POUR VERSION ZBM.MAC.3 ZBM ET ALLIS UNITS OF SAME LEVEL – USER'S GUIDE

31 AOUT 90.

(BANK ACCES FACILITY ET ADA LIKE LANGUAGE IMPLEMENTATION SPECIFICATION)

modifs/.2 col.*

(couches par hierarchie decroissante)

NIVEAU N

BANK Blocs (<>)

.....

<> correspond a un bank pris comme origine des chainages.

from CX(LCX2)

< .....

..... exit from LCX2;

>

. Au niveau 0, cree l' origine de base CX a partir de laquelle seront chainees toutes les references a des banks. exit from INTERDIT a ce niveau.

. Aux niveaux suivants, calcule en plus l' adresse de la nouvelle origine.
(skip thru et read thru ont le meme effet.)

skip thru TV(LTV)

< .....

..... to next LTV;

..... exit from LTV;

>

. Calcule et definit TV(LTV) comme nouvelle origine de bank.

. to next ou > provoque le bouclage sur LTV = next(LTV) jusqu a ce que LTV=0.

read thru CX(LCX) (J=1,NCX)

< .....

..... to next LCX;

.... exit from LCX;
>

. Calcule et definit CX(LCX) comme nouvelle origine de bank.
. Boucle sur un tableau de link sur des CX (cf TV Bank).
. to next passe au CX suivant, sans verification de valeur de LCX.
. Exige de declarer ~CX(.)~ dans le bloc LOCAL_VALID_CHAINS(voir cette commande).

***** DIAGNOSTICS ET ERREURS :
Ces trois instructions interdisent l' alteration des variables d' adresse declarees dans le bloc associe <>. Une tentative declenche le message :
'** ZBM ERROR ** Z–AD SHOULD NOT BE ALTERED'

AUTRES INSTRUCTIONS (hors blocs<>)
.....

get IC(LIC)
. Calcule LIC a partir de l' origine courante.
. Tolere tout alteration de LIC.
***** PAS DE DIAGNOSTIC .

ALLIS instructions associees a ce niveau de ZBM.

```
<
   <
     <
         execute STEADY;
     >
   >
>;
```

(out of the zero–level FROM bloc)
procedure STEADY

```
<
   . . . . .
   . . . . .
>
```

.execute sera remplace par un GOTO Fortran pointant sur l' instruction de declaration de la procedure STEADY. Ainsi, la procedure se trrouve situee en dehors de tout Bank bloc<>, et donc, toutes les variables d' adresse devront etre calculees avant l' execute.
from,read thru, skip thru et get sont interdites. Les chainages sont deconseilles, et ne pourront etre faits qu aun niveau ZBM de N–1, par @ , @@ et next.
***** DIAGNOSTICS ET ERREURS :
Toute tentative d' utiliser les instructions ZBM de chainage provoque le message d' erreur:
'** ALLIS ERROR ** ZBM INSTRUCTION FORBIDDEN IN NON AT–PROCEDURE'

<

```
  <
   <
     at LCX2 execute STEADY;
   >
  >
>;
(  out of the zero–level FROM bloc  )
procedure STEADY at LCX_LOC                                          *
< from CX(LCX_LOC)                                                   *
 <  .....
    .....
 >
>
```

. Procedure et appel de procedure marquee, cad ayant une origine.

***** DIAGNOSTICS ET ERREURS :

Comme pour procedure et execute, la demande d' execution d' une procedure non definie provoque un message d' erreur :

'** ALLIS ERROR ** PROCEDURE XXXXX IS UNDEFINED'

La definition d une procedure sans demande d' execution associee genere le message:

'** ALLIS WARNING ** PROCEDURE XXXXX IS UNUSED'

(sous forme de commentaire)

Une at–procedure, comme une procedure, doit etre declaree entre le >; zero–level from et la carte END; . Elle exige de declarer un from <> bloc qui sera 0–level.L' adresse de l' origine est pure,ent symbolique et sera remplacee par * une variable locale (par exemple ici L10014, cf l'instruction ALLIS: LOCAL_VARIABLE), * sans qu' il soit necessaire de la declarer LOCALe. *

L absence de from<> declenche un message d' erreur :

'** ALLIS ERROR ** AT–PROC SHOULD CONTAIN A 0–LEVEL FROM< '

——————————— NIVEAU N –1 ———————————————————

next(LTV)

. Identique a @LTV@NEXT@ ou @LTV@NX@

. remplacee par l' adresse du bank NEXT de LTV–bank.

```
local_valid_chains:[~CL~SV~CX~TV~TR~], ou ...~CX(.)~..
                   [~CX~IC~],
                   [~etc~FM*~IN~etc~],                               *
end_chains;
```

. Declaration des chaines de graphes orientes de gauche a droite.

. Permet de referencer tout bank (ex TV) ayant une origine a sa gauche dans une chaine de la liste (ex CL,SV ou CX).

. La premiere chaine de la liste permettant le chainage sera retenue.

. La presence de * signifie adressage indirect,et genere,au lieu de OFMIN, * + IQ(adresse de FM+9)+OFMIN (Le 9 etant dans une macro CO := 9). *

. Une seule declaration de liste est autorisee par subroutine. *

. La presence de CX(.) est requise pour la commande READ THRU. Dans ce cas. *

. il est inutile de repeter la chaine sans CX(.).

***** DIAGNOSTICS ET ERREURS :

Toute tentative de declarer plus d' une liste sera sanctionnee par le

message d’ erreur:
’** ZBM ERROR ** ONLY ONE CHAIN LIST BLOC ALLOWED PER SUBROUTINE’

@@
. exemple : @LCL@CL@@CX@
. remplace par @LCL@CL@SV@CX@
donc, @@ interpose dans la chaine la premiere chaine de la liste declaree dans local_valid_chains permettant d’ expliciter la chaine complete.
exemples valides : @LCL@CL@SV@@TR@
@LCL@CL@@TV@TR@
***** DIAGNOSTICS ET ERREURS :
Si la liste ne permet pas d’ expliciter la chaine, @@ declenche le message d’ erreur:
’** ZBM ERROR ** NO PATH FROM xxx TO yyy IN CHAINS’

NIVEAU 0

@LCL@CL@SV@CX@TV
. Generation de l’adresse permettant de referencer le bloc TV a partir de l origine CL d’ adresse LCL.
. Ne necessite aucune declaration de local_valid_chains.
(remplacee par LQ(LQ(LQ(LCL–OCLSV)–OSVCX)–OCXTV)).
***** AUCUN DIAGNOSTIC .

_ *
A L L I S other instructions . *

_ *

*
LOCAL_VARIABLES: (R_DROIT,ATTENDS–TOI,I_ES–TU?) ; *
*
. Remplace dans le texte subsequent ces trois variables par un nom unique reprenant *
la meme initiale (pour type implicite FORTRAN), comme par exemple: *
R_DROIT —> R10016 *
ATTENDS–TOI —> A10017 *
I_ES–TU? —> I10018 *
. (La declaration se fait n’ importe ou, et la substitution reste valide *
jusqu a la carte END;) *
*

Ensemble d instructions pour emuler le ADA CASE instruction: .
.
.
TYPE mach_in OF (truc_1,truc_2,truc_3);
. Instruction de declaration, declaree avant toute instruction FORTRAN executable.
e
... t
mach_in := truc_3;
. Instanciation de la variable mach_in (il va sans dire que le signe = va aussi). s
.
... u

i

CASE mach_in IS v
< WHEN truc_1 <... > a
WHEN truc_3 < execute ... ; > n
WHEN_OTHERS < print*' error' > t
> e

. Permet de manipuler la variable symbolique mach_in en lui donnant des s
valeurs symboliques truc_1 etc .
. Analogue a l' instruction IF..ELSEIF..ELSEIF..ENDIF; .
. WHEN_OTHERS termine obligatoirement le CASE ... IS bloc. .
. Pour les inconditionnels du FORTRAN, elles sont remplacees par I10011 et .
incrementees dans l' ordre de la liste:
et donc truc_3=truc_2 +1=truc_1 +2 (ne pas en abuser).
. De meme, : truc_1=mach_in +1 ...

Boucles DO a multiples indices:
_____

DO ((...(I=1,12,3),J=J1,J2)...)<...>;

. Un seul <> bloc pour plusieurs boucles imbriquees. Equivalent a:
DO ...
<
...
< DO J=J1,J2
< DO I=1,12,3
...
>
>
...
>;
c.a.d comme la regle d' indices de PRINT(((A(i,j),i=..)j=..) *

_____ 31 Aout 90 _____*

========== ADDITIONS DE DECEMBRE 90
==========

Nouveau GET pour les pointeurs en milieu de bank:
_____

Instruction GETI. Le precompilateur reconnait les indexages indirect, cad les
*links, avec un offset correspondant. Example: GETI AG(IPAG) genere un get AG(L1000x),
et IPAG=L1000x+un offset qui est conventionnellement .In_AG (ou In_[nom de classe].
cf utilisation dans l' exemple ci dessous.

Utilisation de symboles pour les Datas de Banks:
_____

Convention generale: Un Bank–name sera assimile a une classe de Bank.
Tout objet cree sera designe par sa L–adresse:
Ainsi, CL(LCL) exprime que l' objet LCL est de classe CL
Sub–banks : Les if SO ou TO (structure ou Transfer) sont donc
des sous–classes . (de CL ou TR par exemple).

L' utilisation des symboles intervient a deux moments de la vie d' un Bank–objet:
avant sa creation, ou ils sont associes a sa classe. Apres creation, ou ils sont

```
 associes a l' objet. Leur utilsisation est directement commentee dans l' exemple suivant.
+DECK,CTHS.
 SUBROUTINE CTHS(LCL);
"====================="
&F
+CDE,Q,ZPERMQ.
+CDE,Z.
+CDE,OTR,OTPLUG,OCPLUG,OCX,OCL,OIC.
    DIMENSION NAME(2)
&M
+SEQ,OSUCL,OSUTV,OSUIC,OSUCX,OSUET.          ..... Thesaurus des symboles des classes
LOCAL_VALID_CHAINS:[~CL~SV~CX~TV~TR~],          utilisees' ici :CL,TV,IC,CX,ET.
              [~SV~CX~IC~],
              [~CL~GA~],
              [~CL~AB~],
END_CHAINS;
 "  "
;
FROM CL(LCL)
<  GETI GA(IPAG); GET AB(IPAB);                .... utilisation de GETI
    "   GAMA IS 0. FOR dE PROCESSING "
  Q(IPAG+1)=0. ;
    " A–1B MATRIX SET TO ZERO"
  SKIP THRU CX(LCX)
   <  GET TV(LTV); IF ((LTV).Nb_Tr_Var.EQ.0) TO NEXT LCX; ..Il faut donc programmer en
utilisant
    GET IC(LIC); IROW=(LIC).LTr_in_Cl+(LCX).L_IF; ICOL=1;    le ZOOM Bank Description
manual.
    J=IROW+ICOL; Q(IPAB+J)=0. ;
  >
;
"   PRINT *,' CTHS–>, LZ',LZ"
" "
RETURN;
"–ENTRY"
ENTRY ITHS(LCL);
" "                                  .... On va creer un Objet CL(LCL). Il faut pre–seter
"–LIFT CL BANK;NO DESCRIPTION WORDS"        des grandeurs apparaissant dans Ndatas du
Z–B–D.m.
Set .SO.N_Desc for CL = 0;                  .... N_Desc en fait partie, pour les deux SUB–Banks
Set .TO.N_Desc for CL = 0                      possibles .SO. ou .TO.
" "                                  .... Alors, l objet peut etre lifte, car les longueurs
Lift LCL of class CL under (LZUPS,–JZLNKS);      fixes sont connues du PRECOMPILATEUR.
Cette
" "                                    instruction genere les IF_SO ...pour les Sub–class.
"  LCL bank filling "
Point on CL(LCL);                          ..... Pour permettre la version courte, il faut
CALL OHL2I('CTHS..',NAME);                     pointer sur un OBJET.(un seul a la fois)
CALL OCLN(NAME,(LCL).Types(1));
..Types(2)  = 040104;                      ..... Version courte. Seuls les dimensions declarees
"–          (DG/DETA) MATRIX IS SINGULAR"    dans le Z–B–D manual sont authorisees.Par
CALL SBIT1(IQ(LCL),3);                         contre, comme en FORTRAN, pas de check d indices.
```

```
..Nb_St_Var       = 1;
..Nm_Cl           = JZNAME;
..Cd_Cl           = JZCODE;
If_SO                                  ..... Equivalent a IF JZESO.EQ.3  <>
  <  ..SO.Nxt_lvl_Ad    = 3;
     ..SO.N_desc       = (CL).FS.SO.N_Desc   >    ..... La partie droite utilise un symbole de classe,
If_TO                                              cf plus bas.
  <  ..TO.N_desc       = (CL).FS.TO.N_Desc   >
" "
" test" PRINT *,..NDatas,..Nlinks,..NStructs,..ZForm,..Ident; .... Utilisation des ZEBRA infos(cf plus
bas)
RETURN;
"–INITIALIZATION OF STRUCTURE PARAMETERS"
ENTRY RTHS(LCL);
RETURN;
" "
>
END;
```

DES MATRICES FOR USERS
======================

GRASP Ct: Dans l exemple, tout est clair (Ct_SV est le symbole PRECOMP.

```
Point on TD(LTD);
"– C+ FILLING"
PROCEDURE FILL_CT  AT LCXLOC
<  FROM CX(LCXLOC)
   < GRASP CT(LC,LSV,LCX) AND FIX NROWS FOR .Ct_SV=(LSV).Nb_St_Var;
   (LC).Ct_SV(ITV,1)=
     CP*( RMRATE +
        .5*..Data(ITEMP+ITV)*RK1/RMRATE*(RNUM1/DEN–DEN1/DEN2*RNUM)
                                                )*DTIME/2.;

 (LC).Ct_SV(ITFOL,1)=–(LC).Ct_SV(ITV,1);" on suppose que ITV et ITFOL pointent
                                  sur deux eta du CPLUG associe."
   >
>
" "
```

Pour la matrice AB (provisoirement):

```
     GET IC(LIC); IROW=(LIC).LTR_in_Cl +(LCX).LFI_in_IF +1;
     DIMENSION AB(IPAB).AB_Mx(*,NN);  "NN est suppose bien calcule.
     CALL UCOPY(Q(IPABLR+1),(IPAB).AB_Mx(1,IROW),NN);
                          AB_Mx, qui pourrait devenir AB_IC_CX par GRASP it
                          est le nom PRECOMP actuel.
```

De quelques details:

Pour le lecteur interesse, les macros pour l objet cree sont du type:

```
+KEEP,OSUCL.
&'(#.CL).Types'='IQ(#1+1)'
&'(#.CL).Types(#)'='IQ(#1+#2)'                 .... Variable dimensionnee.
```

&’(#.CL).IP_ETA’=’IQ(#1+4)’
&’(#.CL).Nb_St_Var’=’IQ(#1+5)’

&’(CL).FS.NLinks’=’9’ .... Longueur fixe pour la classe.
&’(CL).FS.NStructs’=’5’ (.FS. veut dire : ”For SET FOR” instruction).
&’(CL).FS.IO_Check’=’0’
&’(CL).FS.SO.NDatas’=’(CL).FS.SO.N_desc+15’ .... Longueur a fixer pour l objet. Ici, pour une
&’(CL).FS.TO.NDatas’=’(CL).FS.TO.N_desc+9’ sub–classe donnee.
&’OND=(CL).FS.NDatas;’=’If_SO<OND=(CL).FS.SO.NDatas>;.... Cette macro remplace le OND= genere par LIFT si
If_TO<OND=(CL).FS.TO.NDatas>;’ le Z–B–D man. decrit deux (au plus) sub–classes.

&’Lift # of class # under(#,#)’= genere les instructions:
’ONL=(#2).FS.NLinks;.... qui montre comment les 5 variables reservees heritent
ONS=(#2).FS.NStructs; de l’ information du PRECOMPILATEUR via les OSU... seq.
OND=(#2).FS.NDatas; A savoir, Nbres de Liens, Structures,Datas, IOcheck(=0)
ONIO=(#2).FS.IO_Check;
OCHIDH=”#2 ”; et le nom du Format de MZForm associe au nom de Classe.
et :Lift$(#1,#3,#4);’
&’Lift$(#,#,#);’= qui elle meme genere :
’CALL OQZERO(#1,#2,#3,OCHIDH,ONL,ONS,OND);’.... semblable a MQZERO, sans tableau.
Ca veut dire que le lecteur bricoleur peut lui–meme instancier les 5 grandeurs ZEBRA–reservees et faire Lift$.

Recuperation d’ information ZEBRA par symboles.
==========
Ils sont les suivants:
.NDatas
.NStructs’
.NLinks
.ZForm
.Ident
.Status_Wd (Les macros sont dans ZBM.mac)
(Pour completer, il manque les SET status bits: une discution s’ impose).
..... RAMSES 26/12/90 .