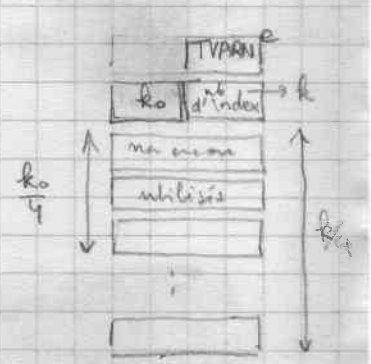
(4)

local {elocal}

elocal
- datav expr {, expr}
- datac exprch {, exprch}
- datai expr {, expr}
- variable } nomi {, nomi}
- chaine
- index #k
- litteral
- access

} dataa nomi {, nomi}




```
YLOCAL: CLR.L  -(SP)
        MOVE   TVARN,-(SP)
MM20:   BSR    DECAN        décode mot clef de elocal
        BNE    MM21
fin                         ↓ pas de mot
        MOVE   (SP)+,D0     ← lea tvarn, A0
        CMP    TVARN(A0),D0
        BLT    errloc1      trop de data
        TST    (SP)+
        BNE    errloc1
        ADD    ?,SP         ← move d0,-(A0); tvarnd
        RTS
MM21:   MEVEN  A3
        TST    D0
        BMI    errloc       → not clef inconnu
        CMP.L  #gdatav,A3   ←
        BNE    MM23
```

datav expr {, expr}

```
MM22:   BSR    WEXPR
        BSR    DECCRV
        BEQ    MM22
        BRA    MM20
```

```
MOVE.B (A3)+,d2
MEVEN  A3
CMP.B  #$72,d2
BNE    MM27
CMP.L  #gdataa,A3
BEQ    GED10
CMP.L  #gaccess,A3
BEQ    GED30
```

```
MM23 : CMP.L   #gdatac, A3
       BNE     MM26
MM25 : BSR     WCHAS
       BSR     DECCRV
       BEQ     MM25
       BRA     MM30
MM26 : CMP.L   #gdatai, A3
       BNE     erris
MM260: BSR     WADR
       MOVEM   (SP)+, D0/D1/D2
       MOVE.L  D3, -(SP)
       ADDQ    #4, D2
       ADDQ    #4, D1
       MOVEM   D0/D1/D2, -(SP)
       BSR     DECCRV
       BEQ     MM260
       BRA     MM20

MM27 : MOVE    (A3)+, D2
       MOVE    (A3), D0
       CMP     #YCVAR-Y, D2
       BNE     MM46

MM28 : MOVE    D0, -(SP)
MM29 : BSR     DECAN
       BNE     MM31
fin
MM30 : ADDQ    #2, SP
       BRA     MM20
MM31 : MOVE    (SP), D0
       BSR     DECCRPG
```

datac expr1 [, expr2 ...]

datai adr [, adr]

sauve la pile et met l'adr

type { CMP #YLITT-Y, D2 / BEQ MM28

→ index

variable / chaine / litteral nom1 [, nom2 ...]

traite nom local simple { teste si fait déjà local / inscription dans pile_proc / modif des listes

```
MM31: MOVEM  (SP),D3/D4        (type / numéro)
      CMP    #$10,D3
      BNE    MM32
      cas littéral
      MOVE   TVARLS,D4          numéro d'index

MM32: BSR    WNWLOC            cache le nom (si nécessaire)
                               inscription dans pile_proc en TMPROC
                               A3 : pointe numéro   A2 : pointe le type
                               [ligne rajoutée de WNWLOC]
      ~~MOVEQ  #1,D2~~
      ~~BSR    DECCRPG~~        (
      BNE    MM34

                               cas var/litt/chaine dimensionné
      ~~LEA    TMPROC,A1~~
                               décale ;-) insère la liste, met D2 = nb d'éléments
      BSR    MM47              conservé

MM34: MOVEM  (SP),D0/D4        (type / numéro)
      CMP    #$10,D0           cas littéral
      BEQ    MM42              →
      MOVE   D4,D1             ↓ cas var/chaine
      ADD    D2,D4
      CMP    TVARU,D4
x     BCC    ERRPL             → pile_var trop petit
      MOVE   D4,(A3)
      MOVE   D4,2(SP)
      CMP    #$40,D0           cas chaine
x     BEQ    MM40              →
                               ↓ cas variables
      MOVE   TVARN,D0            vérifier les variables V(D1+1) à V(min(D4,TVARN))
      CMP    D0,D4                                                  D4
      BLE    MM36
      MOVE   D0,D4
```

```
MM36:CMP   D1,D4
     BLE   MM38          → fin de vérif
     MOVEM  D1/D4,-(SP)
     BSR   LB95C,
     BSR   VERVAR        vérifie  0(D4)
     MOVEM (SP)+,D1/D4
     SUBQ  #1,D4
     BRA   MM36
                                  pousser 0  tant que TVARN < 2(SP)
MM38:MOVE  2(SP),D0
     CMP   TVARN,D0
     BLE   MM44          → fin
     BSR   PUSHNZ        pousse 0
     BRA   MM38

MM40:MOVE  2(SP),D0                cas chaines: pousser ∅
     CMP   TVARN,D0                tant que TVARN < 2(SP)
     BLE   MM44          → fin
     BSR   PUSHNO        pousse ∅
     BRA   MM40

MM42:LEA   TVARLS,A0                cas littéral
     SUB   D2,(A0)
     MOVE  (A0)+,D2
     CMP   (A0),D2       ;tvarl
     BLE   ERRTL         → trop de var/lit

MM44:BSR   DECCRV
     BEQ   MM29
     BRA   MM30
```

```
MM46: CMP   #YCIND-Y,D0          ⊗
      BNE   ERRIS

                                   cas index

      BSR   SIZEIND
      MOVE  (A0),-(SP)             code de la taille

MM48: BSR   DECIN
      BEQ   MM30                   → fin
      MOVEM (SP),D4                taille
      MOVE  #$20,D3                index
      BSR   WNWLOC                 cache le nom/inscription de pile_proc
                                   A3 pointe la valeur A2 le type
                                   D2.L = 1   A1 pointe TMPPROC
      BNE   MM50                   → non suivi de (
      BSR   MA198                  ↓ suivi de (

MM50: MOVE  (SP),D0                size
      BSR   NBIND                  D1.L = nb d'index
      MOVE.W D2,D3                 D2.L = nb de mots long nécessaire
      ADD.L D2,D2                  vérif mémoire
      ADD.L D2,D2
      MOVE.L (A1)+,A6
      ADD.L D2,A0
      ADD   #20,A0
      CMP.L (A1),A0
      BCC   ERRGR                  pile_proc trop petit
      MOVE.L -(A1),A4
      MOVE.L -(A4),D0
      SUBQ  #4,A4                  pointe le
      MOVE.L A4,A3                 A3 1er index   met la table d'index à zéro
MM52: CLR.L (A4)+
      SUBQ.L #4,D2
      BPL   MM52
      MOVE.L D0,(A4)+
      MOVE.L A4,(A1)               tmproc
```

A3 1er index
D1.L nb d'index de la table pas encore initialisés
D2.L numéro de l'index à mettre

```
        MOVEQ #0,D2
MM54:   MOVE  4(SP),D3
X       BEQ   MM56          → il n'y a plus de données
        MOVE  (SP),D0       size
        MOVE.L 4(SP,D3.W),D3    valeur de l'index à mettre
        MOVEM.L D1/D2 /A3,-(SP)
        BSR   WNUMIK        met (A3,D2) reperant l'index
                            ca sera D3.L
        BSR   LB900         effectue l'assignation D3.L → (A3,D2)
        MOVEM.L (SP)+,D1/D2/A3
        SUBQ #4,4(SP)       diminue le nb de données
        ADDQ.L #1,D2
        SUBQ.L #1,D1
        BNE   MM54
        ↓ table remplie
MM56:   BSR   DECCRV
        BEQ   MM48
        BRA   MM30
```