, dimlin,num_col_proc);
matrix_zeros(b);
```

```
/* sync */
/*printf("");  */
  chan_in_message(lenghta,(b+MDATA),in_ports[0]);
/*printf("ca1%d receive from ro1 or ro2 matrix b \n",proc_numb); */
/*print_matrix(b);*/
/*getchar();  */

u=set_matrix(NULL,dimlin,1);
matrix_zeros(u);
w=set_matrix(NULL,dimlin,1);
matrix_zeros(w);
az=set_matrix(NULL,dimlin,num_col_proc);
ry=set_matrix(NULL,dimlin,num_col_proc);
matrix_zeros(ry);
ww=set_matrix(NULL,1,num_col_proc);

/*printf("Ca1%d  is not in  While \n",proc_numb);  */

  /* sync */
 /*chan_out_word(buf,in_ports[0]);*/

while (sweep<=num_col_proc)
{
/*printf("ca1%d is in WHILE \n",proc_numb);*/
 if (fmod(next_column_in_sweep, num_working)==fmod((int)(proc_numb-3+1), num_working))
 {

/* printf("Ca1%d next_column_in_sweep=%d num_working=%d is in if \n",proc_numb,
next_column_in_sweep, num_working );
 printf("Ca1%d sweep=%d dimlin=%d num_col_proc=%d \n", proc_numb,
sweep,dimlin,num_col_proc);*/
   /*   print_matrix(b); */
   submatrix(b, u, first_row, sweep, dimlin, sweep);
  /* printf("ca1 %d first_row %d lenghta %d\n",proc_numb,first_row, dimlin);  */
 /*  printf("ca1 %d u \n",proc_numb);*/
 /*  print_matrix(u);*/
   lenghtu=NLINHAS(u);
   sig=norm(u,lenghtu);
   sigg=-sign(ELEMENTO(u,1,1))*sig;
   den=sig*(sig+modulo(ELEMENTO(u,1,1)));
  /* printf("ca1 %d sig=%f sigg=%f den=%f \n", proc_numb,sig, sigg, den);*/
   ELEMENTO(u,1,1)=sign(ELEMENTO(u,1,1))*(sig+modulo(ELEMENTO(u,1,1)));
/*   printf("ca1 %d u \n",proc_numb);    */
/*   print_matrix(u);    */
   num=1/sqrt(den);
   nummult(num,u,u,0);
 /*  mcopy(w,u);  */
  /* printf("Proc %d u \n",proc_numb);*/
  /* print_matrix(u);*/
   ELEMENTO(b,first_row,sweep)=sigg;
  /* printf("Proc %d b \n",proc_numb); */
  /* print_matrix(b);     */
   if (first_row< dimlin)
     {
      for (loop2=1;loop2<=sweep;loop2++)
```

```
      for (loop=loop2+1;loop<=dimlin;loop++)
        ELEMENTO(b,loop,loop2)=0;
      }
 /* printf("ca1%d b \n",proc_numb);*/
 /* print_matrix(b); */
  sweep=sweep+1;
 /* printf("ca1%d  sweep=%d\n", proc_numb, sweep); */
 /* printf("lenghtu %d\n",lenghtu);   */
  chan_out_word(lenghtu,out_ports[0]);
/* printf("ca1%d send to ro1 or ro2 lenghtu=%d \n",proc_numb, lenghtu); */
  chan_out_message(lenghtu,(u+MDATA), out_ports[0]);
 /* printf("ca1%d: send to ro1 or ro2 u \n" ,proc_numb);
  getchar();*/
 }
 else
 {
 /* printf("Ca1%d is in else waiting u \n",proc_numb);*/
   /*tamu=(int)NLINHAS(b); */
   chan_in_word(&tamu, in_ports[0]);
 /* printf("ca1%d receive from ro1 or ro2 tamu=%d  \n",proc_numb, tamu); */
   /*if (tamu==0) break;*/
   /*u=set_sub_matrix(u,tamu,1,1);*/
   MATRIX_HEADER(u,tamu,1,1);
  /* printf("ca1%d  %d\n",proc_numb, tamu); */
   chan_in_message(tamu,(u+MDATA),in_ports[0]);
  /* printf("ca1 %d receive from ro1 or ro2 tamu=%d\n", proc_numb, tamu); */

  /* printf("matrix u is of  ca1%d \n", proc_numb); */
  /* print_matrix(u);
   getchar();*/
 }
 if (sweep <=num_col_proc){
 /* printf("ca1%d is in if\n", proc_numb); */

     submatrix(b, ry, first_row, sweep, dimlin, num_col_proc);
    /* printf("first_row %d sweep %d dimlin %d num_col_proc %d\n", first_row, sweep, dimlin,
num_col_proc);        */
   /* printf("ca1%d ry \n",proc_numb); */
    /*  print_matrix(ry);   */
/*    getchar();              */
     mmult(u,ry,ww,1,0);
   /* printf("ca1%d ww \n",proc_numb); */
   /*   print_matrix(ww);
     getchar();         */
     mmult(u,ww,az,0,0);
   /* printf("ca1 %d az \n",proc_numb);  */
   /*   print_matrix(az); */
     MSUB(ry,az,ry);
   /* printf("ca1%d ry \n",proc_numb); */
    /* print_matrix(ry);   */
     submatrix2(ry, b, 1, 1, NLINHAS(ry), NCOLUNAS(ry), first_row, sweep);
   /* printf("Proc %d b \n",proc_numb);*/
   /*   print_matrix(b);    */
     first_row =first_row+1;
     next_column_in_sweep=next_column_in_sweep+1;
     if (next_column_in_sweep>num_working)
```

```
    next_column_in_sweep=1;
  /* getchar();
   printf("calc%d tem o first_row=%d e next_column_in_sweep=%d\n", proc_numb, first_row,
next_column_in_sweep); */
 }
}/*end while*/
}
```

ANEXO I

Programa de comunicações entre *C40s* em canais físicos

O código do programa de comunicações entre *C40s* através de canais físicos é o apresentado neste anexo.

O ficheiro de configuração da topologia contituída por *C40s* é o seguinte:

```
processor host  type=pc
processor root  type=t800

processor wrk1  type=c40 kernel="slot4.krn"

wire ? root [0] host [0]
wire ? root[2] wrk1[0]

!task
task afserver ins=1 outs=1
task filter ins=2 outs=2 data=10k
!task driver ins=2 outs=2
!task sender ins=2 outs=2
!task reply ins=1 outs=1

task master ins=3 outs=2   !data=20k
task slave ins=1 outs=1    !stack=1.8k opt=stack:ramblk0 opt=code:ramblk1

bind input master[2] value = 11

default connect physical
connect ? afserver[0] filter[0]

connect ? filter[0] afserver[0]
connect ? filter[1] master[1]
connect ? master[1] filter[1]
connect ? master[0] slave[0]
connect ? slave[0] master[0]


place afserver host
place filter root
place master root
place slave wrk1
```

Os programas a seguir permitem determinar tempos de comunicação na transmissão de um vector de reais entre *C40s*. O primeiro programa, o *Master* envia para o segundo o *Slave* cada um dos elementos de um vector. Depois do slave receber envia de novo os elementos para o *Master*:

```
/*Master*/
/* Times for transmit flots between parallel transputer system */

#include <stdlib.h>
#include <timer.h>
```

```
#include <stdio.h>
#include <chan.h>

main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)

 {
  float buffer [100], tempo;
  int dimlin=100, m;
  int inicio, fim, i=1;

  for (i=10;1<=dimlin;i+=10){
   m=i*4;
   inicio = timer_now();
   chan_out_message(m,buffer, out_ports[0]);
   chan_in_message(m,buffer, in_ports[0]);
   fim = timer_now();
   tempo = (float)(fim-inicio);
   printf("%f\n",(float)tempo);
  }
 }


/*Slave*/
/* Times for transmit flots between parallel transputer system */

#include <chan.h
#include <ieee.h
main(int argc, char *argv[], char *envp[], CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
 float buffer[100];
 int i, dimlin=100, m;
  for (i=10;i<=dimlin;i+=10) {
   chan_in_message(i, buffer, in_ports[0]);
   ieee_single_to_float_vec(buffer,i);
   ieee_single_from_float_vec(buffer,i);
   chan_out_message(i, buffer, out_ports[0]);
  }
 }
```

ANEXO J

Programa de comunicações entre *C40s* em canais virtuais

O código do programa de comunicações entre *C40s* através de canais virtuais é o apresentado neste anexo.

O ficheiro de configuração da topologia contituída por *C40s* é o seguinte:

```
! configurer file for determine comunication time between C4/C4
!Hardware configuration
processor root type=TMS320C40
processor root1 type=TMS320C40
wire ? root[0] root1[3]
wire ? root[5] root1[2]

!Software configuration
task master ins=1 outs=1 stack=5.4k heap=1.1k  opt=code opt=code:ramblk0 opt=heap:ramblk1
task slave ins=1 outs=1 stack=5.4k heap=1.1k opt=code opt=code:ramblk0 opt=heap:ramblk1

place master root
place slave root1

default connect virtual
!UPR MAX = 100
!UPR BUFFERS = 4

connect ? master[0] slave[0]
connect ? slave[0] master[0]
```

Os programas a seguir permitem determinar tempos de comunicação na transmissão de um vector de reais entre *C40s*. O primeiro programa, o *Master* envia para o segundo o *Slave* cada um dos elementos de um vector. Depois do slave receber envia de novo os elementos para o *Master*:

```
/*Master*/
/* Times for transmit flots between parallel transputer system */

#include <link.h>
#include <chan.h>
#include <c40tim.h>
#include <stdio.h>
#include <timer.h>
/*#define to_ms timer_rate()*1000 */

#define MAX 2000

main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
  float buffer[MAX];
  float tempo[MAX];
  int dimlin=MAX, i, inicio, fim, k;
/*   printf ("velocidede de relogio: %d\n",tim_rate()/1000000);*/
  tim_claim(0,TIM_INTERNAL | TIM_CLOCK_MODE | TIM_TCLK_LOW);
```

```
/*   printf ("velocidede de relogio %d \n:",tim_rate()/1000000);*/
   tim_set_rate(50000000);
   tim_stopwatch(0);


   for (i=1;i<=dimlin;i+=1){
     inicio = tim_now(0);
     chan_out_message(i, buffer, out_ports[0]);
     chan_in_message(i, buffer, in_ports[0]);
     fim = tim_now(0);
     tempo[i]=(float)tim_seconds(fim-inicio)*1000;
/*     printf("Tempo total: %f %s i=%d \n",tempo[i]," ms", i);*/
     printf("%f\n",tempo[i]);
/*     getchar();*/

   }
}


/*Slave*/
/* Times for transmit flots between parallel transputer system */

#include <chan.h>
#include <link.h>
#define MAX 2000

main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
  float buffer[MAX];
  int i, dimlin=MAX;
  for (i=1;i<=dimlin;i+=1)
    {
    chan_in_message(i, buffer, in_ports[0]);
    chan_out_message(i, buffer, out_ports[0]);
    }
}
```

ANEXO K

Programa Paralelo em *Transputers* e *C40*

O ficheiro de configuração da topologia contituída por *Transputer* e *C40* é o seguinte:

```
! configurer file for parallel T8/C40 system
!hardware configuration

processor host type=pc
processor root type= t800
processor wrk1 type=C40 kernel="slot2.krn"
wire ?  root[0] host[0]
wire ?  root[3] wrk1[0]

!Software configuration

task afserver ins=1 outs=1
task filter ins=2 outs=2 data=10k
task ro1 ins=3 outs=3 priority=0 stack=1.5k heap=100k opt=stack
task ro2 ins=2 outs=2 priority=0 stack=1.5k heap=100k opt=stack:ramblk0
task ca1 ins=1 outs=1 priority=1 data=800k
task ca2 ins=1 outs=1 priority=1 stack=1.5k heap=800k opt=stack:ramblk1

place afserver host
place filter root
place ro1  root
place ro2  wrk1
place ca1 root
place ca2 wrk1

default connect physical

connect ? afserver[0] filter[0]
connect ? filter[0] afserver[0]
connect ? filter[1] ro1[1]
connect ? ro1[1] filter[1]
connect ? ro1[0] ca1[0]
connect ? ca1[0] ro1[0]
connect ? ro1[2] ro2[1]
connect ? ro2[1] ca1[2]
connect ? ro2[0] ca2[0]
connect ? ca2[0] ro2[0]
```

O código a seguir corresponde ao processo *router1* (*ro12*) de alta prioridade situado no *transputer root*:

```
/* ro1 in transputer */

#include <chan.h>
#include <stdio.h>
#include <alt.h>
#include <math.h>
#include <stdlib.h>
#include "matmacro.h"
#include "matcalc.h"
#include "matio.h"
#include <timer.h>
```

```
#include <thread.h>

float to_ms()
  { if (thread_priority()==0) return 1000;
     else return 15.625;}

main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
  int proc_numb=1;
  float inicio,fim, inicioc, fimc, result=0, tempoc=0;
  int lenghta=0, tamu=0, lena=0,  lenu=0, tam=0, tamuu=0, le=0;
  int i=1,  cont=1, rem, n, num_col_proc=0, num_working=2;
  int dimlin, dimcol;

  matrix a=NULL;
  matrix u=NULL;
  matrix b=NULL;
  matrix v=NULL;

  printf("\n LINHAS:\n");
  scanf("%d",&dimlin);
  while (cont)
  {
   printf("\n COLUNAS:\n");
   scanf("%d",&dimcol);
  printf("%d\n", dimcol);
   n=dimcol/num_working;
   rem=dimcol-n*num_working;
   printf("%d\n", rem);
   if (rem!=0)
    printf("\n");
   else
    cont=0;
   }

  u=set_matrix(NULL,dimlin,1);
  v=set_matrix(NULL,dimlin,1);
  matrix_zeros(v);
  num_col_proc=dimcol/num_working;
printf("1:\n");
printf("num_col_proc=%d\n", num_col_proc);
getchar();
  a=set_matrix(NULL,dimlin,dimcol);
  matrix_zeros(a);
printf("1:\n");
print_matrix_full(a);
getchar();
  load_matrix(a,"1.2, 1.3, 1.4, 2.9, 2.1,2.2,3.3,3.9,2.1,2.1,2.3,4.9,2.3,4.1,3.9,4.5, 5.5, 4.5, 6.1,4.1,5.6, 7.8,
,9.9, 4.6,7.1,1.1,1.1,1.1,1.1,1.1,1.1,1.1,1.1,1.1,2.2,2.2,2.2,3.3,3.3,3.3,4.4, 5.5,5.5,5.5,5.5,5.5,5.5,5.5,1.1,1.1 !");
printf("1:\n");
print_matrix_full(a);
getchar();
  b=set_matrix(NULL,dimlin,num_col_proc);
  matrix_zeros(b);
printf("1:\n");
```

```
/*print_matrix_full(b); */
getchar();
  chan_out_word(num_col_proc,out_ports[0]);
printf ("Ro1 : send to cal num_col_proc\n", num_col_proc);
  chan_out_word(dimlin,out_ports[0]);
printf("Ro1 : send to ca1 dimlin\n", dimlin);
  lenghta = dimlin*num_col_proc;
printf("lenghta %d \n", lenghta);
  lena=lenghta*4;
  chan_out_word(lena, out_ports[0]);
printf("Ro1 :send to ca1_lena 3\n", lena);
  submatrix(a,b,1,1,dimlin,dimcol/2);
  chan_out_message(lena,(b+MDATA), out_ports[0]);
printf("Ro1  send to ca1_b 4\n");
print_matrix_full(b);
getchar();

  chan_out_word(dimcol,out_ports[2]);
printf ("Ro1 send to ro2 dimcol\n", dimcol);
  chan_out_word(num_col_proc,out_ports[2]);
printf("Ro1 send to ro2 num_col_proc\n", num_col_proc);
  chan_out_word(dimlin,out_ports[2]);
printf("Ro1 send to ro2 dimlin\n", dimlin);
  chan_out_word(lena, out_ports[2]);
printf("Ro1 send to ro2 lena\n", lena);
  submatrix(a,b,1,dimcol/2+1, dimlin, dimcol);
  chan_out_message(lena,(b+MDATA), out_ports[2]);
printf("Ro1 send to ro2 b\n");
/*print_matrix_full(b); */
getchar();

  chan_in_message(lena,(b+MDATA),in_ports[2]);
printf("Ro1 : receive from ro2 b\n");
print_matrix_full(b);
getchar();
  inicio=(float) timer_now();
 /* inicioc=(float) timer_now); */

/*   printf("%d\n",dimcol);*/
  while(i<=dimcol)
/*   while (i <= (int)(dimcol-num_working+proc_numb)) */
   {
   int c;
/*    printf("Ro1 i=%d: in while 1 \n",i);*/
   c = alt_wait(2, in_ports[0], in_ports[2]);
   if (c == 0)
   {
  chan_in_word(&lenu, in_ports[0]);
   /*printf("Ro1 : receive from ca1lenu b\n");*/
   tam=lenu/4;
   /*u=set_sub_matrix(u,tam,1,1);*/
   MATRIX_HEADER (u,tam,1,1);
   chan_in_message(lenu, (u+MDATA), in_ports[0]); /* u*/
  /*printf("Ro1 : receive from ca1u b\n");*/
  /*fimc=(float) timer_now();
  tempoc=(float)(fimc-inicioc)/to_ms();
```

```
  printf("Tempo Comunia‡äes1: %f %s\n",tempoc,"milisegundos");
  getchar(); */
//   printf("ro1 receive from ca1 u\n");
//   print_matrix(u);
   if (i<dimcol){
    chan_out_word(lenu, out_ports[2]);
//    printf("ro1 send to ro2 lenu=%d\n", lenu);
    chan_out_message(lenu, (u+MDATA), out_ports[2]);
//    printf("ro1 send to ro2 matrix u\n");

 /* fimc=(float) timer_now();
  tempoc=(float)(fimc-inicioc)/to_ms();
  printf("Tempo Comunia‡äes2: %f %s\n",tempoc,"milisegundos");
  getchar();  */
   }
   }

   else
   {
    chan_in_word(&lenu, in_ports[2]);
//    printf("ro1 receive from  ro2 lenu=%d  1\n",lenu);
    tam=lenu;
//    printf("tam=%d\n",tam);
    /*u=set_sub_matrix(u,tam,1,1);*/
    MATRIX_HEADER (u,tam,1,1);
    chan_in_message(tam*4, (u+MDATA), in_ports[2]);  /*  u*/
//    printf("ro1 receive from ro2  matrix u\n");
//    print_matrix(u);

 /* fimc=(float) timer_now();
  tempoc=(float)(fimc-inicioc)/to_ms();
 printf("Tempo Comunia‡äes3: %f %s\n",tempoc,"milisegundos");
  getchar(); */
   if (i<dimcol){

    tamu=tam-1;
    /*printf("tamu=%d\n",tamu);
    getchar();*/
    tamuu=(int)tamu*4;
    /*printf("tamuu=%d\n",tamuu);
    getchar();*/
    chan_out_word(tamuu, out_ports[0]);
 /* fimc=(float) timer_now();
 tempoc=(float)(fimc-inicioc)/to_ms();
 printf("Tempo Comunia‡äes4: %f %s\n",tempoc,"milisegundos");
 getchar(); */

    /*v=set_sub_matrix(v,tamu,1,1);*/
    /*MATRIX_HEADER (v,tamu,1,1);
    submatrix(u,v,2,1,tam,1);*/
    /*u=set_sub_matrix(u,tamu,1,1);*/
    u++;
    MATRIX_HEADER (u,tamu,1,1);
    /*mcopy(v,u);*/
    chan_out_message(tamuu, (u+MDATA), out_ports[0]); /* u*/
//    printf("Ro1 send to calc1 matrix u\n");
```

```
  /* fimc=(float) timer_now();
   tempoc=(float)(fimc-inicioc)/to_ms();
   printf("Tempo Comunia‡äes5: %f %s\n",tempoc,"milisegundos");
   getchar(); */
     }
    }
    i=i+1;
   }
    fim=(float) timer_now();
    result=(float)(fim-inicio)/to_ms();
    printf("Router1: %f %s\n",result,"milisegundos");
   /* printf("Tempo Comunia‡äes: %f %s\n",tempoc,"milisegundos"); */
}
```

O código a seguir corresponde ao processo *router2* (*ro2*) de alta prioridade situado no *C40 wrk1*:

```
/* ro2 in C40 */


#include <chan.h>
#include <c40tim.h>
/*#include <stdio.h> */
#include <alt.h>
#include <math.h>
#include <stdlib.h>
#include "matcalc.h"
/*#include "matio.h" */
#include <timer.h>
#include <ieee.h>
#define to_ms timer_rate()*1000

struct map{
      CHAN *dataready;
      CHAN *phys_chan;
      CHAN *internal_chan;
};

CHAN   datar1,datar0,internal0,internal1;
void guard(void *arg)
{
  struct map *s =(struct map *)arg;
  int buf;
  for (;;){
     chan_in_word(&buf,s->dataready);
     chan_in_word(&buf, s->phys_chan);
     chan_out_word(buf, s->internal_chan);
  }
}
main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
int proc_numb=2;
```

```
  int lena, lenu=0,   dim, tamu, buf;
  int i=1, x, cont=1, rem, n,num_col_proc, num_working=2, m, c;
  int dimlin, dimcol;

  matrix a=NULL;
  matrix u=NULL;
  matrix b=NULL;
  matrix v=NULL;
  struct map s0, s1;

  s0.phys_chan=in_ports[0];
  s0.internal_chan=&internal0;
  s0.dataready=&datar0;
  chan_init(&internal0);
  chan_init(&datar0);

  s1.phys_chan=in_ports[1];
  s1.internal_chan=&internal1;
  s1.dataready=&datar1;
  chan_init(&internal1);
  chan_init(&datar1);

  thread_new(guard, 1024, &s0);
  thread_new(guard, 1024, &s1);

  u=set_matrix(NULL,dimlin,1);
  v=set_matrix(NULL,dimlin,1);
  matrix_zeros(v);

/*ro2 receive from ro1*/
  chan_in_word(&dimcol,in_ports[1]);
  chan_in_word(&num_col_proc,in_ports[1]);
  chan_in_word(&dimlin,in_ports[1]);
  chan_in_word(&lena, in_ports[1]);
  lena=lena/4;
  b=set_matrix(NULL,dimlin,num_col_proc);
  chan_in_message(lena,(b+MDATA), in_ports[1]);
  ieee_single_to_float_vec((b+MDATA),lena);
  ieee_single_from_float_vec((b+MDATA),lena);
  chan_out_message( lena,(b+MDATA), out_ports[1]);

/*ro2 receive from ro1*/
  chan_out_word(num_col_proc,out_ports[0]);
  chan_out_word(dimlin,out_ports[0]);
  chan_out_word(lena, out_ports[0]);
  chan_out_message(lena,(b+MDATA), out_ports[0]);

  /* sync */
 /* chan_in_word(&buf, out_ports[1]);*/
  chan_out_word(1000, &datar0);
  chan_out_word(1000, &datar1);

/*  while(i<=dimcol)*/
 while (i <= (int)(dimcol-num_working+proc_numb))
 /*while (i <= dimcol)*/
   {
```

```
    int c;
    c = alt_wait(2, &internal0, &internal1);

    /*chan_out_word(1, out_ports[1]);
    break;*/
   /* c = alt_wait(2, in_ports[0], in_ports[1]);*/
    if (c == 0)
    {
    chan_in_word(&lenu, &internal0);  /*length  u*/
    /*u=set_sub_matrix(u,lenu,1,1);*/
    MATRIX_HEADER(u,lenu,1,1);
    chan_in_message(lenu, (u+MDATA), in_ports[0]); /* u*/
    /* tell alt data ready */
    chan_out_word(i, &datar0);
   if(i<dimcol){
    chan_out_word(lenu, out_ports[1]);  /* lenght u*/
    ieee_single_from_float_vec((u+MDATA),lenu);
    chan_out_message(lenu, (u+MDATA), out_ports[1]); /* u*/
    }
    }
    else
    {

    chan_in_word(&lenu, &internal1);
    lenu=lenu/4;
  /*u=set_sub_matrix(u,lenu,1,1); */
    MATRIX_HEADER(u,lenu,1,1);
    chan_in_message(lenu, (u+MDATA), in_ports[1]);  /*  u*/
    ieee_single_to_float_vec((u+MDATA), lenu);
     /* tell alt data ready */
      chan_out_word(i, &datar1);
   if(i<dimcol){
    tamu=lenu-1;
    if (tamu==-1) tamu=1;
    chan_out_word(tamu, out_ports[0]);  /* comprimento u*/
   /*v=set_sub_matrix(v,tamu,1,1);
    submatrix(u,v,2,1,lenu,1);*/
    /*u=set_sub_matrix(u,tamu,1,1);
    mcopy(v,u);*/
    u++;
    MATRIX_HEADER(u,tamu,1,1);
    chan_out_message(tamu, (u+MDATA), out_ports[0]); /* u*/
    }
    }
    i=i+1;
    }
  }
```

É apresentado o código do processos de cálculo de baixa prioridade *cal* situado no *Transputer root* :

```
/* cal in  transputer */

#include <math.h>
#include <chan.h>
#include <alt.h>
#include <stdlib.h>
#include "matmacro.h"
#include "matcalc.h"
#include <timer.h>
#include <thread.h>

float to_ms()
  { if (thread_priority()==0) return 1000;
      else return 15.625;}
float modulo(float n)
{
 if (n < 0) n = -n;
 return(n);
}
float norm(matrix u, int l)
{
  float n=0.0;
  float soma=0.0;
  float soma_r=0.0;
  int i;
  for (i=1;i<=l;i++){
        n=modulo(ELEMENTO(u, i, 1));
        if (n < 0) n = -n;
        soma = soma+n*n;
  }
  soma_r=sqrt(soma);
  return(soma_r);
}
int sign(float u)
{
   if (u>0)
     return 1;
   if (u==0)
     return 0;
   if (u<0)
     return -1;
}
main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
int proc_numb=3;
int loop,loop2;
int sweep=1, next_column_in_sweep=1, num_working=2, first_row=1;
float sig, sigg, den, num;
int cont=1, num_col_proc, dimlin;
int   contad=0, lena, lenghtu, tamuu, lenu=0, tamuuu;
```

```
matrix az=NULL;
matrix ry=NULL;
matrix u=NULL;
matrix w=NULL;
matrix ww=NULL;
matrix b=NULL;

/* receive from ro1*/
 chan_in_word(&num_col_proc,in_ports[0]);
 chan_in_word(&dimlin,in_ports[0]);
 chan_in_word(&lena, in_ports[0]);
 b=set_matrix(NULL, dimlin,num_col_proc);
 matrix_zeros(b);
 chan_in_message(lena,(b+MDATA),in_ports[0]);
u=set_matrix(NULL,dimlin,1);
matrix_zeros(u);
w=set_matrix(NULL,dimlin,1);
matrix_zeros(w);
az=set_matrix(NULL,dimlin,num_col_proc);
ry=set_matrix(NULL,dimlin,num_col_proc);
matrix_zeros(ry);
ww=set_matrix(NULL,1,num_col_proc);
while (sweep<=num_col_proc)
/*while(1)*/
{
 if (fmod(next_column_in_sweep, num_working)==fmod((int)(4-proc_numb), num_working))
 {
  submatrix(b, u, first_row, sweep, dimlin, sweep);
  lenghtu=(int)NLINHAS(u);
  sig=norm(u,lenghtu);
  sigg=-sign(ELEMENTO(u,1,1))*sig;
  den=sig*(sig+modulo(ELEMENTO(u,1,1)));
  ELEMENTO(u,1,1)=sign(ELEMENTO(u,1,1))*(sig+modulo(ELEMENTO(u,1,1)));
  num=1/sqrt(den);
  nummult(num,u,u,0);
  /*mcopy(w,u);*/
  ELEMENTO(b,first_row,sweep)=sigg;
  if (first_row< dimlin)
    {
     for (loop2=1;loop2<=sweep;loop2++)
       for (loop=loop2+1;loop<=dimlin;loop++)
         ELEMENTO(b,loop,loop2)=0;
    }
  sweep=sweep+1;
  lenu=(int)lenghtu*4;
  chan_out_word(lenu,out_ports[0]);
  chan_out_message(lenu,(u+MDATA), out_ports[0]);
}
else
{
 chan_in_word(&tamuu, in_ports[0]);
 tamuuu=(int)tamuu/4;
 /*u=set_sub_matrix(u,tamuuu,1,1);*/
 MATRIX_HEADER(u,tamuuu,1,1);
 chan_in_message(tamuu,(u+MDATA),in_ports[0]);
}
```

```
 if (sweep <=num_col_proc){
    submatrix(b, ry, first_row, sweep, dimlin, num_col_proc);
    mmult(u,ry,ww,1,0);
    mmult(u,ww,az,0,0);
    MSUB(ry,az,ry);
    submatrix2(ry, b, 1, 1,(int)(NLINHAS(ry)), (int)(NCOLUNAS(ry)), first_row, sweep);
    first_row =first_row+1;
    next_column_in_sweep=next_column_in_sweep+1;
    if (next_column_in_sweep>num_working)
     next_column_in_sweep=1;
 }
}/*end do while*/
}
```

É apresentado o código do processos de cálculo de baixa prioridade *ca1* situado no *Transputer root* :

```
/* ca2 in C40*/

#include <math.h>
#include <chan.h>
#include <c40tim.h>
#include <alt.h>
#include <stdlib.h>
#include "matmacro.h"
#include "matcalc.h"
#include <timer.h>
#define linhas 100
#define to_ms timer_rate()*1000

float modulo(float n)
{
 if (n < 0) n = -n;
 return(n);
}
float norm(matrix u, int l)
{
 float n=0.0;
 float soma=0.0;
 float soma_r=0.0;
 int i;
 for (i=1;i<=l;i++){
      n=modulo(ELEMENTO(u, i, 1));
      if (n < 0) n = -n;
      soma = soma+n*n;
 }
 soma_r=sqrt(soma);
 return(soma_r);
}

int sign(float u)
{
  if (u>0)
    return 1;
```

```
  if (u==0)
    return 0;
  if (u<0)
    return -1;
}
main(int argc, char *argv[], char *envp[], CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
int proc_numb=4;
int loop,loop2, i, dim, buf;
int sweep=1, next_column_in_sweep=1, num_working=2, first_row=1;
float sig, sigg, den, num;
int cont=1,  num_col_proc, dimlin;
int k, m, contad=0, lena,  lenu=0, tamu;
matrix az=NULL;
matrix ry=NULL;
matrix u=NULL;
matrix w=NULL;
matrix ww=NULL;
matrix b=NULL;

/*ca2 receive from ro2*/
 chan_in_word(&num_col_proc,in_ports[0]);
 chan_in_word(&dimlin,in_ports[0]);
 b=set_matrix(NULL, dimlin,num_col_proc);
 matrix_zeros(b);
 chan_in_word(&lena, in_ports[0]);

/* sync */
  chan_in_message(lena,(b+MDATA),in_ports[0]);
u=set_matrix(NULL,dimlin,1);
matrix_zeros(u);
w=set_matrix(NULL,dimlin,1);
matrix_zeros(w);
az=set_matrix(NULL,dimlin,num_col_proc);
ry=set_matrix(NULL,dimlin,num_col_proc);
matrix_zeros(ry);
ww=set_matrix(NULL,1,num_col_proc);



  /* sync */
 /*chan_out_word(buf,in_ports[0]);*/
while (sweep<=num_col_proc)
/*while(1)*/
{
if (fmod(next_column_in_sweep, num_working)==fmod((int)(4-proc_numb), num_working))
{
 submatrix(b, u, first_row, sweep, dimlin, sweep);
 lenu=NLINHAS(u);
 sig=norm(u,lenu);
 sigg=-sign(ELEMENTO(u,1,1))*sig;
 den=sig*(sig+modulo(ELEMENTO(u,1,1)));
 ELEMENTO(u,1,1)=sign(ELEMENTO(u,1,1))*(sig+modulo(ELEMENTO(u,1,1)));

 num=1/sqrt(den);
```

ANEXO L

Programa de comunicações entre *Transputers* e *C40*

```
  nummult(num,u,u,0);
  /*mcopy(w,u);*/
  ELEMENTO(b,first_row,sweep)=sigg;
  if (first_row< dimlin)
    {
     for (loop2=1;loop2<=sweep;loop2++)
       for (loop=loop2+1;loop<=dimlin;loop++)
         ELEMENTO(b,loop,loop2)=0;
     }
  sweep=sweep+1;
  chan_out_word(lenu,out_ports[0]);

  chan_out_message(lenu,(u+MDATA), out_ports[0]);
 }
 else
 {
  tamu=(int)NLINHAS(b);
  chan_in_word(&tamu, in_ports[0]);
  /*u=set_sub_matrix(u,tamu,1,1);*/
  MATRIX_HEADER(u,tamu,1,1);
  chan_in_message(tamu,(u+MDATA),in_ports[0]);
 }
 if (sweep <=num_col_proc){
    submatrix(b, ry, first_row, sweep, dimlin, num_col_proc);
    mmult(u,ry,ww,1,0);
    mmult(u,ww,az,0,0);
    MSUB(ry,az,ry);
    submatrix2(ry, b, 1, 1, NLINHAS(ry), NCOLUNAS(ry), first_row, sweep);
    first_row =first_row+1;
    next_column_in_sweep=next_column_in_sweep+1;
    if (next_column_in_sweep>num_working)
      next_column_in_sweep=1;
 }
}/*end do while*/
}
```

O código do programa de comunicações entre o *Transputer* e o *C40* é o apresentado neste anexo.

O ficheiro de configuração da topologia contituída pelo Transputer e pelo *C40* é o seguinte:

```
! configurer file
processor host  type=pc
processor root  type=t800
processor wrk1  type=c40 kernel="slot4.krn"
wire ? root [0] host [0]
wire ? root[2] wrk1[0]

!task
task afserver ins=1 outs=1
task filter ins=2 outs=2 data=10k
task master ins=3 outs=2   !data=20k
task slave ins=1 outs=1    !stack=1.8k opt=stack:ramblk0 opt=code:ramblk1

bind input master[2] value = 11

default connect physical
connect ? afserver[0] filter[0]
connect ? filter[0] afserver[0]
connect ? filter[1] master[1]
connect ? master[1] filter[1]
connect ? master[0] slave[0]
connect ? slave[0] master[0]

place afserver host
place filter root
place master root
place slave wrk1
```

Os programas a seguir permitem determinar tempos de comunicação na transmissão de um vector de reais entre *C40s*. O primeiro programa, o *Master* envia para o segundo o *Slave* cada um dos elementos de um vector. Depois do slave receber envia de novo os elementos para o *Master*:

```
/*Master*/
/* Times for transmit flots between parallel transputer/C40 system */

#include <stdlib.h>
#include <timer.h>
#include <stdio.h>
#include <chan.h>

main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
 {
  float buffer [100], tempo;
  int dimlin=100, m;
```

```
    int inicio, fim, i=1;
    for (i=10;1<=dimlin;i+=10){
     m=i*4;
     inicio = timer_now();
     chan_out_message(m,buffer, out_ports[0]);
     chan_in_message(m,buffer, in_ports[0]);
     fim = timer_now();
     tempo = (float)(fim-inicio);
     printf("%f\n",(float)tempo);
    }
  }


/*Slaver*/
/* Times for transmit flots between parallel transputer/C40 system */

#include <chan.h>
#include <ieee.h>

main(int argc, char *argv[], char *envp[],
    CHAN *in_ports[], int ins, CHAN *out_ports[], int outs)
{
  float buffer[100];
  int i, dimlin=100, m;

  for (i=10;i<=dimlin;i+=10)
    {
    chan_in_message(i, buffer, in_ports[0]);
    ieee_single_to_float_vec(buffer,i);
    ieee_single_from_float_vec(buffer,i);
    chan_out_message(i, buffer, out_ports[0]);
    }
}
```