```
SOURCE ACCESS NAME=      PPC2.P359.V081180.SRC.FORNEXT
OBJECT ACCESS NAME=      PPC2.P359.OBJ.FORNEXTS
LISTING ACCESS NAME=     PPC2.P359.V081180.LST.FORNEXTS
ERROR ACCESS NAME=       .XMAERR
OPTIONS=                 /REF
MACRO LIBRARY PATHNAME=


LINE      KEY   NAME

 0002      A    VERSION
                    =>PPC2.P359.V081180.SRC.P359
```

```
       IDT  'FORNXT'
*************************************************************
*
* FFFFFF  OOOO   RRRRR       N     N  EEEEE X     X  TTTTTTT*
* F      O    O  R    R      NN    N  E      X   X      T   *
* F      O    O  R    R      N N   N  E       X X       T   *
* FFF    O    O  RRRRR  ===  N  N  N  EEE      X        T   *
* F      O    O  R  R        N   N N  E       X X       T   *
* F      O    O  R   R       N    NN  E      X   X      T   *
* F       OOOO   R    R      N     N  EEEEE X     X     T   *
*                                                           *
*           PPPP      3333     555555      9999             *
*           P   P    3     3   5          9    9            *
*           P   P          3   5          9    9            *
*           PPPP       3333    55555       99999            *
*           P              3         5         9            *
*           P        3     3   5     5    9    9            *
*           P         3333      5555       9999             *
*                                                           *
*************************************************************
```

```
0055                    DEF   NFOR,NNEXT,C8
0056 0000
0057                    REF   VDPRD,EOSTMT,EOLINE,STLN
0058                    REF   C4,C24,C40,CBH67,FLTONE
0059                    REF   FAC,FAC2,FAC6,ERRMUV
0060                    REF   BUFLEV,VSPTR,PGMPTR,PGMPT1,EXTRAM,EXTRM1,PRGFLG
0061                    REF   SYM,GETV,VPUSH,VPOP,ASSG,SMB,GET1,GETV1,MOVFAC
0062                    REF   SAVREG,SETREG,SADD,SCOMPB,VPOP20,ERRSYN,ERRT
0063                    REF   PGMCHR,PSHPRS,PARSE,CONT,PUTV1
0064                    REF   R1LB,STVSPT,GOTO90,ERR
0065 0000
0066       0082  SSEP$  EQU   >82
0067       0083  TREM$  EQU   >83
0068       008C  FOR$   EQU   >8C
0069       0096  NEXT$  EQU   >96
0070       00A1  SUB$   EQU   >A1
0071       00B1  TO$    EQU   >B1
0072       00B2  STEP$  EQU   >B2
0073       00BE  EQ$    EQU   >BE
0074       00C7  STRIN$ EQU   >C7
0075       00C9  LN$    EQU   >C9
```

```
0077            ****************************************************************
0078            *       FOR Statement                                          *
0079            *       Builds up a stack entry for the FOR statement.         *
0080            *       Checks the syntax of a FOR statement and also          *
0081            *       checks to see if the loop is executed at all.          *
0082            *       The loop is not executed if the limit of the           *
0083            *       of the FOR is > the initial value and the step         *
0084            *       is positive or the limit of the FOR is < the           *
0085            *       initial value and the step is negative.                *
0086            *                                                              *
0087            *        A stack entry for a 'FOR' stmt looks like:             *
0088            *                                                              *
0089            *  +-------------------------------------------------------+   *
0090            *  | PTR TO S.T.  | >67 |      | Value Space |   BUFLEV     |   *
0091            *  |   ENTRY      |     |      |   Pointer   |              |   *
0092            *  |--------------------------------------------------------+   *
0093            *  | FOR line # |  FOR  line |                              |   *
0094            *  | table ptr  |   Pointer  |                              |   *
0095            *  |--------------------------------------------------------+   *
0096            *  |                    Increment Value                     |   *
0097            *  |--------------------------------------------------------+   *
0098            *  |                        Limit                           |   *
0099            *  +-------------------------------------------------------+   *
0100            ****************************************************************
0101 0000
0102 0000 D208  NFOR    MOVB R8,R8              EOL?
0103 0002 1501          JGT  NFOR1              If symbol name - OK
0104 0004 107C          JMP  ERRCDT             If EOL or Token - ERROR
0105 0006 06A0  NFOR1   BL   @SYM               Get ptr to s.t. entry
     0008 0000
0106 000A 06A0          BL   @GETV              Get 1st byte of symbol
     000C 0000
0107 000E 0000          DATA FAC                  entry
0108 0010 0241          ANDI R1,>C700           Check string,func & array
     0012 C700
0109 0014 1670          JNE  BERMUV             If any of the above-error
0110 0016-0288          CI   R8,EQ$*256         Must have '='
     0018 BE00
0111 001A 1671          JNE  ERRCDT             If not - error
0112 001C 06A0          BL   @SMB               Get index's value space
     001E 0000
0113 0020 04E0          CLR  @FAC2              Dummy entry ID on the stack
     0022 0000
0114 0024 C820          MOV  @BUFLEV,@FAC6      Save buffer level
     0026 0000
     0028 0000
0115            *
0116            *       Search stack for another FOR entry with the
0117            *           same loop variable.  If one is found, remove it.
0118            *
0119 002A C0E0          MOV  @VSPTR,R3          Copy stack pointer
     002C 0000
0120            *
0121            *       See if end of stack
0122 002E 8803  NFOR1A  C    R3,@STVSPT         Check stack underflow
     0030 0000
0123 0032 1228          JLE  NFOR1E             Finished with stack scan
0124            *       See if FOR entry
0125 0034 06A0          BL   @GET1              Get ptr to s.t. entry
     0036 0000
```

```
0126 0038 C001          MOV  R1,R0              Move it to use later
0127                 *--------------------CONDITIONAL ASSEMBLY--------------------*
0128                    ASMIF VERS=DX1C
0129                    MOVB *R14+,R1           Read stack ID
0130
0131                    ASMELS
0132 003A
0133 003A D060          MOVB @VDPRD,R1          Read stack ID
     003C 0000
0134                    ASMEND
0135                 *------------------END OF CONDITIONAL ASSEMBLY---------------*
0136 003E 9801          CB   R1,@CBH67          Is stack entry a FOR?
     0040 0000
0137 0042 1606          JNE  NFOR1B             No - 8-byte regular entry
0138 0044
0139                 *  Compare loop variables
0140 0044 8800          C    R0,@FAC            Loop vars match?
     0046 000E'
0141 0048 1309          JEQ  NFOR1C             Yes
0142 004A 0223          AI   R3,-32             Skip this FOR entry
     004C FFE0
0143 004E 10EF          JMP  NFOR1A             Loop
0144 0050 9801  NFOR1B  CB   R1,@CBH6A          Hit a subprogram entry?
     0052 00AF'
0145 0054 1317          JEQ  NFOR1E             Yes-don't scan anymore
0146 0056 0223          AI   R3,-8              Skip 8-byte stack entry
     0058 FFF8
0147 005A 10E9          JMP  NFOR1A             Loop
0148 005C
0149                 *  Found matching loop variable
0150                 *  Move stack down 32 bytes
0151 005C
0152 005C C0A0  NFOR1C  MOV  @VSPTR,R2          Copy stack pointer
     005E 002C'
0153 0060 6083          S    R3,R2              Calc. # of bytes to move
0154 0062 130D          JEQ  NFOR1D             0 bytes, skip move
0155 0064 C103          MOV  R3,R4              Destination pointer
0156 0066 0224          AI   R4,-24             Place to move to
     0068 FFE8
0157      006C' C8      EQU  $+2
0158 006A 0223          AI   R3,8               Point at entry above FOR entry
     006C 0008
0159 006E 06A0  NFOR1F  BL   @GETV1             Get the byte
     0070 0000
0160 0072 06A0          BL   @PUTV1             Put the byte
     0074 0000
0161 0076 0583          INC  R3                 Inc From pointer
0162 0078 0584          INC  R4                 Inc to pointer
0163 007A 0602          DEC  R2                 Decrement counter
0164 007C 16F8          JNE  NFOR1F             Loop if not done
0165 007E
0166 007E 6820  NFOR1D  S    @C32,@VSPTR        Adjust top of stack
     0080 0198'
     0082 005E'
0167 0084
0168                 *  Now put new FOR entry on stack
0169 0084 06A0  NFOR1E  BL   @VPUSH             Reserve space for limit,
     0086 0000
0170 0088 06A0          BL   @VPUSH                 increment,
     008A 0086'
```

```
                                        BL    @VPUSH                    and 2nd info entry
0171 008C 06A0
     008E 008A'
                                        MOVB  @CBH67,@FAC2              FOR ID on stack
0172 0090 D820
     0092 0040'
     0094 0022'
                                        BL    @PGMCHR                   Get next character
0173 0096 06A0
     0098 0000
                                        BL    @PSHPRS                   Push symbol I.D. entry
0174 009A 06A0
     009C 0000
                                        BYTE  TO$                       Parse the initial value
0175 009E   B1
0176 009F   63  CBH63                   BYTE  >63                       Use the wasted byte
0177 00A0-0288                          CI    R8,TO$*256                TO?
     00A2 B100
                                        JNE   ERRCDT                    No - error
0178 00A4 162C
                                        BL    @PGMCHR
0179 00A6 06A0
     00A8 0098'
                                        BL    @PSHPRS                   Push initial and get limit
0180 00AA 06A0
     00AC 009C'
                                        BYTE  STEP$
0181 00AE   B2
0182 00AF   6A  CBH6A                   BYTE  >6A                       Constant >6A
0183 00B0 9820                          CB    @CBH63,@FAC2              If a string value
     00B2 009F'
     00B4 0094'
                                        JL    BERR6                     Its an error
0184 00B6 1A1D
0185 00B8 6820                          S     @C40,@VSPTR
     00BA 0000
     00BC 0082'
                                        BL    @VPUSH                    Push the limit
0186 00BE 06A0
     00C0 008E'
                                        BL    @EOSTMT                   At the end of statement?
0187 00C2 06A0
     00C4 0000
                                        JEQ   NFOR2                     Yes-default incr to 1
0188 00C6 131D
                                        CI    R8,STEP$*256              STEP?
0189 00C8-0288
     00CA B200
                                        JNE   ERRCDT                    No - Its an error
0190 00CC 1618
                                        A     @C32,@VSPTR               Correct stack pointer
0191 00CE A820
     00D0 0198'
     00D2 00BC'
                                        BL    @PGMCHR
0192 00D4 06A0
     00D6 00A8'
                                        BL    @PARSE                    Get the increment
0193 00D8 06A0
     00DA 0000
                                        BYTE  TREM$,0
0194 00DC   83
     00DD   00
                                        S     @C32,@VSPTR               Get stack to needed place
0195 00DE 6820
     00E0 0198'
     00E2 00D2'
                                        MOV   @FAC,R0                   Can't have zero increment
0196 00E4 C020
     00E6 0046'
                                        JEQ   ERRBV                     If 0, its an error
0197 00E8 1308
                                        CB    @CBH63,@FAC2              Can't have string value
0198 00EA 9820
     00EC 009F'
     00EE 00B4'
                                        JHE   NFOR3                     If numeric - OK
0199 00F0 140F
0200 00F2 0460  BERR6                   B     @ERRT                     * STRING NUMBER MISMATCH
     00F4 0000
0201 00F6 0460  BERMUV                  B     @ERRMUV                   * MULTIPLY USED VARIABLE
     00F8 0000
0202 00FA 0460  ERRBV                   B     @GOTO90
     00FC 0000
```

```
02C3 C0FE 0460  ERRCDT B    @ERRSYN
     0100 0000
0204 0102 0200  NFOR2  LI   R0,FAC
     0104 00E6'
0205 0106 C020         MOV  @FLTONE,*R0+      Put a floating one in
     0108 0000
0206 010A 04F0         CLR  *R0+
0207 010C 04F0         CLR  *R0+
0208 010E 04D0         CLR  *R0
0209 0110 06A0  NFOR3  BL   @VPUSH            Push the step
     0112 00C0'
0210 0114 0201         LI   R1,FAC            Optimize to save bytes
     0116 0104'
0211 0118 CC60         MOV  @EXTRAM,*R1+      Save line # pointer
     011A 0000
0212 011C C460         MOV  @PGMPTR,*R1       Save ptr w/in the line
     011E 0000
0213 0120 0611         DEC  *R1               Back up so get last char
0214 0122 06A0         BL   @VPUSH            Push it too!
     0124 0112'
0215 0126 A820         A    @C16,@VSPTR       Point to initial value
     0128 0156'
     012A 00E2'
0216 012C 06A0         BL   @VPOP             Get initial value
     012E 0000
0217 0130 06A0         BL   @ASSG             Assign it
     0132 0000
0218 0134 A820         A    @C8,@VSPTR        Restore to top of entry
     0136 006C'
     0138 012A'
0219 013A
0220              *    Check to see if execute loop at all
0221 013A
0222 013A 06A0         BL   @VPOP             Get ptr to value
     013C 012E'
0223 013E 06A0         BL   @MOVFAC           Get value
     0140 0000
0224 0142 6820         S    @C16,@VSPTR       Point at limit
     0144 0156'
     0146 0138'
0225 0148 06A0         BL   @SCOMPB           Compare them
     014A 0000
0226              *    VSPTR is now below the FOR entry
0227 014C 02C4         STST R4                Save the status
0228 014E 1309         JEQ  NFOR03            If =
0229 0150 C0E0         MOV  @VSPTR,R3
     0152 0146'
0230      0156' C16    EQU  $+2
0231 0154 0223         AI   R3,16
     0156 0010
0232 0158 06A0         BL   @GETV1            Check negative step
     015A 0070'
0233 015C 1107         JLT  NFOR05            If a decrement
0234 015E 0A14         SLA  R4,1              Check out of limit
0235 0160 1507         JGT  NFOR07            Out of limit
0236 0162 A820  NFOR03 A    @C32,@VSPTR       Leave the entry on
     0164 0198'
     0166 0152'
0237 0168 0460         B    @CONT     <<<<<<< Result is w/in limit
     016A 0000
```

```
0238 016C
0239 016C 0A14  NFOR05 SLA  R4,1              Check out of limit
0240 016E 15F9         JGT  NFOR03            Result is w/in limit
0241 0170
0242                   *     Initial value is not within the limit.
0243                   *     Therefore, the loop is not executed at all.  Must
0244                   *     skip the code in the body of the loop
0245 0170
0246 0170 0203  NFOR07 LI   R3,1              FOR/NEXT pair counter
     0172 0001
0247 0174 06A0  NFOR09 BL   @EOLINE           Check end of line
     0176 0000
0248 0178 1339         JEQ  NFOR13            Is end of line
0249 017A 06A0         BL   @PGMCHR           Get 1st token on line
     017C 00D6'
0250 017E-0288 NFOR10 CI   R8,NEXT$*256       If NEXT
     0180 9600
0251 0182 1619         JNE  NFOR11            If not
0252 0184 0603         DEC  R3                Decrement counter
0253 0186 162C         JNE  NFOR12            If NOT matching next
0254 0188 06A0         BL   @PGMCHR           Get 1st char of loop var
     018A 017C'
0255 018C 1183         JLT  ERRCDT            If token
0256 018E 06A0         BL   @SYM              Get s.t. pointer to check mat
     0190 0008'
0257 0192 C0E0         MOV  @VSPTR,R3         Correct to top of entry
     0194 0166'
0258      0198' C32    EQU  $+2
0259 0196 0223         AI   R3,32
     0198 0020
0260 019A 06A0         BL   @GET1             Get pointer
     019C 0036'
0261 019E 8801         C    R1,@FAC           Match?
     01A0 0116'
0262 01A2 1605         JNE  ERRFNN            No match
0263 01A4 0460         B    @CONT             Continue  <<<<<<  THE WAY
     01A6 016A'
0264 01A8 A820  ERRFN  A    @C4,@EXTRAM
     01AA 0000
     01AC 011A'
0265 01AE 0200  ERRFNN LI   R0,>0B03          FOR NEXT NESTING
     01B0 0B03
0266 01B2 0460         B    @ERR
     01B4 0000
0267                   *
0268                   *
0269 01B6-0288 NFOR11 CI   R8,SUB$*256        Hit a SUB?
     01B8 A100
0270 01BA 13F9         JEQ  ERRFNN            Yes-can't find matching next
0271 01BC-0288         CI   R8,FOR$*256       FOR?
     01BE 8C00
0272 01C0 1601         JNE   NFOR20           No.. Check some more.
0273 01C2 0583         INC   R3               Increment depth
0274 01C4-0288 NFOR20 CI   R8,LN$*256         Line number token?
     01C6 C900
0275 01C8 1602         JNE  NFOR30            No.. Check some more.
0276 01CA 05E0         INCT @PGMPTR           Skip the line number.
     01CC 011E'
0277 01CE-0288 NFOR30 CI   R8,STRIN$*256      String?
     01D0 C700
```

```
0302                    *       NEXT4 and NEXT2A were moved from in-line to here
0303                    *       in an effort to make the "normal" path through
0304                    *       the NEXT code as straight-line as possible.
0305 021E
0306 021E 6820  NEXT4   S       @C24,@VSPTR         LOOP VARIABLES DON'T MATCH
     0220 0C00
     0222 0194'
0307 0224 1006          JMP     NEXT2
0308 0226 06A0  NEXT2B  BL      @VPUSH              Keep stack information.
     0228 0124'
0309 022A
0310 022A 0200  NEXT2A  LI      R0,>0C03            NEXT WITHOUT FOR
     022C 0C03
0311 022E 0460          B       @ERR
     0230 0184'
0312                    ********************************************************
0313                    *                                                      *
0314                    *       NEXT statement handler - find the matching FOR *
0315                    *       statement on the stack, add the increment to the *
0316                    *       current value of the inder variable and check to *
0317                    *       see if execute the loop again.  If loop-variable's *
0318                    *       value is still within bounds, goto the top of the *
0319                    *       loop, otherwise, flush the FOR entry off the stack *
0320                    *       and continue with the statement following the NEXT *
0321                    *       statement.                                     *
0322                    *                                                      *
0323                    ********************************************************
0324 0232 06A0  NNEXT   BL      @SYM                GET S.T.  I.D.
     0234 0190'
0325                    *       MOV     @FAC,R4     SYM/FBSYMB leaves value in R4
0326 0236 8820  NEXT2   C       @VSPTR,@STVSPT      CHECK FOR BOTTOM OF STACK
     0238 0222'
     023A 0030'
0327 023C 12F6          JLE     NEXT2A              IF AT BOTTOM ->NEXT W/O FOR
0328 023E 06A0          BL      @VPOP               GET 'FOR' ENTRY OFF STACK
     0240 013C'
0329 0242 9820          CB      @FAC2,@CBH67        CHECK FOR 'FOR' ENTRY
     0244 00EE'
     0246 0092'
0330 0248 16EE          JNE     NEXT2B              Is not a 'FOR' entry-error
0331 024A 8804          C       R4,@FAC             CHECK IF MATCHING 'FOR' ENTRY
     024C 01A0'
0332 024E 16E7          JNE     NEXT4               Is not a match - so check more
0333 0250 C0E0          MOV     @VSPTR,R3           Check BUFLEV for match
     0252 0238'
0334 0254 0223          AI      R3,14               Point at the BUFLEV in stack
     0256 000E
0335 0258 06A0          BL      @GET1               Read it
     025A 019C'
0336 025C 8801          C       @FAC6,@BUFLEV       SAME LEVEL?
     025E 0026'
0337 0260 16A6          JNE     ERRFNN              NO - ITS AN ERROR
0338 0262 6820          S       @C8,@VSPTR
     0264 006C'
     0266 0252'
0339 0268 06A0          BL      @MOVFAC             GET INDEX VALUE
     026A 0140'
0340 026C 06A0          BL      @SAVREG             SAVE BASIC REGS
     026E 0000
0341 0270 06A0          BL      @SADD               ADD IN THE INCREMENT
```

```
     0272 0000
0342 0274 06A0          BL    @SETREG          RESTORE BASIC REGS
     0276 0000
0343 0278 A820          A     @C24,@VSPTR
     027A 0220'
     027C 0266'
0344 027E 06A0          BL    @ASSG            SAVE NEW INDEX VALUE
     0280 0122'
0345 0282 6820          S     @C16,@VSPTR      POINT TO THE LIMIT
     0284 0156'
     0286 027C'
0346 0288 06A0          BL    @SCOMPB          TEST W/IN LIMIT
     028A 014A'
0347 028C 02C4          STST R4                SAVE RESULT OF COMPARE
0348 028E 1309          JEQ   NEXT5            IF = DO LAST LOOP
0349 0290 C0E0          MOV   @VSPTR,R3        CHECK FOR A DECREMENT
     0292 0286'
0350 0294 0223          AI    R3,16            Point at incr/decr
     0296 0010
0351 0298 06A0          BL    @GETV1           Get 1st byte and set condition
     029A 015A'
0352 029C 1116          JLT   NEXT6            If was a decrement
0353 029E 0A14          SLA   R4,1             Check if out of limit
0354 02A0 1512          JGT   NEXT8            Out of limit
0355 02A2 A820  NEXT5   A     @C32,@VSPTR      Point to 'FOR' I.D. entry
     02A4 019B'
     02A6 0292'
0356 02A8 C0E0          MOV   @VSPTR,R3        GOTO TOP OF 'FOR' LOOP
     02AA 02A6'
0357 02AC 0223          AI    R3,-8            Point to old EXTRAM
     02AE FFF8
0358 02B0 06A0          BL    @GET1            Get new EXTRAM
     02B2 025A'
0359 02B4 C801          MOV   R1,@EXTRAM       Put it in
     02B6 0202'
0360 02B8 05C3          INCT R3                POINT AT OLD PGMPTR
0361 02BA 06A0          BL    @GET1            Get old PGMPTR
     02BC 0282'
0362 02BE C801          MOV   R1,@PGMPTR       Put it in
     02C0 020C'
0363 02C2 06A0          BL    @PGMCHR          Get 1st token in line
     02C4 0214'
0364 02C6 0460  NEXT8   B     @CONT            Continue on
     02C8 01A6'
0365 02CA
0366            *       TEST LIMIT FOR DECREMENT
0367 02CA
0368 02CA 0A14  NEXT6   SLA   R4,1             Check if out of limit
0369 02CC 15EA          JGT   NEXT5            If within limit - continue
0370 02CE 10FB          JMP   NEXT8            Continue PARSE
0371                    END
NO ERRORS,   0008 WARNINGS
```

LABEL		VALUE	DEFN	REFERENCES							
[illegible]		0200		0157	0220	0258					
[illegible]	R	[illegible]	0061	0217	0344						
[illegible]		[illegible]	0201	0109							
[illegible]		[illegible]	0200	0124							
[illegible]	R	[illegible]	0060	0114	0336						
[illegible]		[illegible]	0230	0215	0224	0245					
[illegible]	R	027A	0058	0306	0342						
[illegible]		[illegible]	0258	0166	0191	0195	0236	0355			
C4	R	01F4	0058	0264	0290						
C40	R	008A'	0058	0185							
CB	D	006C'	0157	0055	0218	0338					
CBH63		009F'	0176	0183	0198						
CBH67	R	0246'	0058	0136	0172	0329					
CBH6A		00AF'	0182	0144							
CONT	R	0208'	0063	0237	0263	0364					
CX10		0001	0003	0004	0128						
EDLINE	R	0176'	0057	0247							
EDSTKT	R	0219	0057	0187	0284	0298					
ED$		008E	0073	0110							
ERR	R	[illegible]	0064	0266	0311						
[illegible]		00F4	0202	0177							
[illegible]		00FE'	0203	0104	0111	0178	0190	0255			
ERRFN		014E	0264	0292							
ERRFNN		014E'	0265	0262	0270	0289	0337				
ERRMUV	R	00FE'	0059	0201							
ERRSYN	R	0100'	0062	0203							
ERRT	R	00F4'	0062	0200							
EXTRAM	R	0286'	0060	0211	0264	0290	0291	0293	0359		
EXTRM1	R		0060								
FAC	R	0240'	0059	0107	0140	0196	0204	0210	0261	0331	
FAC2	R	0244'	0059	0113	0172	0183	0198	0329			
FAC6	R	0028'	0059	0114							
FLTONE	R	0108'	0058	0205							
FOR$		008C	0068	0271							
GET1	R	02BC'	0061	0125	0260	0335	0358	0361			
GETV	R	000C'	0061	0106							
GETV1	R	029A'	0061	0159	0232	0351					
GOTO90	R	00FC'	0064	0202							
LN$		0009	0075	0274							
MOVFAC	R	026A'	0061	0223	0339						
NEXT$		0096	0069	0250							
NEXT2		0236'	0326	0307							
NEXT2A		022A'	0310	0327							
NEXT2B		0226'	0308	0330							
NEXT4		021E'	0306	0332							
NEXT5		02A2'	0355	0348	0369						
NEXT6		02CA'	0368	0352							
NEXT8		02C6'	0364	0354	0370						
NFOR	D	0000'	0102	0055							
NFOR03		0162'	0236	0228	0240						
NFOR05		016C'	0239	0233							
NFOR07		0170'	0246	0235							
NFOR09		0174'	0247	0286	0299						
NFOR1		0006'	0105	0103							
NFOR10		017E'	0250	0300							
NFOR11		0186'	0269	0251							
NFOR12		01E0'	0283	0253	0278						
NFOR13		01EC'	0288	0248							
NFOR1A		002E'	0122	0143	0147						
NFOR1B		0050'	0144	0137							

```
%FOR1C  0060'  0152  0141
%FOR1D  007E'  0156  0154
%FOR1E  0084'  0159  0123  0145
%FOR1F  008E   0159  0164
 FOR2   0112   1204  0186
%FOR20  0104   0274  0272  0283
 FOR3   0110'  0209  0199
%FOR30  011E   0277  0275
NNEXT   D 0232'  0324  0055
P359    0000   0003  0003
PARSE   R 00DA'  0063  0193
PGMCHR  R 0204'  0063  0173  0179  0192  0249  0254  0279  0283  0294  0297
                       0363
PGMPT1  R 0210'  0060  0296
PGMPTR  R 0200'  0060  0212  0276  0281  0293  0295  0362
PRGFLG  R 01EE'  0060  0288
PSHPRS  R 00A0'  0063  0174  0180
PUTV1   R 0074'  0063  0160
R0      0000           0126  0140  0196  0204  0205  0206  0207  0208  0265
                       0288  0310
R1      0001           0108  0126  0133  0136  0144  0210  0211  0212  0213
                       0261  0336  0359  0362
R10     000A           0296
R1L3    R        0064
R2      0002           0152  0153  0163
R3      0003           0119  0122  0142  0146  0153  0155  0158  0161  0229
                       0231  0246  0252  0257  0259  0273  0333  0334  0349
                       0350  0356  0357  0360
R4      0004           0155  0156  0162  0227  0234  0239  0331  0347  0353
                       0368
R8      0008           0102  0102  0110  0177  0189  0250  0269  0271  0274
                       0277  0280  0281  0282  0295
SADD    R 0272'  0062  0341
SAVREG  R 026E'  0062  0340
SCOMPB  R 028A'  0062  0225  0346
SETREG  R 0276'  0062  0342
SMB     R 001E'  0061  0112
SSEP$   0082     0066
STEP$   00B2     0072  0181  0189
STLN    R 01FC'  0057  0291
STRIN$  00C7     0074  0277
STVSPT  R 023A'  0064  0122  0326
SUB$    00A1     0070  0269
SYM     R 0234'  0061  0105  0256  0324
TO$     00B1     0071  0175  0177
TREM$   0083     0067  0194
VDPRD   R 003C'  0057  0133
VERMAC  M        A0001 0003
VERS    0000     0003  0004  0128
VPOP    R 0240'  0061  0216  0222  0328
VPOP20  R        0062
VPUSH   R 022B'  0061  0169  0170  0171  0186  0209  0214  0308
VSPTR   R 024A'  0060  0119  0152  0166  0185  0191  0195  0215  0218  0224
                       0229  0236  0257  0306  0326  0333  0338  0343  0345
                       0349  0355  0356
```