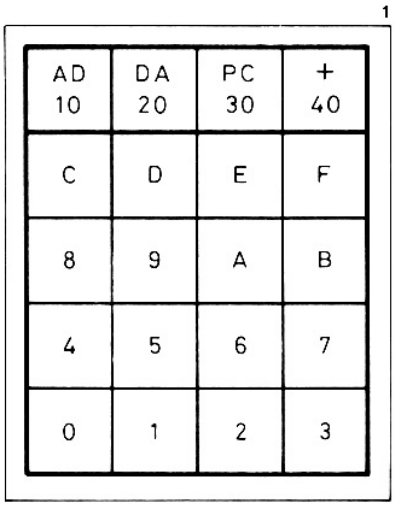
Mini-assembler

Small, but brave, short and powerful

M. Dohmen
R. Koekoek

Are you annoyed time and again by having to calculate branch offsets for branch instructions? We are not anymore, because the “Mini-assembler” described here makes programming the KIM considerably easier in many respects. The program itself requires only 254 bytes of RAM/ROM and will therefore be a welcome aid to many KIM users when writing other programs.

Fig. 1 Keyboard layout for the Mini-assembler.



Operation

What does this Mini-assembler do? Labels can be placed easily, without losing programming space. Once placed, they can be looked up easily later. The address and the corresponding label then appear on the display. Labels can be used with all branch instructions and all 3-byte instructions (thus also, for example, with LDA ABS).

This works as follows. Once you have entered the program, set the NMI vector to 0200 (the starting address of the Mini-assembler). To place a label, make sure that the address where the label is to be placed is shown on the display. A label name is a combination of digits. The KIM keyboard is therefore used differently in the program, as shown in Fig. 1. A total of 80 decimal labels can consequently be used, represented by the numbers 00 through 4F hex. The way in which the labels are stored in memory is shown in Fig. 2.

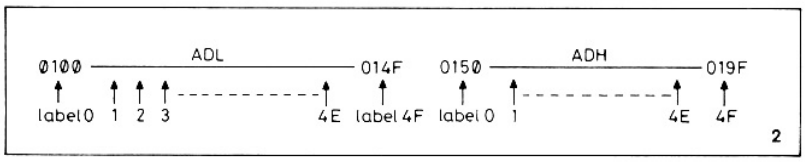


Fig. 2 Overview of how the labels are stored in memory.

To program labels 00 through 0F hex, press the “ST” key and then a key between 0 and F hex corresponding to the label you want to place (label 02 is: “ST”, “2”). When you release that key you are back in the monitor program and can continue entering the program. The labels are divided into groups: 00 through 0F is label group 0, 10 through 1F is label group 1, etc.
The example mentioned above therefore concerned label group 0. To program other label groups, proceed as follows.
Label group 1: “ST”, “AD” (short), “0” (through F).
Label group 2: “ST”, “DA” (short), “0” (through F).
Label group 3: “ST”, “PC” (short), “0” (through F).
Label group 4: “ST”, “+” (short), “0” (through F).
The “AD”, “DA”, “PC” and “+” keys therefore indicate the label group. These keys have another function, which will be discussed later. If you want to use these keys to indicate a label group, they must be pressed briefly. As with label group 0, after pressing a digit key you return to the monitor and can continue entering the program. If you continue holding the key down, the label number appears on the data display as a check. A second function is “find label”. If you suspect that you have already used a label, or no longer remember where a particular label is located, you can find it by pressing “ST”, then briefly “GO”, and then the label name. After these operations you are back in the monitor program and the display shows the address belonging to the label just searched for. If you now want to continue entering from the address where you left off, simply press the “PC” key. If you know the address where you placed a label but do not know which label it is, enter the relevant address while still in the monitor and then press “ST”. The address display remains at the address you entered. The data display shows FF if no label was placed at that address, and the label name if a label was placed there. If you then want to continue entering a program, to enter Data mode you must hold “DA” down until the label number shown on the data display is replaced by the actual data. To enter Address mode, do all of the above but use the “AD” key instead of “DA”. The same applies to “PC” and “+”. After this, everything starts again from the beginning. An overview of all functions can be found in Fig. 3.

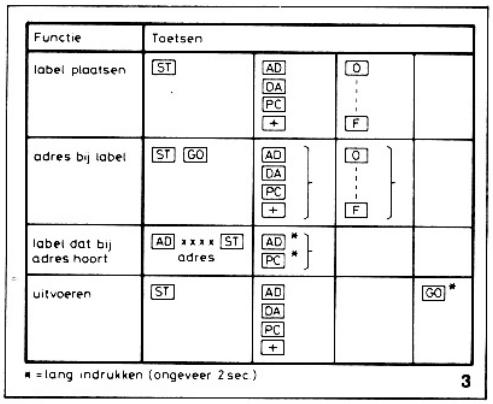


Fig. 3 Overview of the functions to be used.

Running the program

Once you have entered the program being developed, including the labels, you can replace them per label group. For example, if you have used label group 0, put label 00 at the starting address and label 0F at the ending address ("AD", start address, "ST", 0, "AD", end address, "ST", F). Now enter the starting address again after "AD" and then press "ST". Next, hold the "GO" key down until the end address appears on the address display and the starting label appears on the data display. When this has happened, all branch instructions and all 3-byte instructions should point to the correct addresses.

Indicating labels in a program

All relative branch instructions end in 10000 binary. When you do not want to use a label, enter a relative branch in the normal way, for example:

20 1F 1F

E6 00

DO F9

F9 is now not a label, but an actual offset. If you want to refer to labels, change X0 to X2. The second byte of the (now no longer existing) instruction gives the label number. In the example this becomes:

"AD", 0300,

"ST", 1 (label 01)

20 1F 1F

E6 00

D2 01 (label 01)

Labels can also be used with 3-byte instructions. The first byte remains unchanged. The second byte is replaced by the label number and the third byte is always F2, provided that page F2 is not used for RAM/ROM. In our example there was a 3-byte instruction. We will replace this with a 3-byte instruction using a label. Address 1F1F hex must receive label 02; the rest remains the same.

“AD”, 1F1F, “ST”, 2,
“AD”, 0300, “ST”, 1,
20 02 F2
E6 00
D2 01

To illustrate what has been discussed, the program “Checksum” from Listing 1 is entered step by step using labels. Label group 1 is used.

“AD”, 0300, “ST”, “AD”, 0:
label l0 at starting address
“DA”, A2 + FE +
86 + F6 +
86 + F7 +
“ST” “AD” 1:
label 11 at address 0306
BD + 12 + F2 +:
label 12 at address 01FF
20 + 13 + F2 +:
label 13 at address 1F91
CA +
D2 + 11 +
4C + 14 + F2 +:
label 14 at address 1C22

Now label 12 must be placed at address 01FF, label 13 at 1F91, and label 14 at address 1C22.

“AD”, 01FF, “ST”, “AD”, 2
“AD”, 1F91, “ST”, “AD”, 3
“AD”, 1C22, “ST”, “AD”, 4

Now all labels have been placed and only the end and start addresses have to be specified.

“AD”, 0312, “ST”, “AD”, F
“AD”, 0300, “ST”, “AD”, 0

The labels can now be replaced. Press “ST” and then “AD”, because label group 1 must be converted. Next, hold the “GO” key down until the end address appears. If everything has gone correctly, all labels have been replaced and the program works. After running this program, address 00F6 should contain 81 hex and address 00F7 should contain F7 hex.

The program “Checksum” is a kind of addition program that adds together all bytes of the “Mini-assembler” program and places the sum at addresses 00F6 and 00F7 hex. It is therefore a check that the “Mini-assembler” has been entered correctly. This little program uses a routine from the KIM monitor, so it does not work directly on other systems, but the same also applies to the “Mini-assembler” itself.

Multiple label groups

If a program contains more labels than one label group can hold, all start labels (indicating the starting address) and all end labels (indicating the ending address) must be the same in order to convert all label groups at once. If you keep the label groups separate for the various parts of a program (for example, for subroutines), you only need to replace one label group, leaving the other groups untouched.

It is possible to place multiple labels at one address. When searching for a particular label, the highest label is always shown on the data display. One advantage of this program is that it is not always necessary to refer to labels. Those people who have made calculating branch offsets their hobby can therefore continue doing so. When you want to develop a new program, you can use the program from Listing 1 (see Part 1) to erase the labels.

Relative branches

X2 is used instead of X0 for relative branches. This was done first of all to distinguish it from X0, but also to prevent a program from going wrong if you forget to convert the labels. After all, the microprocessor does not recognize the instruction XXX10010 and will therefore stop. Only a reset can get the microprocessor running again. The 3-byte instructions end in F2. With branch instructions, the microprocessor therefore jumps to page F2. If no memory is present there, it encounters instruction F2, which also satisfies the bit pattern XXX10010.

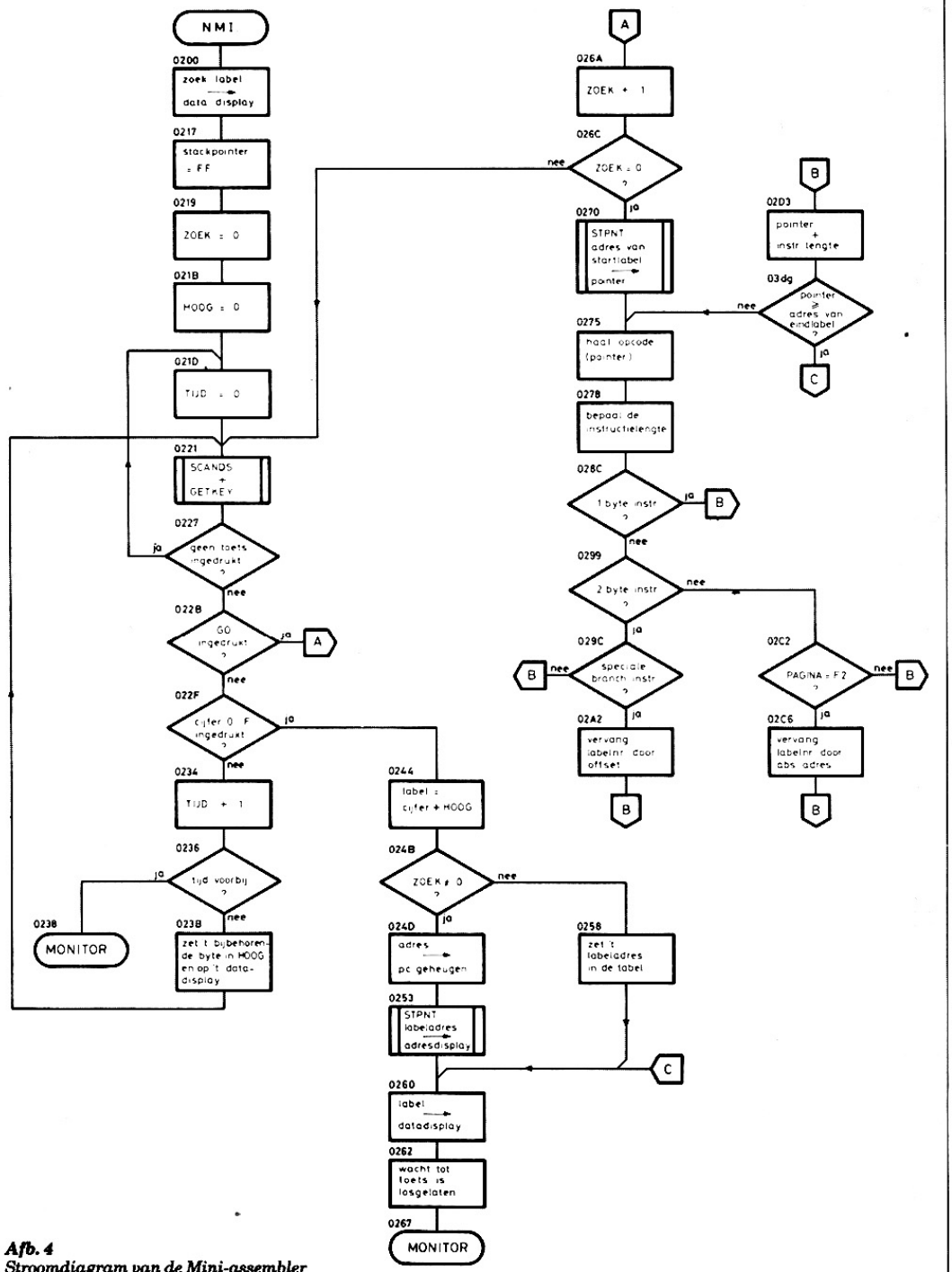
Program

When the “ST” key is pressed, an NMI is generated. The microprocessor therefore jumps from the monitor to 0200 hex, where the stack pointer is set correctly and the label corresponding to the address visible on the address display is searched for (see Fig. 4). The timer for the “GO” key is set to 0, as is the memory location “High”, which is used to remember the label group. As long as no key is pressed, “Time” remains at zero. When “AD”, “DA”, “PC” or “+” is pressed, “Time” starts running and the corresponding label group is placed in “High”. If the key is held down for a long time (“Time” becomes zero again), the program returns to the monitor. The corresponding code is now interpreted and executed as a monitor function. If a digit is pressed, its value is added to “High”, which thus forms the final label. If the program is in search mode, the address shown on the display is placed in the monitor’s PC memory (00EF, 00F0 hex). The address belonging to the label just entered is placed on the display. As soon as the last key is released, the program returns to the monitor. If the program is not in search mode, the address belonging to the label is placed in the table. When the “GO” key is pressed, “Search” is incremented and, as long as “Search” is not 00 again, the program returns to the keyboard routine. If the “GO” key is pressed only briefly, “Search” will not be zero and the program is in search mode. If the “GO” key is held down for a long time, then at some point (after approximately 1 s) “Search” becomes zero and the program begins replacing the labels. The address of the starting label is placed in the pointer. The

opcode of the first instruction is then fetched. The instruction length is subsequently determined. One-byte instructions are skipped. With 2-byte instructions it looks for the following bit pattern: XXX10010. If the instruction satisfies this, the last four bits are made 0 and the second byte of the instruction is replaced by the offset. With 3-byte instructions it first looks at the last byte. If this is F2, it replaces the last two bytes with the absolute address. The pointer is then advanced to the next instruction. If the pointer passes the end address, the program returns to the monitor as soon as the “GO” key is released. Finally, a few tips.

Now that you can work with labels, you will probably also want to remove or insert instructions. This is very easy if all relative branch instructions and all 3-byte instructions have been replaced by labels. You then use a “normal” shifting program and, when shifting, briefly check whether the address concerned represents a label (page 01 in KIM RAM). If it contains a label, the address of the label must be replaced by the new address.

It is often useful to put labels on cassette. This can be done by saving the memory block from address 0100 through 019F hex.



Afb. 4
Stroomdiagram van de Mini-assembler

Listing 1 Program for erasing the labels and the example program “Checksum”.

Listing 2 Program of the Mini-assembler

Mini-assembler

Lijst 1

```
          MICRO-WARE ASSEMBLER 65XX-1.0 PAGE 01

0010:
0020:
0030:                 MINI-ASSEMBLER
0040:
0050:                 M. DOHMEN
0060:                 R. KOEKOEK
0070:
0080:                 MICROPROCESSOR  6502
0090:
0100:
0110: 0200                    ORG   $0200
0120:
0130:                 ZERO-PAGE ADRESSEN
0140:                 VAN DE KIM
0150:
0160: 0200            PCL     *     $00EF
0170: 0200            PCH     *     $00F0
0180: 0200            INL     *     $00F8
0190: 0200            INH     *     $00F9
0200: 0200            POINTL  *     $00FA
0210: 0200            POINTH  *     $00FB
0220: 0200            HOOG    *     $00FD
0230: 0200            ZOEK    *     $00FE
0240:
0250:                 TIJDELIJKE ADRESSEN
0260:
0270: 0200            TABA    *     $0100
0280: 0200            ENDAD   *     $010F
0290: 0200            TABB    *     $0150
0300: 0200            POINTD  *     $015F
0310: 0200            TABD    *     $02E9
0320:
0330:                 SUBROUTINES UIT DE
0340:                 KIM-MONITOR
0350:
0360: 0200            TTYKB   *     $1C22
0370: 0200            MONA    *     $1C83
0380: 0200            INCPT   *     $1F63
0390: 0200            GETKEY  *     $1F6A
0400: 0200            SCANDS  *     $1F1F
0410:
0420:
0430:                 HOOFDPROGRAMMA
0440:
0450:
0460: 0200 A2 4F      START   LDXIM $4F
0470: 0202 A5 FA      LOOPA   LDAZ  POINTL  VERGELIJK ADRESPOINTER
0480: 0204 DD 00 01           CMPAX TABA    MET DE TABEL, WAARBIJ
0490: 0207 D0 07              BNE   SEA     X ALS INDEX DIENT
0500: 0209 A5 FB              LDAZ  POINTH
0510: 020B DD 50 01           CMPAX TABB
0520: 020E F0 03              BEQ   SKIP
0530: 0210 CA         SEA     DEX
0540: 0211 10 EF              BPL   LOOPA
0550: 0213 86 F9      SKIP    STXZ  INH     LABEL NAAR DATADISPLAY
0560: 0215 A2 FF              LDXIM $FF
0570: 0217 9A                 TXS           STACKPOINTER IS FF
0580: 0218 E8                 INX           X IS 0
0590: 0219 86 FE              STXZ  ZOEK    ZOEK IS 0
0600: 021B 86 FD              STXZ  HOOG    HOOG IS 0
0610: 021D A9 00      LOOPB   LDAIM #00
0620: 021F 85 F8              STAZ  INL     TIJD IS 0
0630: 0221 20 1F 1F   SHOW    JSR   SCANDS
0640: 0224 20 6A 1F           JSR   GETKEY
0650: 0227 C9 15              CMPIM $15     TOETS INGEDRUKT?
0660: 0229 F0 F2              BEQ   LOOPB   NEEN, TERUG
0670: 022B C9 13              CMPIM $13     GO-TOETS?
0680: 022D F0 3B              BEQ   GO      JA, VOER UIT
0690: 022F C9 10              CMPIM $10     CIJFER?
0700: 0231 90 11              BCC   HEX     JA, VERDER
0710: 0233 A8                 TAY           AD/DA/PC OF +
0720: 0234 E6 F8              INCZ  INL     HOOG TIJD OP
0730: 0236 D0 03              BNE   SKIPO   WACHT 1 SEC
0740: 0238 4C 83 1C           JMP   MONA     TERUG NAAR MONITOR
0750: 023B BE E9 02   SKIPO   LDXAY TABD    HAAL 16-TAL UIT TABEL
0760: 023E 86 F9              STXZ  INH     NAAR DATADISPLAY
0770: 0240 86 FD              STXZ  HOOG    EN IN HOOG
0780: 0242 D0 DD              BNE   SHOW
0790: 0244 05 FD      HEX     ORAZ  HOOG    BEPAAL LABELNUMMER
0800: 0246 AA                 TAX           IS HOOG + LAATSTE CIJFER
0810: 0247 A5 FA              LDAZ  POINTL
0820: 0249 A4 FE              LDYZ  ZOEK    ZOEKMODE?
0830: 024B F0 0B              BEQ   STORE   NEEN, VERDER
0840: 024D 85 EF              STAZ  PCL     JA, ZET ADRES
0850: 024F A5 FB              LDAZ  POINTH  IN PCL EN PCH
0860: 0251 85 F0              STAZ  PCH
0870: 0253 20 EE 02           JSR   STPNT   LABELADRES NAAR DISPLAY
0880: 0256 90 08              BCC   END
0890: 0258 9D 00 01   STORE   STAAX TABA    ZET ADRES IN
0900: 025B A5 FB              LDAZ  POINTH  DE TABEL
0910: 025D 9D 50 01           STAAX TABB
0920: 0260 86 F9      END     STXZ  INH     LABEL NAAR DATADISPLAY
0930: 0262 20 1F 1F   LOOPC   JSR   SCANDS  WACHT, TOT TOETS
0940: 0265 D0 FB              BNE   LOOPC   WORDT LOSGELATEN
0950: 0267 4C 77 1C           JMP   TTYKB   TERUG NAAR MONITOR
0960: 026A E6 FE      GO      INCZ  ZOEK    HOOG ZOEK OP
0970: 026C D0 B3              BNE   SHOW    1 SEC VOORBIJ?
0980: 026E A6 FD              LDXZ  HOOG    HAAL LABELGROEP
0990: 0270 20 EE 02           JSR   STPNT   STARTADRES NAAR POINTER
1000: 0273 A0 00      GTBYTE  LDYIM $00
1010: 0275 B1 FA              LDAIY POINTL  HAAL OPCODE
1020: 0277 A8                 TAY           BEWAAR OPCODE
1030: 0278 A2 00              LDXIM $00     LENGTE VAN
1040: 027A C9 20              CMPIM $20     DE INSTRUCTIE
1050: 027C F0 16              BEQ   THREE
1060: 027E 29 9F              ANDIM $9F
1070: 0280 F0 51              BEQ   NEXT    1-BYTE INSTRUCTIE,X=0
1080: 0282 29 1D              ANDIM $1D
1090: 0284 C9 19              CMPIM $19
1100: 0286 F0 0C              BEQ   THREE
1110: 0288 29 0D              ANDIM $0D
1120: 028A C9 08              CMPIM #08
1130: 028C F0 45              BEQ   NEXT    1-BYTE INSTRUCTIE,X=0
1140: 028E 29 0C              ANDIM $0C
1150: 0290 C9 0C              CMPIM #0C
1160: 0292 F0 01              BNE   TWO
1170: 0294 E8         THREE   INX           3-BYTES INSTRUCTIE,X=2
1180: 0295 E8         TWO     INX           2-BYTES INSTRUCTIE,X=1
1190: 0296 98                 TYA           HAAL OPCODE
1200: 0297 A0 01              LDYIM $01
1210: 0299 CA                 DEX
1220: 029A D0 20              BNE   RDBYTE  2-BYTES INSTRUCTIE?
1230: 029C 29 1F              ANDIM $1F     SPECIALE
1240: 029E 49 12              EORIM $12     SPRONG-
1250: 02A0 D0 16              BNE   LOOPD   INSTRUCTIE?
1260: 02A2 B1 FA              LDAIY POINTL  HAAL TWEEDE BYTE
1270: 02A4 AA                 TAX           (=LABEL)
1280: 02A5 BC 00 01           LDYAX TABA    ZET
1290: 02A8 88                 DEY           DE
1300: 02A9 98                 TYA           LABEL
1310: 02AA 18                 CLC           OM
1320: 02AB E5 FA              SBCZ  POINTL  IN
1330: 02AD A0 01              LDYIM $01     EEN
1340: 02AF 91 FA              STAIY POINTL  OFFSET      *
1350: 02B1 88                 DEY           VERANDER
1360: 02B2 B1 FA              LDAIY POINTL  DE OPCODE
1370: 02B4 29 F0              ANDIM $F0     TWEEDE NIBBLE IS 0
1380: 02B6 91 FA              STAIY POINTL
1390: 02B8 A2 01      LOOPD   LDXIM $01
1400: 02BA D0 17              BNE   NEXT
1410: 02BC B1 FA      RDBYTE  LDAIY POINTL  HAAL TWEEDE BYTE
1420: 02BE AA                 TAX
1430: 02BF C8                 INY
1440: 02C0 B1 FA              LDAIY POINTL  HAAL DERDE BYTE
1450: 02C2 C9 F2              CMPIM #F2     LAATSTE BYTE IS F2?
1460: 02C4 D0 0B              BNE   SKIPTH
1470: 02C6 BD 50 01           LDAAX TABB    ZET HET ADRES
1480: 02C9 91 FA              STAIY POINTL  UIT DE TABEL
1490: 02CB 88                 DEY           ACHTER DE OPCODE
1500: 02CC BD 00 01           LDAAX TABA
1510: 02CF 91 FA              STAIY POINTL
1520: 02D1 A2 02      SKIPTH  LDXIM $02
1530: 02D3 20 63 1F   NEXT    JSR   INCPT   VERHOOG DE POINTER
1540: 02D6 CA                 DEX           MET X + 1
1550: 02D7 10 FA              BPL   NEXT
1560: 02D9 A6 FD              LDXZ  HOOG    HAAL LABELGROEP
1570: 02DB A5 FB              LDAZ  POINTH  VERGELIJK
1580: 02DD DD 5F 01           CMPAX POINTD  DE POINTER
1590: 02E0 90 91              BCC   GTBYTE  MET HET
1600: 02E2 D0 07              BNE   LOOPE   EINDADRES
1610: 02E4 A5 FA              LDAZ  POINTL
1620: 02E6 DD 0F 01           CMPAX ENDAD
1630: 02E9 90 88              BCC   GTBYTE
1640: 02EB 4C 60 02   LOOPE   JMP   END
1650: 02EE BD 00 01   STPNT   LDAAX TABA
1660: 02F1 85 FA              STAZ  POINTL
1670: 02F3 BD 50 01           LDAAX TABB
1680: 02F6 85 FB              STAZ  POINTH
1690: 02F8 60                 RTS
1700: 02F9 10                 =     $10
1710: 02FA 20                 =     $20
1720: 02FB 40                 =     $40
1730: 02FC 50                 =     $50
1740: 02FD 30                 =     $30
```

Lijst 2

```
1180: 0300                    ORG   $0300
1190:
1200:
1210: 0300            CHKHI   *     $00F6
1220: 0300            CHKSUM  *     $00F7
1230: 0300            CHECK   *     $01FF
1240: 0300            MONIT   *     $1C22
1250: 0300            CHK     *     $1F91
1260:
1270:
1280:                 PROGRAMMA OM LABELS
1290:                 TE WISSEN
1300:
1310: 0300 A9 FF      WIS     LDAIM $FF
1320: 0302 A2 00              LDXIM $00
1330: 0304 48         LOOPU   PHA
1340: 0305 CA                 DEX
1350: 0306 D0 FC              BNE   LOOPU
1360: 0308 4C 22 1C           JMP   MONIT
1370:
1380:
1390: 0300                    ORG   $0300
1400:
1410:                 VOORBEELDPROGRAMMA OM DE
1420:                 CHECKSUM TE BEREKENEN VAN
1430:                 DE MINI-ASSEMBLER
1440:
1450: 0300 A2 FE      CSUM    LDXIM $FE
1460: 0302 86 F6              STX   CHKHI
1470: 0304 86 F7              STX   CHKSUM
1480: 0306 BD FF 01   LOOPF   LDAAX CHECK
1490: 0309 20 91 1F           JSR   CHK
1500: 030C CA                 DEX
1510: 030D D0 F7              BNE   LOOPF
1520: 030F 4C 22 1C           JMP   MONIT
```