tion's operand, determines whether it is legal for use with an instruction of that category and then stores the machine-language code for that instruction in memory locations 997, 998 and 999. Those locations are not used unless a second cassette recorder is called upon. If you use a second recorder, you must find another safe location; page 0, locations 0, 1 and 2 are all right if no calls to a user defined machine-language function (USR(X)) are needed.

Once the instruction has been found, the operand has been interpreted and the numbers are stored in locations 997 to 999, the assembler branches to line 1000. The length of the machine-language code for that line of assembly language is stored in the variable BY, which determines the locations to get written to tape.

Lines 1030 to 1050 are necessary for earlier PETs, which did not leave enough space between records when writing data files to tape. These lines can be omitted on later machines, though they do no harm when included. They just give a blip of current to the tape recorder motor to leave a little gap on the tape.

Using the assembler

Once the assembler has been typed in, saved on tape (!) and debugged, you just type "RUN" to start. The beginning address can put the machine language code wherever it does not interfere with other programs that call upon it; the extremely short LOADER program to get it in could be almost squeezed into one line if space is critical. To assemble, just type in the assembly language in standard format, one line at a time, and conclude it with "END". All jumps must be to literal, explicit addresses and all "immediate" instructions must say what constant is to be loaded – no symbols allowed. Branches are calculated by the program; for them, the absolute address of the target statement is input.

Upon receiving the input line, the program translates it and displays the address and the (decimal) opcodes generated. After END, type in or load the LOADER, rewind the data tape, and load it in. It's that simple.

Program Listing continued

```
216 POKE 997,OP+8:BY=1:GOTO 1000

220 IF LEFT$(C$,1)<> "#" GOTO 230

221 REM HANDLE "IMMEDIATE" INSTRUCTIONS HERE

222 POKE 998,VAL(RIGHT$(C$,L-1)):BY=2

224 IF CA=1 THEN POKE 997,OP+8:GOTO 1000

226 IF CA=4 OR CA=5 THEN POKE 997,OP:GOTO 1000

227 REM CATEGORIES 1,4,5 ARE ONLY ONES ALLOWED HERE

228 PRINT "NO":GOTO 100

230 IF LEFT$(C$,1)<>"(" GOTO 260

231 REM CHECK FOR VARIOUS INDIRECT INSTRUCTIONS

232 IF RIGHT$(C$,3)<>"),Y" GOTO 240

233 REM IT IS AN "(INDIRECT),Y"

234 POKE 998,VAL(MID$(C$,2,L-4)):BY=2

236 IF CA=1 THEN POKE 997,OP+16:GOTO 1000

237 REM IF NOT CATEGORY 1, ERROR

238 PRINT "NO":GOTO 100

240 IF RIGHT$(C$,3)<>",X)" GOTO 250

242 POKE 998,VAL(MID$(C$,2,L-4)):BY=2

243 REM IT MUST BE CATEGORY 1, ELSE ERROR

244 IF CA=1 THEN POKE 997,OP:GOTO 1000

246 PRINT "NO":GOTO 100

250 IF RIGHT$(C$,1)<>")" THEN PRINT "NO":GOTO 100

251 REM IT BETTER BE A JMP (INDIRECT), ELSE ERROR

252 IF CA<>6 THEN PRINT "NO": GOTO 100

254 N=VAL(MID$(C$,2,L-2)):HI=INT(N/256):
      BY=3

255 REM HI=HIGH PART OF ADDRESS

256 POKE 999,HI:POKE 998,N-256*HI

258 POKE 997,OP+32:GOTO 1000

260 IF RIGHT$(C$,2)<>",X" GOTO 280

262 N=VAL(LEFT$(C$,L-2))

264 IF N> 255 GOTO 270

265 REM HANDLE "ZERO PAGE,X" HERE

266 POKE 998,N:BY=2:IF CA=2 THEN POKE 997,OP+16:
      GOTO 1000

268 IF CA=1 OR CA=3 OR CA=5 THEN POKE 997,OP+20:
      GOTO 1000

269 PRINT "NO":GOTO 100

270 HI=INT(N/256):BY=3

272 POKE 999,HI:POKE 998,N-256*HI

274 IF CA=2 THEN POKE 997,OP+24:GOTO 1000

276 IF CA=1 OR CA=3 OR CA=5 THEN POKE 997,OP+28:
      GOTO 1000

278 PRINT "NO":GOTO 100

280 IF RIGHT$(C$,2)<>",Y" GOTO 300

282 N=VAL(LEFT$(C$,L-2))

284 IF N>255 GOTO 290

285 REM HANDLE ZERO PAGE,Y HERE

286 POKE 998,N:BY=2

287 IF CA=2 THEN POKE 997,OP+16:GOTO 1000

288 IF CA=5 THEN POKE 997,OP+20:GOTO 1000

289 REM CONTINUE HERE!--SOME ZERO PAGE,Y MUST BE
      TREATED AS ABSOLUTE,Y

290 HI=INT(N/256):POKE 999,HI:POKE 998,N-256*HI:
      BY=3

292 IF CA=1 THEN POKE 997,OP+24:GOTO 1000
```