Notes on the back story of this document:

The *HP-86/87* binary **`CMAT 1.0`**, was created by me at the time I was professionally working with these machines back in the 80's. It consists of 644 lines of *Capricorn* assembler code which I wrote to speed up my firm's engineering programs without requiring our clients to buy the very expensive and hard-to-get ***Matrix*** ROM(s).

`CMAT` implements 32 basic matrix functions and statements in a very novel way, namely using *strings* (say `Vertical_Loads$`) instead of array variables. This is the only way I found to implement array functions on my own, as the official *HP* documentation I had at hand for the *HP-86/87 Assembler ROM* (namely its *Owner's Handbook*) was extremely poor and gave very little details on the necessary calls to the System ROMs or the structures, polls and pointers involved. As there was no Internet back then and local *HP* people seemed to know even less than me, I opted for using strings, which at least were somewhat documented.

In the end, and although the functions/statements implemented were somewhat *ad-hoc* for our programs, the approach worked nicely and indeed they were significantly speeded up at no additional cost or inconvenience to our customers.

Eventually, I intended to upgrade it to a new version, **`CMAT 2.0`**, including many more and much more conventional matrix functions and statements, but could never get to do it because a decision was made by the powers-that-be to abandon *Series 80* and use *Series 200* machines instead. Those machines *(HP 9816/26/36)* were essentially *incompatible* with *Series 80*, their respective BASIC dialects were very different in many important aspects and the binaries were utterly incompatible too (*Capricorn* vs. *68000*) so our many programs would have to be rewritten from scratch, which was worth the trouble and the expense because the new machines were at least 15x-30x faster than the older ones and we had to remain competitive, so I wrote instead **`XLATOR`**, *a Series 80-to-Series 200* converter.

The source code is heavily commented (in Spanish) so you'll be able to easily follow it. Enjoy !

Valentin Albillo, 05-09-2021

```
10000 !*************************************************************************
10100 !*                                                                       *
10200 !*      CMAT - Operac. matriciales con strings (c) VALENTIN ALBILLO      *
10300 !*                                                                       *
10400 !*************************************************************************
10500 !
10600 BPGM#     EQU 53
10700           NAM 53,CMAT
10800           DEF RUNTIM
10900           DEF ASCIIS
11000           DEF PARSE
11100           DEF ERMSG
11200           DEF INIT
11300 !
11400 !-------------------------------------------------------------------------
11500 !
11600 RUNTIM    BSZ 2
11700           DEF REV
11800           DEF CALF
11900           DEF CNUM
12000           DEF CDIM
12100           DEF CALM
12200           DEF CREC
12300           DEF CMAX
12400           DEF CMIN
12500           DEF CSUM
12600           DEF CSMA
12700           DEF CRST
12800           DEF CMLT
12900           DEF CDVD
13000           DEF CSUV
13100           DEF CREV
13200           DEF CMUV
13300           DEF CDIV
13400           DEF C<=M
13500           DEF C<M
13600           DEF C=M
13700           DEF C>M
13800           DEF C>=M
13900           DEF C#M
14000           DEF C<=V
14100           DEF C<V
14200           DEF C=V
14300           DEF C>V
14400           DEF C>=V
14500           DEF C#V
14600           DEF CVAL
14700           DEF CCER
14800           DEF CREDIM
14900 !
15000 !-------------------------------------------------------------------------
15100 !
15200 PARSE     BSZ 2
15300           BSZ 2
```

```
15400           BSZ 2
15500           BSZ 2
15600           DEF CDIMPAR
15700           DEF CALMPAR
15800           BSZ 2
15900           BSZ 2
16000           BSZ 2
16100           BSZ 2
16200           BSZ 2
16300           BSZ 2
16400           BSZ 2
16500           BSZ 2
16600           BSZ 2
16700           BSZ 2
16800           BSZ 2
16900           BSZ 2
17000           BSZ 2
17100           BSZ 2
17200           BSZ 2
17300           BSZ 2
17400           BSZ 2
17500           BSZ 2
17600           BSZ 2
17700           BSZ 2
17800           BSZ 2
17900           BSZ 2
18000           BSZ 2
18100           BSZ 2
18200           DEF CVALPAR
18300           DEF CCERPAR
18400           DEF CDIMPAR
18500           BYT 377,377
18600 !
18700 !-----------------------------------------------------------------------------
18800 !
18900 ASCIIS    BSZ 0
19000           ASP "REVISION"
19100           ASP "CALF$"
19200           ASP "CNUM"
19300           ASP "CDIM"
19400           ASP "CALM"
19500           ASP "CREC"
19600           ASP "CMAX"
19700           ASP "CMIN"
19800           ASP "CSUMT"
19900           ASP "CSUMM"
20000           ASP "CRESM"
20100           ASP "CMULM"
20200           ASP "CDIVM"
20300           ASP "CSUMV"
20400           ASP "CRESV"
20500           ASP "CMULV"
20600           ASP "CDIVV"
20700           ASP "CMEIM"
20800           ASP "CMENM"
20900           ASP "CIGUM"
21000           ASP "CMAYM"
21100           ASP "CMAIM"
21200           ASP "CDISM"
21300           ASP "CMEIV"
```

```
21400           ASP  "CMENV"
21500           ASP  "CIGUV"
21600           ASP  "CMAYV"
21700           ASP  "CMAIV"
21800           ASP  "CDISV"
21900           ASP  "CVAL"
22000           ASP  "CCER"
22100           ASP  "CREDIM"
22200 !
22300 !-----------------------------------------------------------------------
22400 !
22500 ERMSG     BYT 200,200,200,200,200
22600           BYT 200,200,200,200
22700           ASP "nada por ahora"
22800           ASP "nada por ahora"
22900           ASP "$ NO TIENE 8 BYTES"
23000           ASP "DIM NO IGUALES"
23100           BYT 377
23200 !
23300 !-----------------------------------------------------------------------
23400 !
23500 INIT      RTN
23600 !
23700 !-----------------------------------------------------------------------
23800 !
23900 CALMPAR   BSZ 0                   ! Parse KEYWORD <str.>,<num>,<num>,<num>
24000           PUBD R43,+R6
24100           JSB =STREX+
24200           JEZ CLMPERR
24300           JSB =GETCMA
24400           JSB =NUMVAL
24500           JEZ CLMPERR
24600           JSB =GETCMA
24700           JSB =NUMVAL
24800           JEZ CLMPERR
24900           JSB =GETCMA
25000           JSB =NUMVAL
25100           JEZ CLMPERR
25200 CLMPOK    LDB R56,=IPGM#
25300           LDB R57,=371
25400           POBD R55,-R6
25500           STMI R55,=PTR2-
25600           RTN
25700 CLMPERR   POBD R57,-R6
25800           JSB =ERROR+
25900           BYT 88D
26000 !
26100 CDIMPAR   BSZ 0                   ! Parse KEYWORD <str.ref),<num>,<num>
26200           PUBD R43,+R6
26300           JSB =SCAN
26400           JSB =STRREF
26500           JEZ CLMPERR
26600           JSB =GETCMA
26700           JSB =NUMVAL
26800 CVALPEX   JEZ CLMPERR
26900           JSB =GETCMA
27000           JSB =NUMVAL
27100 CCERPEX   JEZ CLMPERR
27200           JMP CLMPOK
27300 !
```

```
27400 CVALPAR   BSZ 0                   ! Parse KEYWORD (str.ref),(num)
27500           PUBD R43,+R6
27600           JSB =SCAN
27700           JSB =STRREF
27800           JMP CVALPEX
27900 !
28000 CCERPAR   BSZ 0                   ! Parse KEYWORD (str.ref)
28100           PUBD R43,+R6
28200           JSB =SCAN
28300           JSB =STRREF
28400           JMP CCERPEX
28500 !
28600 !-----------------------------------------------------------------------------
28700 !
28800           BYT 0,56                ! Funcion string sin parametros
28900 REV       BIN
29000           LDM R43,=44D,0
29100           DEF DATE
29200           BYT 0
29300           ADMD R45,=BINTAB
29400           PUMD R43,+R12
29500           RTN
29600           ASC "38/21/03 .A.S ROCEHF -. ollibIA nitnelaV )c("
29700 DATE      BSZ 0
29800 !
29900 !-----------------------------------------------------------------------------
30000 !
30100 STORE     BSZ 8D                  ! lugar para almac. un numero
30200 !
30300 BINTAB    DAD 104070
30400 STREX+    DAD 23721
30500 STRREF    DAD 24056
30600 GETCMA    DAD 23477
30700 ADDROI    DAD 52745
30800 SUBROI    DAD 52724
30900 MPYROI    DAD 53517
31000 DIV2      DAD 52436
31100 LEQ.      DAD 62662
31200 LT.       DAD 62643
31300 EQ.       DAD 62623
31400 GR.       DAD 62705
31500 GEQ.      DAD 62734
31600 UNEQ.     DAD 62632
31700 MAX10     DAD 56144
31800 MIN10     DAD 56125
31900 NUMVAL    DAD 22406
32000 ERROR+    DAD 10220
32100 PTR2-     DAD 177715
32200 PTR1-     DAD 177711
32300 PTR1      DAD 177710
32400 PTR2      DAD 177714
32500 RESMEM    DAD 31741
32600 INTMUL    DAD 53673
32700 SCAN      DAD 21110
32800 ONEB      DAD 12153
32900 ERRBP#    DAD 103371
33000 ERROR     DAD 10224
33100 !
33200 !*****************************************************************************
33300 !*
```

```
33400 !* A$=CALF$(N) - Devuelve N convertido en string de 8 bytes
33500 !*
33600 !*************************************************************
33700 !
33800          BYT 20,56              ! Funcion string, 1 param. num.
33900 CALF     BIN
34000          LDM R55,=8D,0,0        ! reservamos 8 bytes
34100          JSB =RESMEM            !    id.
34200          POMD R40,-R12          ! sacamos el numero del stack
34300          STMD R65,=PTR2         ! PTR2 apunta a esta memoria reservada
34400          STMI R40,=PTR2-        ! mandamos el numero a la mem. res.
34500          LDM R46,=8D,0          ! long. de la string
34600          PUMD R46,+R12          ! al stack
34700          PUMD R65,+R12          ! y tambien la add.
34800          RTN
34900 !
35000 !*************************************************************
35100 !*
35200 !* A=CNUM(A$) - Devuelve A$ (8 bytes) convertida en numero
35300 !*
35400 !*************************************************************
35500 !
35600          BYT 30,55              ! Funcion numerica, 1 param. string
35700 CNUM     POMD R45,-R12          ! add. de la string
35800          POMD R30,-R12          ! long. de la string
35900          BIN                    ! para la comparacion
36000          CMM R30,=8D,0          ! es 8 ?
36100          JZR CNUMOK             ! si, vale
36200          LDB R20,=BPGM#         ! no, error
36300          STBD R20,=ERRBP#       ! preparamos el error
36400          JSB =ERROR             ! llamamos a la rutina de error
36500          BYT 244D               ! error 244
36600          POMD R36,-R6           ! eliminamos un RTN
36700          RTN                    ! y volvemos al sistema operativo
36800 CNUMOK   STMD R45,=PTR2         ! PTR2 apunta a la string.
36900          LDMI R50,=PTR2-        ! leemos los 8 bytes en R50-57
37000          PUMD R50,+R12          ! al stack
37100          RTN
37200 !
37300 !*************************************************************
37400 !*
37500 !* CCER A$ - Hace todos los elementos de A$ iguales a 0
37600 !*
37700 !*************************************************************
37800 !
37900          BYT 241                ! BASIC statement
38000 CCER     BIN                    ! para lo que haga falta
38100          LDM R70,=0,0,0,0,0,0,0,0 ! mandamos 0  <(CLM R70)
38200          PUMD R70,+R12          ! al stack
38300          JMP CVAL               ! CVAL se encargara del resto
38400 !
38500 !*************************************************************
38600 !*
38700 !* CVAL A$,N - Hace todos los elementos de A$ iguales a N
38800 !*
38900 !*************************************************************
39000 !
39100          BYT 241                ! BASIC statement
39200 CVAL     BIN                    ! para lo que haga falta
39300          POMD R70,-R12          ! sacamos N del stack
```

```
39400           POMD R65,-R12           ! saca la add. del stack
39500           POMD R50,-R12           ! limpiamos el stack (11 bytes)
39600           POMD R55,-R12           !   id.
39700           STMD R65,=PTR2          ! PTR2 apunta a la string
39800           LDMI R46,=PTR2-         ! leemos # elem. en R46-47
39900           LDMI R54,=PTR2-         ! saltamos F y C, PTR2 apunta al 1er el.
40000           JMP CDIMLOP             ! salimos por CDIMLOP que hace el resto
40100 !
40200 !*****************************************************************
40300 !*
40400 !* CDIM A$,F,C - Dim. A$ como matriz de F fil,C col, y la pone a 0
40500 !*
40600 !*****************************************************************
40700 !
40800           BYT 241                 ! BASIC statement
40900 CDIM      BIN                     ! para lo que haga falta
41000           LDMD R10,=BINTAB        ! llamaremos a
41100           JSB X10,CREDIM          ! la subrutina CREDIM
41200           LDM R70,=0,0,0,0,0,0,0,0 ! colocamos 0 en R70-77 (con R70)
41300 CDIMLOP   DCM R46                 ! si no quedan mas elementos
41400           JNG CDIMYAS             ! ya esta
41500           STMI R70,=PTR2-         ! mandamos el 0 a la string
41600           JMP CDIMLOP             ! y vamos a por otro elemento
41700 CDIMYAS   RTN
41800 !
41900 !*****************************************************************
42000 !*
42100 !* CREDIM A$,F,C - Redim. A$ como matriz de F fil,C col
42200 !*
42300 !*****************************************************************
42400 !
42500           BYT 241                 ! BASIC statement
42600 CREDIM    BIN                     ! para lo que haga falta
42700           JSB =ONEB               ! C a R46-47
42800           STM R46,R20             ! guardamos C en R20-21
42900           JSB =ONEB               ! F a R76-77
43000           STM R46,R22             ! guardamos F en R22-23
43100           STM R20,R66             ! pasamos C a R66-67
43200           JSB =INTMUL             ! hallamos F*C en R54-57
43300           LDM R46,R54             ! lo pasamos a R46-47
43400           STM R46,R56             ! a R56 para operar
43500           ADM R56,R56             !  *2
43600           ADM R56,R56             !  *4
43700           ADM R56,R56             !  *8
43800           ADM R56,=6,0            ! 8*F*C+6 bytes necesit.
43900           POMD R65,-R12           ! sac. la add. del stack
44000           POMD R70,-R12           ! limpiamos el stack
44100           POMD R75,-R12           !   id.  (11 bytes)
44200           ADM R65,=2,0,0          ! le sumamos 2 a la add.
44300           STMD R65,=PTR2          ! PTR2 apunta a la long. de $
44400           STMI R56,=PTR2-         ! long.=8*f*c+4
44500           STMI R46,=PTR2-         ! # elementos a la string.
44600           STMI R22,=PTR2-         ! F a la string
44700           STMI R20,=PTR2-         ! C    id.
44800           RTN
44900 !
45000 !*****************************************************************
45100 !*
45200 !*  CALM A$,I,J,N - Almacena el valor N en la fila I, col J de A$
45300 !*
```

```
45400 !*************************************************************************
45500 !
45600           BYT 241                 ! BASIC statement
45700 CALM      BIN                     ! para la aritmetica
45800           POMD R40,-R12           ! sacamos N del stack
45900           STM R46,R30             ! guardamos R46-47 en R30-31 (por ONEB)
46000           LDMD R10,=BINTAB        ! vamos a llamar a
46100           JSB X10,CARAUX          ! la subrutina que halla la add. de (I,J)
46200           STM R30,R46             ! reensamblamos el numero
46300           STMI R40,=PTR2-         ! lo colocamos en su sitio
46400           RTN
46500 !
46600 !*************************************************************************
46700 !*
46800 !*  N=CREC(A$,I,J) - Pone en N el valor del elemento I,J de A$
46900 !*
47000 !*************************************************************************
47100 !
47200           BYT 0,70,55             ! Fun.num.,1 param.string,2 numericos
47300 CREC      BIN                     ! para la aritmetica
47400           LDMD R10,=BINTAB        ! llamamos a la subrutina
47500           JSB X10,CARAUX          ! que calcula la add. de (I,J)
47600           LDMI R50,=PTR2-         ! leemos los 8 bytes en R50-57
47700           PUMD R50,+R12           ! al stack
47800           RTN
47900 !
48000 !-------------------------------------------------------------------------
48100 !
48200 ! CARAUX - Subrut. que parte de tener I,J en el stack, y vuelve con
48300 !           PTR2 apuntando al elemento que ocupa esa posicion en la $
48400 !-------------------------------------------------------------------------
48500 !
48600 CARAUX    JSB =ONEB               ! sacamos J del stack
48700           STM R46,R20             ! guardamos J en R20-21
48800           JSB =ONEB               ! sacamos I del stack a R76-77
48900           DCM R76                 ! hallamos I-1
49000           POMD R65,-R12           !   id. la add. del stack
49100           POMD R26,-R12           !   id. la long. (no nos hace falta)
49200           STMD R65,=PTR2          ! PTR2 apunta a la string
49300           LDMI R66,=PTR2-         ! pasamos sobre el #elem.
49400           LDMI R66,=PTR2-         ! leemos F en R66-67
49500           LDMI R66,=PTR2-         ! leemos C en R66-67
49600           JSB =INTMUL             ! C*(I-1)
49700           LDM R56,R54             ! este valor a R56-57
49800           ADM R56,R20             ! C*(I-1)+J en R56-57
49900           DCM R56                 ! C*(I-1)+J-1
50000           ADM R56,R56             !  *2
50100           ADM R56,R56             !  *4
50200           ADM R56,R56             !  *8
50300           STM R56,R55             ! ponemos el offset en R55-57
50400           CLB R57                 ! el 3er byte es 0
50500           LDMD R75,=PTR2          ! recuperamos PTR2
50600           SBM R75,R55             ! le restamos el offset al PTR2
50700           STMD R75,=PTR2          ! PTR2 ahora apunta a la posicion
50800           RTN
50900 !
51000 !*************************************************************************
51100 !*
51200 !* A=CMAX(A$)  - Da el valor del mayor elemento de la matriz A$
51300 !* A=CMIN(A$)  -        "       menor                 "
```

```
51400 !* A=CSUMT(A$) -          "     de la suma de todos los elementos de A$
51500 !*
51600 !*************************************************************************
51700 !
51800           BYT 30,55                ! F. num, 1 param. string
51900 CMAX      LDM R46,=MAX10           ! MAX
52000           JMP CMAXAUX              !
52100           BYT 30,55                !
52200 CMIN      LDM R46,=MIN10           ! MIN
52300           JMP CMAXAUX              !
52400           BYT 30,55                !
52500 CSUM      LDM R46,=ADDROI          ! +
52600 CMAXAUX   BIN                      ! para lo que haga falta
52700           LDMD R26,=BINTAB         ! para alm.rel.
52800           LDB R45,=316             ! esto completa el JSB=oper
52900           STMD R45,X26,CMAXOP      ! colocamos todo en CMAXOP
53000           POMD R65,-R12            ! sacamos la add. del stack
53100           POMD R26,-R12            !   id. la long.
53200           STMD R65,=PTR2           ! PTR2 apunta a la string
53300           LDMI R26,=PTR2-          ! leemos en R26-27 el $elem.
53400           LDMI R64,=PTR2-          ! pasamos sobre F y C
53500           LDMI R50,=PTR2-          ! sacamos el 1er elem.de la $
53600           PUMD R50,+R12            ! lo mandamos al stack
53700           DCM R26                  ! R26-27: F*C-1
53800 CMAXLOP   BIN                      ! para el decremento
53900           DCM R26                  ! si no quedan mas elementos
54000           JNG CMAXYAS              ! ya esta
54100           LDMI R50,=PTR2-          ! sacamos otro elemento
54200           PUMD R50,+R12            ! al stack
54300           BCD                      ! para MAX10
54400 CMAXOP    BSZ 3                    ! realiza la operacion
54500           JMP CMAXLOP              ! repetimos
54600 CMAXYAS   RTN
54700 !
54800 !*************************************************************************
54900 !*
55000 !* A$=CSUMM(B$,C$) - Suma dos matrices elemento a elemento
55100 !* A$=CRESM(B$,C$) - Resta                  "
55200 !* A$=CMULM(B$,C$) - Multiplica             "
55300 !* A$=CDIVM(B$,C$) - Divide                  "
55400 !* A$=CMEIM(B$,C$) - Halla B$(=C$            "
55500 !* A$=CMENM(B$,C$) -  id.   B$( C$           "
55600 !* A$=CIGUM(B$,C$) -  id.   B$= C$           "
55700 !* A$=CMAYM(B$,C$) -  id.   B$) C$           "
55800 !* A$=CMAIM(B$,C$) -  id.   B$)=C$          "
55900 !* A$=CDISM(B$,C$) -  id.   B$()C$          "
56000 !*
56100 !*************************************************************************
56200 !
56300           BYT 52,56                ! F. string, 2 arg. string
56400 CSMA      LDM R46,=ADDROI          ! +
56500           JMP COPERAUX             !
56600           BYT 52,56                !
56700 CRST      LDM R46,=SUBROI          ! -
56800           JMP COPERAUX             !
56900           BYT 52,56                !
57000 CMLT      LDM R46,=MPYROI          ! *
57100           JMP COPERAUX             !
57200           BYT 52,56                !
57300 CDVD      LDM R46,=DIV2            ! /
```

```
57400            JMP COPERAUX          !
57500            BYT 52,56             !
57600 C<=M       LDM R46,=LEQ.         !
57700            JMP COPERAUX          !
57800            BYT 52,56             !
57900 C<M        LDM R46,=LT.          !
58000            JMP COPERAUX          !
58100            BYT 52,56             !
58200 C=M        LDM R46,=EQ.          !
58300            JMP COPERAUX          !
58400            BYT 52,56             !
58500 C>M        LDM R46,=GR.          !
58600            JMP COPERAUX          !
58700            BYT 52,56             !
58800 C>=M       LDM R46,=GEQ.         !
58900            JMP COPERAUX          !
59000            BYT 52,56             !
59100 C#M        LDM R46,=UNEQ.        !
59200 COPERAUX   BIN                   ! para lo que haga falta
59300            LDMD R26,=BINTAB      ! para almac. rel.
59400            LDB R45,=316          ! esto es el JSB
59500            STMD R45,X26,CSMAOP   ! colocamos todo en OPERAC
59600            LDMD R55,=PTR1        ! guardamos PTR1
59700            PUMD R55,+R6          ! en el stack R6
59800            POMD R45,-R12         ! add. de C$
59900            POMD R30,-R12         ! long. de C$
60000            STMD R45,=PTR2        ! PTR2 apunta a C$
60100            LDMI R32,=PTR2-       ! leemos #elem. en R32-33
60200            LDMI R20,=PTR2-       ! leemos FC en R20-21
60300            LDMI R22,=PTR2-       !  id.   CC en R22-23
60400            LDMD R75,=PTR2        ! leemos la nueva add. de C$
60500            PUMD R75,+R6          ! y la mand. al +R6
60600            POMD R55,-R12         ! add. de B$
60700            POMD R30,-R12         ! long. de B$ (debe ser la misma)
60800            STMD R55,=PTR2        ! PTR2 apunta a B$
60900            LDMI R32,=PTR2-       ! leemos #elem. en R32-33
61000            LDMI R24,=PTR2-       ! leemos FB en R24-25
61100            LDMI R26,=PTR2-       !  id.   CB en R26-27
61200            LDMD R75,=PTR2        ! leemos la nueva add. de B$
61300            PUMD R75,+R6          ! y la mand. al +R6
61400            CMM R20,R24           ! FC=FB ?
61500            JNZ CSMAERR           ! no, error
61600            CMM R22,R26           ! CC=CB ?
61700            JNZ CSMAERR           ! no,error
61800            PUMD R30,+R12         ! la long. A$ al stack
61900            STM R30,R55           ! ponemos la long. en R55-57
62000            CLB R57               ! solo 2 bytes
62100            JSB =RESMEM           ! reservamos memoria
62200            STMD R65,=PTR2        ! PTR2 apunta a la mem.lib.
62300            STMI R32,=PTR2-       ! # elem. a la string
62400            STMI R24,=PTR2-       ! F a la string
62500            STMI R26,=PTR2-       ! C a la string
62600            PUMD R65,+R12         ! add. al stack
62700            LDM R26,R32           ! num. elem. en R26-27
62800 CSMALOP    BIN                   ! para el ciclo
62900            DCM R26               ! quedan mas ?
63000            JNG CSMAYAS           ! no, se acabo
63100            POMD R55,-R6          ! sac. add. de B$ de R6
63200            STMD R55,=PTR1        ! PTR1 apunta a B$
63300            LDMI R70,=PTR1-       ! leemos B$(I,J)
```

```
63400           LDMD R55,=PTR1          ! actualizamos la add.
63500           PUMD R70,+R12           ! al stack operativo
63600           POMD R65,-R6            ! sac. add. de C$ de R6
63700           STMD R65,=PTR1          ! PTR1 apunta a C$
63800           LDMI R70,=PTR1-         ! leemos C$(I,J)
63900           LDMD R65,=PTR1          ! actualizamos la add.
64000           PUMD R70,+R12           ! al stack
64100           PUMD R65,+R6            ! reponemos add. C$
64200           PUMD R55,+R6            !     id.   add. B$
64300           BCD                     ! para la +
64400 CSMAOP    BSZ 3                   ! B$ (oper) C$
64500           POMD R70,-R12           ! lo sacamos del stack
64600           STMI R70,=PTR2-         ! y lo mandamos a A$
64700           JMP CSMALOP             ! repetimos
64800 CSMAYAS   POMD R52,-R6            ! limpiamos el stack R6
64900           POMD R55,-R6            ! recuperamos PTR1
65000           STMD R55,=PTR1          ! y lo reponemos
65100           RTN
65200 CSMAERR   POMD R52,-R6            ! limpiamos el R6
65300           POMD R55,-R6            ! recuperamos PTR1
65400           STMD R55,=PTR1          ! lo reponemos
65500           LDB R20,=BPGM#          ! preparamos el err.
65600           STBD R20,=ERRBP#        !    id.
65700           JSB =ERROR              !    id.
65800           BYT 243D                ! error 243
65900           POMD R36,-R6            ! eliminamos un RTN
66000           RTN
66100 !
66200 !*************************************************************************
66300 !*
66400 !* A$=CSUMV(B$,N) - Suma el valor N a todos los elementos de B$
66500 !* A$=CRESV(B$,N) - Resta                    "
66600 !* A$=CMULV(B$,N) - Multiplica               "
66700 !* A$=CDIVV(B$,N) - Divide                   "
66800 !* A$=CMEIV(B$,N) - Halla B$(=N              "
66900 !* A$=CMENV(B$,N) -  id.  B$< N              "
67000 !* A$=CIGUV(B$,N) -  id.  B$= N              "
67100 !* A$=CMAYV(B$,N) -  id.  B$> N              "
67200 !* A$=CMAIV(B$,N) -  id.  B$>=N             "
67300 !* A$=CDISV(B$,N) -  id.  B$<>N             "
67400 !*
67500 !*************************************************************************
67600 !
67700           BYT 50,56               ! F. string, 2 arg. string
67800 CSUV      LDM R46,=ADDROI         ! +
67900           JMP VALERAUX            !
68000           BYT 50,56               !
68100 CREV      LDM R46,=SUBROI         ! -
68200           JMP VALERAUX            !
68300           BYT 50,56               !
68400 CMUV      LDM R46,=MPYROI         ! *
68500           JMP VALERAUX            !
68600           BYT 50,56               !
68700 CDIV      LDM R46,=DIV2           ! /
68800           JMP VALERAUX            !
68900           BYT 50,56               !
69000 C<=V      LDM R46,=LEQ.           !
69100           JMP VALERAUX            !
69200           BYT 50,56               !
69300 C<V       LDM R46,=LT.            !
```

```
69400           JMP VALERAUX            !
69500           BYT 50,56               !
69600 C=V       LDM R46,=EQ.            !
69700           JMP VALERAUX            !
69800           BYT 50,56               !
69900 C>V       LDM R46,=GR.            !
70000           JMP VALERAUX            !
70100           BYT 50,56               !
70200 C>=V      LDM R46,=GEQ.           !
70300           JMP VALERAUX            !
70400           BYT 50,56
70500 C#V       LDM R46,=UNEQ.
70600 VALERAUX  BIN                     ! para lo que haga falta
70700           LDMD R26,=BINTAB        ! para almac. rel.
70800           LDB R45,=316            ! esto es el JSB
70900           STMD R45,X26,VALOP      ! colocamos todo en OPERAC
71000           LDMD R55,=PTR1          ! guardamos PTR1
71100           PUMD R55,+R6            ! en +R6
71200           POMD R50,-R12           ! sacamos N del stack
71300           STMD R50,X26,STORE      ! lo guardamos
71400           POMD R45,-R12           !  id. add. B$
71500           POMD R22,-R12           !  id. long. B$ en R22-23
71600           STMD R45,=PTR2          ! PTR2 apunta a B$
71700           STM R22,R55             ! prepara RESMEM
71800           CLB R57                 ! solo 2 bytes
71900           JSB =RESMEM             ! reservamos memoria
72000           STMD R65,=PTR1          ! PTR1 apunta a la memoria
72100           PUMD R22,+R12           ! long. al stack
72200           PUMD R65,+R12           ! add. al stack
72300           LDMI R20,=PTR2-         ! # elem. a R20-21
72400           LDMI R44,=PTR2-         ! leemos F,C en R44-47
72500           STMI R20,=PTR1-         ! # elem. a la str.
72600           STMI R44,=PTR1-         ! F,C a la string
72700 VALLOP    BIN                     ! para el dec.
72800           DCM R20                 ! si no quedan mas
72900           JNG VALYAS              ! ya esta
73000           LDMI R40,=PTR2-         ! leemos un elemento
73100           PUMD R40,+R12           ! al stack
73200           LDMD R50,X26,STORE      ! recuperamos N
73300           PUMD R50,+R12           ! al stack
73400           BCD                     ! para la operac.
73500 VALOP     BSZ 3                   ! operacion
73600           POMD R50,-R12           ! sacamos el resultado
73700           STMI R50,=PTR1-         ! y lo mandamos a la str.
73800           JMP VALLOP              ! repetimos
73900 VALYAS    POMD R55,-R6            ! restauramos PTR1
74000           STMD R55,=PTR1          ! a su valor original
74100           RTN
74200 !
74300           FIN
```