

```
1  program X1_ALGOL_60_compiler(input,output,lib_tape);

2  const d2 =      4;
3         d3 =      8;
4         d4 =     16;
5         d5 =     32;
6         d6 =     64;
7         d7 =    128;
8         d8 =    256;
9         d10 =   1024;
10        d12 =   4096;
11        d13 =   8192;
12        d15 =  32768;
13        d16 =  65536;
14        d17 = 131072;
15        d18 = 262144;
16        d19 = 524288;
17        d20 =1048576;
18        d21 =2097152;
19        d22 =4194304;
20        d23 =8388608;
21        d24 =16777216;
22        d25 =33554432;
23        d26 =67108864;
24        mz  =134217727;

25        gvc0 =    138;  {0-04-10}
26        tlib =    800;  {0-25-00}
27        plie =   6783;  {6-19-31}
28        bim  =    930;  {0-29-02}
29        nlscop =    31;
30        nlsc0 =    48;
31        mlib =    800;  {0-25-00}
32        klie =  10165;  {9-29-21}
33        crfb =    623;  {0-19-15}
34        mcpb =    928;  {0-29-00}

35  var  tlsc,plib,flib,klib,nlib,
36        rht,vht,qc,scan,rfsb,rnsa,rnsb,rnsc,rnsd,
37        dl,inw,fnw,dflag,bflag,oflag,
38        nflag,kflag,
39        iflag,mflag,vflag,aflag,sflag,eflag,jflag,pflag,fflag,
40        bn,vlam,pnlv,gvc,lv,oh,id,nid,ibd,
41        inba,fora,forc,psta,pstb,spe,
42        arra,arrb,arrc,arrd,ic,aic,rlaa,rlab,qa,qb,
```

```

43     rlsc,flsc,klsc,nlsc: integer;
44     bitcount,bitstock: integer;
45     store: array[0..12287] of integer;
46     rns_state: (ps,ms,virginal);
47     rfs_case,nas_stock,pos: integer;
48     word_del_table: array[10..38] of integer;
49     flex_table: array[0..127] of integer;
50     opc_table: array[0..112] of integer;

51     rlib,mcpe: integer;

52     lib_tape: text;

53     ii: integer;

54 procedure stop(n: integer);
55 {emulation of a machine instruction}
56 begin writeln(output);
57     writeln(output,'*** stop ',n div d5:1,'-',n mod d5:2,' ***');
58     halt
59 end {stop};

60 function read_flexowriter_symbol: integer;           {LK}
61 label 1,2;
62 var s,fts: integer;
63 begin
64     1: read(input,s);
65     if rfsb = 0
66     then if (s = 62 {tab}) or (s = 16 {space}) or (s = 26 {crlf})
67     then goto 2
68     else if (s = 122 {lc}) or (s = 124 {uc}) or (s = 0 {blank})
69     then begin rfsb:= s {new flexowriter shift}; goto 1 end
70     else if s = 127 {erase} then goto 1
71     else stop(19) {flexowriter shift undefined};
72     2: fts:= flex_table[s];
73     if fts > 0
74     then if rfsb = 124
75     then {uppercase} read_flexowriter_symbol:= fts div d8
76     else {lowercase} read_flexowriter_symbol:= fts mod d8
77     else if fts = -0 then stop(20) {wrong parity}
78     else if fts = -1 then stop(21) {undefined punching}
79     else if s = 127 {erase} then goto 1
80     else begin rfsb:= s {new flexowriter shift}; goto 1 end
81 end {read_flexowriter_symbol};

```



```

127         sym:= read_flexowriter_symbol;
128         goto 1
129     end
130     else sym:= 97 {comment}
131 else begin sym:= read_flexowriter_symbol;
132     if sym = 163 {_}
133     then begin repeat sym:=
134         read_flexowriter_symbol
135     until sym <> 163;
136     if (sym > 9) and (sym <= 32)
137     then if sym = 29 {t}
138     then begin sym:=
139         read_flexowriter_symbol;
140         if sym = 163 {_}
141         then begin repeat
142             sym:=
143                 read_flexowriter_symbol
144         until sym <> 163;
145         if sym = 14 {e}
146         then sym:= 94 {step}
147         else sym:= 113 {string}
148         end
149         else stop(12)
150         end
151         else begin wdt2:=
152             word_del_table[sym] div 128;
153             if wdt2 = 0
154             then sym:= wdt1 + 64
155             else sym:= wdt2
156             end
157         else stop(13)
158         end
159         else stop(12)
160     end;
161     repeat nas_stock:= - read_flexowriter_symbol;
162     if nas_stock = - 163 {_}
163     then repeat nas_stock:= read_flexowriter_symbol
164     until nas_stock <> 163
165     until nas_stock <= 0
166     end {word delimiter}
167     else if sym = 70 {>} then sym:= 71 {>=}
168     else if sym = 72 {=} then sym:= 80 {eqv}
169     else if sym = 74 {<} then sym:= 73 {<=}
170     else if sym = 76 {~} then sym:= 79 {imp}
171     else if sym = 124 {:} then sym:= 68 {div}

```

```

172         else stop(13)
173         end
174         else stop(14) {? or " or '}
175     end;
176     next_ALGOL_symbol:= sym
177 end {next_ALGOL_symbol};

178 procedure read_next_symbol;                                {ZY}
179 label 1;
180 begin
181 1: case rns_state of
182   ps: begin dl:= next_ALGOL_symbol;
183         {store symbol in symbol store;}
184         if rnsa > d7
185         then begin rnsa:= rnsa div d7;
186                 store[rnsb]:= store[rnsb] + dl * rnsa
187                 end
188         else begin rnsa:= d15; rnsb:= rnsb + 1; store[rnsb]:= dl * rnsa;
189                 if rnsb + 8 > plib then stop(25)
190                 end
191         end;
192   ms: begin {take symbol from symbol store;}
193         dl:= (store[rnsd] div rnc) mod d7;
194         if rnc > d7
195         then rnc:= rnc div d7
196         else begin rnc:= d15; rnsd:= rnsd + 1 end
197         end;
198   virginal:
199     begin qc:= 0; rfs_case:= 0; nas_stock:= 1;
200     if scan > 0 {prescan}
201     then begin rns_state:= ps;
202             {initialize symbol store;}
203             rnsb:= bim + 8; rnsd:= bim + 8; rnsa:= d22; rnc:= d15;
204             store[rnsb]:= 0;
205             end
206     else rns_state:= ms;
207     goto 1
208     end
209   end {case}
210 end {read_next_symbol};

211 procedure read_until_next_delimiter;                        {FT}
212 label 1,3,4,5;
213 var marker,elsc,bexp: integer;

```

```

214     function test1: boolean;
215     begin if dl = 88 {.}
216         then begin dflag:= 1;
217             read_next_symbol; test1:= test1
218         end
219         else if dl = 89 {ten} then goto 1
220         else test1:= dl > 9
221     end {test1};

222     function test2: boolean;
223     begin if dl = 89 {ten} then inw:= 1; test2:= test1
224     end {test2};

225     function test3: boolean;
226     begin read_next_symbol; test3:= test1
227     end {test3};

228     begin {body of read_until_next_delimiter}
229         read_next_symbol;
230         nflag:= 1;
231         if (dl > 9) and (dl < 63) {letter}
232         then begin dflag:= 0; kflag:= 0; inw:= 0;
233             repeat fnw:= (inw mod d6) * d21; inw:= inw div d6 + dl * d21;
234                 read_next_symbol
235             until (inw mod d3 > 0) or (dl > 62);
236             if inw mod d3 > 0
237             then begin dflag:= 1;
238                 fnw:= fnw + d23; marker:= 0;
239                 while (marker = 0) and (dl < 63) do
240                     begin marker:= fnw mod d6 * d21; fnw:= fnw div 64 + dl * d21;
241                         read_next_symbol
242                     end;
243                     while marker = 0 do
244                         begin marker:= fnw mod d6 * d21;
245                             fnw:= fnw div d6 + 63 * d21
246                         end;
247                         while dl < 62 do read_next_symbol
248                             end;
249                             goto 4;
250                     end;
251                 kflag:= 1; fnw:= 0; inw:= 0; dflag:= 0; elsc:= 0;
252                 if test2 {not (dl in [0..9,88,89])}
253                 then begin nflag:= 0;
254                     if (dl = 116 {true}) or (dl = 117 {false})
255                     then begin inw:= dl - 116;

```

```

256             dflag:= 0; kflag:= 1; nflag:= 1;
257             read_next_symbol;
258             goto 4
259         end;
260     goto 5
261 end;
262 repeat if fnw < d22
263     then begin inw:= 10 * inw + dl;
264             fnw:= 10 * fnw + inw div d26;
265             inw:= inw mod d26;
266             elsc:= elsc - dflag
267         end
268     else elsc:= elsc - dflag + 1
269 until test3;
270 if (dflag = 0) and (fnw = 0)
271 then goto 4;
272 goto 3;
273 1: if test3 {not (dl in [0..9,88,89])}
274     then if dl = 64 {plus}
275         then begin read_next_symbol; dflag:= dl end
276         else begin read_next_symbol; dflag:= - dl - 1 end
277     else dflag:= dl;
278     while not test3 {dl in [0..9,88,89]} do
279     begin if dflag >= 0
280         then dflag:= 10 * dflag + dl
281         else dflag:= 10 * dflag - dl + 9;
282         if abs(dflag) >= d26 then stop(3)
283     end;
284     if dflag < 0 then dflag:= dflag + 1;
285     elsc:= elsc + dflag;
286 3: {float}
287     if (inw = 0) and (fnw = 0)
288     then begin dflag:= 0; goto 4 end;
289     bexp:= 2100 {2**11 + 52; P9-characteristic};
290     while fnw < d25 do
291     begin inw:= 2 * inw; fnw:= 2 * fnw + inw div d26; inw:= inw mod d26;
292         bexp:= bexp - 1
293     end;
294     if elsc > 0
295     then repeat fnw:= 5 * fnw; inw:= (fnw mod 8) * d23 + (5 * inw) div 8;
296             fnw:= fnw div 8;
297             if fnw < d25
298             then begin inw:= 2 * inw; fnw:= 2 * fnw + inw div d26;
299                     inw:= inw mod d26;
300                     bexp:= bexp - 1

```

```

301         end;
302         bexp:= bexp + 4; elsc:= elsc - 1;
303         until elsc = 0
304     else if elsc < 0
305     then repeat if fnw >= 5 * d23
306         then begin inw:= inw div 2 + (fnw mod 2) * d25;
307             fnw:= fnw div 2; bexp:= bexp + 1
308         end;
309         inw:= 8 * inw; fnw:= 8 * fnw + inw div d26;
310         inw:= inw mod d26 + fnw mod 5 * d26;
311         fnw:= fnw div 5; inw:= inw div 5;
312         bexp:= bexp - 4; elsc:= elsc + 1
313     until elsc = 0;
314     inw:= inw + 2048;
315     if inw >= d26
316     then begin inw:= 0; fnw:= fnw + 1;
317         if fnw = d26 then begin fnw:= d25; bexp:= bexp + 1 end
318     end;
319     if (bexp < 0) or (bexp > 4095) then stop(4);
320     inw:= (inw div 4096) * 4096 + bexp;
321     dflag:= 1;
322 4: oflag:= 0;
323 5:
324 end {read_until_next_delimiter};

325 procedure fill_t_list(n: integer);
326 begin store[tlsc]:= n; tlsc:= tlsc + 1
327 end {fill_t_list};

328 procedure prescan;                                     {HK}

329     label 1,2,3,4,5,6,7;
330     var bc,mbc: integer;

331     procedure fill_prescan_list(n: integer); {n = 0 or n = 1}   {HF}
332     var i,j,k: integer;
333     begin {update plib and prescan_list chain:}
334         k:= plib; plib:= k - dflag - 1; j:= k;
335         for i:= 2*bc + n downto 1 do
336             begin k:= store[j]; store[j]:= k - dflag - 1; j:= k end;
337         {shift lower part of prescan_list down over dfag + 1 places:}
338         k:= plib;
339         if dflag = 0
340         then for i:= j - plib downto 1 do

```

```

341         begin store[k]:= store[k+1]; k:= k + 1 end
342     else begin {shift;}
343         for i:= j - plib - 1 downto 1 do
344             begin store[k]:= store[k+2]; k:= k + 1 end;
345             {enter fnw in prescan_list;}
346             store[k+1]:= fnw
347         end;
348         {enter inw in prescan_list;}
349         store[k]:= inw
350     end {fill_prescan_list};

351     procedure augment_prescan_list;                                {HH}
352     begin dflag:= 1; inw:= plie; fnw:= plie - 1;
353         fill_prescan_list(0)
354     end {augment_prescan_list};

355     procedure block_introduction;                                {HK}
356     begin fill_t_list(bc); fill_t_list(-1) {block-begin marker};
357         mbc:= mbc + 1; bc:= mbc;
358         augment_prescan_list
359     end {block_introduction};

360     begin {body of prescan}
361         plib:= plie; store[plie]:= plie - 1; tlsc:= tlib;
362         bc:= 0; mbc:= 0; qc:= 0; rht:= 0; vht:= 0;
363         fill_t_list(dl); {dl should be 'begin'}
364         augment_prescan_list;
365     1: bflag:= 0;
366     2: read_until_next_delimiter;
367     3: if dl <= 84 {+,-,*,/,_.,|^,>,>=,=<=<,|,~,^,',_~,_,=,goto,if,then,else}
368         then {skip;} goto 1;
369         if dl = 85 {for}
370         then begin block_introduction; goto 1 end;
371         if dl <= 89 {do,comma,period,ten} then {skip;} goto 1;
372         if dl = 90 {;} then begin fill_prescan_list(0); goto 2 end;
373         if dl = 91 {;}
374         then begin while store[tlsc-1] < 0 {block-begin marker} do
375             begin tlsc:= tlsc - 2; bc:= store[tlsc] end;
376             if rht <> 0 then stop(22); if vht <> 0 then stop(23);
377             goto 1
378         end;
379         if dl <= 97 {:=,step,until,while,comment} then {skip;} goto 1;
380         if dl <= 99 {(,)}
381         then begin if dl = 98 then rht:= rht + 1 else rht:= rht - 1;
382             goto 1

```

```

383     end;
384   if dl <= 101 {[,]}
385   then begin if dl = 100 then vht:= vht + 1 else vht:= vht - 1;
386             goto 1
387           end;
388   if dl = 102 {<}
389   then begin repeat if dl = 102 {<} then qc:= qc + 1;
390                 if dl = 103 {>} then qc:= qc - 1;
391                 if qc > 0 then read_next_symbol
392                 until qc = 0;
393             goto 2
394           end;
395   if dl = 104 {begin}
396   then begin fill_t_list(dl);
397             if bflag <> 0 then goto 1;
398             read_until_next_delimiter;
399             if (dl <= 105) or (dl > 112) then goto 3;
400             tlsc:= tlsc - 1 {remove begin from t_list};
401             block_introduction;
402             fill_t_list(104) {add begin to t_list again};
403             goto 3;
404           end;
405   if dl = 105 {end}
406   then begin while store[tlsc-1] < 0 {block-begin marker} do
407             begin tlsc:= tlsc - 2; bc:= store[tlsc] end;
408             if rht <> 0 then stop(22); if vht <> 0 then stop(23);
409             tlsc:= tlsc - 1 {remove corresponding begin from t_list};
410             if tlsc > tlib then goto 1;
411             goto 7 {end of prescan}
412           end;
413   if dl <= 105 {dl = >} then goto 1;
414   if dl = 111 {switch}
415   then if bflag = 0
416         then {declarator}
417             begin read_until_next_delimiter {for switch identifier};
418                 fill_prescan_list(0); goto 6
419             end
420         else {specifier}
421             goto 5;
422   4: if dl = 112 {procedure}
423   then if bflag = 0
424         then {declarator}
425             begin bflag:= 1;
426                 read_until_next_delimiter {for procedure identifier};
427                 fill_prescan_list(1); block_introduction; goto 6

```

```

428         end
429         else {specificier}
430         goto 5;
431     if dl > 117 {false} then stop(8);
432 5: read_until_next_delimiter;
433 6: if dl <> 91 {;} then goto 4;
434     goto 2;
435 7:
436 end {prescan};

437 procedure intro_new_block2;                                {HW}
438 label 1;
439 var i,w: integer;
440 begin inba:= d17 + d15;
441 1: i:= plib; plib:= store[i]; i:= i + 1;
442     while i <> plib do
443         begin w:= store[i];
444             if w mod 8 = 0 {at most 4 letters/digits}
445                 then i:= i + 1
446             else begin store[nlib+nlsc]:=store[i+1]; i:= i + 2; nlsc:= nlsc + 1 end;
447                 store[nlib+nlsc]:= w; nlsc:= nlsc + 2;
448                 if nlib + nlsc > i then stop(15);
449                 store[nlib+nlsc-1]:= bn * d19 + inba
450             end;
451             if inba <> d18 + d15
452                 then begin inba:= d18 + d15; goto 1 end;
453             lvc:= 0
454 end {intro_new_block2};

455 procedure intro_new_block1;                                {HW}
456 begin fill_t_list(nlsc); fill_t_list(161);
457     intro_new_block2
458 end {intro_new_block1};

459 procedure intro_new_block;                                  {HW}
460 begin bn:= bn + 1; intro_new_block1
461 end {intro_new_block};

462 procedure bit_string_maker(w: integer);                    {LL}
463 var head,tail,i: integer;
464 begin head:= 0; tail:= w mod d10;
465     {shift (head,tail) bitcount places to the left;}
466     for i:= 1 to bitcount do
467         begin head:= 2 * head + tail div d26; tail:= (tail mod d26) * 2

```

```

468     end {shift};
469     bitstock:= bitstock + tail; bitcount:= bitcount + w div d10;
470     if bitcount > 27
471     then begin bitcount:= bitcount - 27;
472             store[rnsb]:= bitstock; bitstock:= head; rnsb:= rnsb + 1;
473             if rnsb = rnsd
474             then if nlib + nlsc + 8 < plib
475                     then begin {shift text, fli, kli and nli}
476                             for i:= nlib + nlsc - rnsd - 1 downto 0 do
477                                 store[rnsd+i+8]:= store[rnsd+i];
478                                 rnsd:= rnsd + 8; flib:= flib + 8;
479                                 klib:= klib + 8; nlib:= nlib + 8
480                             end
481                             else stop(25)
482                         end
483     end {bit_string_maker};

484     procedure address_coder(a: integer);                                {LS}
485     var w: integer;
486     begin w:= a mod d5;
487         if w = 1 then w:= 2048 {2*1024 + 0} else
488         if w = 2 then w:= 3074 {3*1024 + 2} else
489         if w = 3 then w:= 3075 {3*1024 + 3}
490         else w:= 6176 {6*1024 + 32} + w;
491         bit_string_maker(w);
492         w:= (a div d5) mod d5;
493         if w = 0 then w:= 2048 {2*1024 + 0} else
494         if w = 1 then w:= 4100 {4*1024 + 4} else
495         if w = 2 then w:= 4101 {4*1024 + 5} else
496         if w = 4 then w:= 4102 {4*1024 + 6} else
497         if w = 5 then w:= 4103 {4*1024 + 7}
498         else w:= 6176 {6*1024 + 32} + w;
499         bit_string_maker(w);
500         w:= (a div d10) mod d5;
501         if w = 0 then w:= 1024 {1*1024 + 0}
502         else w:= 6176 {6*1024 + 32} + w;
503         bit_string_maker(w)
504     end {address_coder};

505     procedure fill_result_list(opc,w: integer);                        {ZF}
506     var j: 8..61;
507     begin rlsc:= rlsc + 1;
508         if opc < 8
509         then begin address_coder(w);
510                 w:= (w div d15) * d15 + opc;

```

```

511         if w = 21495808 { 2S  0 A } then w:= 3076 {3*1024 +  4} else
512         if w = 71827459 { 2B  3 A } then w:= 3077 {3*1024 +  5} else
513         if w = 88080386 { 2T 2X0 } then w:= 4108 {4*1024 + 12} else
514         if w = 71827456 { 2B  0 A } then w:= 4109 {4*1024 + 13} else
515         if w =  4718592 { 2A  0 A } then w:= 7280 {7*1024 + 112} else
516         if w = 71303170 { 2B 2X0 } then w:= 7281 {7*1024 + 113} else
517         if w = 88604673 { 2T  1 A } then w:= 7282 {7*1024 + 114} else
518         if w =         0 { 0A 0X0 } then w:= 7283 {7*1024 + 115} else
519         if w =  524291 { 0A  3 A } then w:= 7284 {7*1024 + 116} else
520         if w = 88178690 {N 2T 2X0 } then w:= 7285 {7*1024 + 117} else
521         if w = 71827457 { 2B  1 A } then w:= 7286 {7*1024 + 118} else
522         if w =  1048577 { 0A 1X0 B } then w:= 7287 {7*1024 + 119} else
523         if w = 20971522 { 2S 2X0 } then w:= 7288 {7*1024 + 120} else
524         if w =  4784128 {Y 2A  0 A } then w:= 7289 {7*1024 + 121} else
525         if w =  8388608 { 4A 0X0 } then w:= 7290 {7*1024 + 122} else
526         if w =  4390912 {Y 2A 0X0 P} then w:= 7291 {7*1024 + 123} else
527         if w = 13172736 {Y 6A  0 A } then w:= 7292 {7*1024 + 124} else
528         if w =  1572865 { 0A 1X0 C } then w:= 7293 {7*1024 + 125} else
529         if w =  524288 { 0A  0 A } then w:= 7294 {7*1024 + 126}
530         else begin address_coder(w div d15 + opc * d12);
531             w:= 7295 {7*1024 + 127}
532         end
533     end {opc < 8}
534 else if opc <= 61
535 then begin j:= opc;
536     case j of
537         8: w:= 10624 {10*1024+384};  9: w:=  6160 { 6*1024+ 16};
538         10: w:= 10625 {10*1024+385}; 11: w:= 10626 {10*1024+386};
539         12: w:= 10627 {10*1024+387}; 13: w:=  7208 { 7*1024+ 40};
540         14: w:=  6161 { 6*1024+ 17}; 15: w:= 10628 {10*1024+388};
541         16: w:=  5124 { 5*1024+  4}; 17: w:=  7209 { 7*1024+ 41};
542         18: w:=  6162 { 6*1024+ 18}; 19: w:=  7210 { 7*1024+ 42};
543         20: w:=  7211 { 7*1024+ 43}; 21: w:= 10629 {10*1024+389};
544         22: w:= 10630 {10*1024+390}; 23: w:= 10631 {10*1024+391};
545         24: w:= 10632 {10*1024+392}; 25: w:= 10633 {10*1024+393};
546         26: w:= 10634 {10*1024+394}; 27: w:= 10635 {10*1024+395};
547         28: w:= 10636 {10*1024+396}; 29: w:= 10637 {10*1024+397};
548         30: w:=  6163 { 6*1024+ 19}; 31: w:=  7212 { 7*1024+ 44};
549         32: w:= 10638 {10*1024+398}; 33: w:=  4096 { 4*1024+  0};
550         34: w:=  4097 { 4*1024+  1}; 35: w:=  7213 { 7*1024+ 45};
551         36: w:= 10639 {10*1024+399}; 37: w:= 10640 {10*1024+400};
552         38: w:= 10641 {10*1024+401}; 39: w:=  7214 { 7*1024+ 46};
553         40: w:= 10642 {10*1024+402}; 41: w:= 10643 {10*1024+403};
554         42: w:= 10644 {10*1024+404}; 43: w:= 10645 {10*1024+405};
555         44: w:= 10646 {10*1024+406}; 45: w:= 10647 {10*1024+407};

```

```

556         46: w:= 10648 {10*1024+408}; 47: w:= 10649 {10*1024+409};
557         48: w:= 10650 {10*1024+410}; 49: w:= 10651 {10*1024+411};
558         50: w:= 10652 {10*1024+412}; 51: w:= 10653 {10*1024+413};
559         52: w:= 10654 {10*1024+414}; 53: w:= 10655 {10*1024+415};
560         54: w:= 10656 {10*1024+416}; 55: w:= 10657 {10*1024+417};
561         56: w:= 5125 { 5*1024+ 5}; 57: w:= 10658 {10*1024+418};
562         58: w:= 5126 { 5*1024+ 6}; 59: w:= 10659 {10*1024+419};
563         60: w:= 10660 {10*1024+420}; 61: w:= 7215 { 7*1024+ 47}
564     end {case}
565     end {opc <= 61}
566     else if opc = 85{ST}
567     then w:= 5127 { 5*1024 + 7}
568     else w:= 10599 {10*1024 + 359} + opc;
569     bit_string_maker(w)
570 end {fill_result_list};

571 procedure main_scan;                                     {EL}

572     label 1,2,3,64,66,69,70,76,81,82,8201,8202,83,8301,84,8401,85,8501,
573           86,8601,87,8701,8702,8703,8704,8705,
574           90,91,92,94,95,96,98,9801,9802,9803,9804,99,100,101,
575           102,104,105,1052,106,107,108,1081,1082,1083,
576           109,110,1101,1102,1103,111,112,1121,1122,1123,1124;

577 procedure fill_t_list_with_delimiter;                   {ZW}
578 begin fill_t_list(d8*oh+dl)
579 end {fill_t_list_with_delimiter};

580 procedure fill_future_list(place,value: integer);       {FU}
581 var i: integer;
582 begin if place >= klib
583     then begin if nlib + nlsc + 16 >= plib then stop(6);
584             for i:= nlib + nlsc - 1 downto klib do
585                 store[i+16]:= store[i];
586                 klib:= klib + 16; nlib:= nlib + 16
587             end;
588             store[place]:= value
589         end {fill_future_list};

590 procedure fill_constant_list(n: integer);               {KU}
591 var i: integer;
592 begin if klib + klsc = nlib
593     then begin if nlib + nlsc + 16 >= plib then stop(18);
594             for i:= nlib + nlsc - 1 downto nlib do

```

```

595         store[i+16]:= store[i];
596         nlib:= nlib + 16
597     end;
598     if n >= 0
599     then store[klib+klsc]:= n
600     else {one's complement representation} store[klib+klsc]:= mz + n;
601         klsc:= klsc + 1
602     end {fill_constant_list};

603     procedure unload_t_list_element(var variable: integer);           {ZU}
604     begin tlsc:= tlsc - 1; variable:= store[tlsc]
605     end {unload_t_list_element};

606     procedure fill_output(c: integer);
607     begin pos:= pos + 1;
608         if c < 10 then write(chr(c+ord('0')))
609         else if c < 36 then write(chr(c-10+ord('a')))
610         else if c < 64 then write(chr(c-37+ord('A')))
611         else if c = 184 then write(' ')
612         else if c = 138
613             then begin write(' ':8 - (pos - 1) mod 8);
614                     pos:= pos + 8 - (pos - 1) mod 8
615                 end
616         else begin writeln; pos:= 0 end
617     end {fill_output};

618     procedure offer_character_to typewriter(c: integer);               {HS}
619     begin c:= c mod 64;
620         if c < 63 then fill_output(c)
621     end {offer_character_to typewriter};

622     procedure label_declaration;                                       {FY}
623     var id,id2,i,w: integer;
624     begin id:= store[nlib+nid];
625         if (id div d15) mod 2 = 0
626         then begin {preceding applied occurrences}
627                 fill_future_list(flib+id mod d15,rlsc)
628             end
629         else {first occurrence}
630             store[nlib+nid]:= id - d15 + 1 * d24 + rlsc;
631             id:= store[nlib+nid-1];
632             if id mod d3 = 0
633             then begin {at most 4 letters/digits}
634                     i:= 4; id:= id div d3;
635                     while (id mod d6) = 0{void} do

```

```

636         begin i:= i - 1; id:= id div d6 end;
637         repeat offer_character_to_typewriter(id);
638             i:= i - 1; id:= id div d6
639         until i = 0
640     end
641 else begin id2:= store[nlib+nid-2];
642         id2:= id2 div d3 + (id2 mod d3) * d24;
643         w:= (id2 mod d24) * d3 + id div d24;
644         id:= (id mod d24) * d3 + id2 div d24;
645         id2:= w;
646         i:= 9;
647         repeat offer_character_to_typewriter(id);
648             i:= i - 1;
649             w:= id2 div d6 + (id mod d6) * d21;
650             id:= id div d6 + (id2 mod d6) * d21;
651             id2:= w
652         until i = 0
653     end;
654     fill_output(138{TAB});
655     w:= rlsc;
656     for i:= 1 to 3 do
657         begin offer_character_to_typewriter(w div d10 div 10);
658             offer_character_to_typewriter(w div d10 mod 10);
659             w:= (w mod d10) * d5;
660             if i < 3 then fill_output(184{SPACE})
661         end;
662         fill_output(139{NLCR})
663     end {label_declaration};

664 procedure test_first_occurrence;                                {LF}
665 begin id:= store[nlib+nid];
666     if (id div d15) mod 2 = 1 {first occurrence}
667     then begin id:= id - d15 - id mod d15 + 2 * d24 + flsc;
668             if nid <= nlsc0 {MCP}
669             then fill_future_list(flib+flsc,store[nlib+nid]);
670                 store[nlib+nid]:= id;
671                 flsc:= flsc + 1
672             end
673     end {test_first_occurrence};

674 procedure new_block_by_declaration1;                            {HU}
675 begin fill_result_list(0,71827456+bn) {2B 'bn' A};
676     fill_result_list(89{SCC},0);
677     pnlv:= 5 * 32 + bn; vlam:= pnlv
678 end {new_block_by_declaration1};

```

```

679  procedure new_block_by_declaration;                                {HU}
680  begin if store[tlsc-2] <> 161{block-begin marker}
681    then begin tlsc:= tlsc - 1 {remove 'begin'};
682            fill_result_list(0,4718592) {2A 0 A};
683            fill_result_list(1,71827456+rlsc+3) {2B 'rlsc+3' A};
684            fill_result_list(9{ETMP},0);
685            fill_result_list(2,88080384+flsc) {2T 'flsc'};
686            fill_t_list(flsc); flsc:= flsc + 1;
687            intro_new_block;
688            fill_t_list(104{begin});
689            new_block_by_declaration1
690    end
691  end {new_block_by_declaration};

692  procedure fill_name_list;                                         {HN}
693  begin nlsc:= nlsc + dflag + 2;
694    if nlsc + nlib > plib then stop(16);
695    store[nlib+nlsc-1]:= id; store[nlib+nlsc-2]:= inw;
696    if inw mod d3 > 0 then store[nlib+nlsc-3]:= fnw
697  end {fill_name_list};

698  procedure reservation_of_local_variables;                         {KY}
699  begin if lvc > 0
700    then begin fill_result_list(0,4718592+lvc) {2A 'lvc' A};
701            fill_result_list(0,8388657) {4A 17X1};
702            fill_result_list(0,8388658) {4A 18X1}
703    end
704  end {reservation_of_local_variables};

705  procedure address_to_register;                                    {ZR}
706  begin if id div d15 mod 2 = 0 {static addressing}
707    then if id div d24 mod d2 = 2 {future list}
708      then fill_result_list(2,
709        71303168+id mod d15{2B 'FLI-address'})
710      else fill_result_list(id div d24 mod 4,
711        71827456+id mod d15{2B 'static address' A})
712    else fill_result_list(0,
713        21495808+id mod d15{2S 'dynamic address' A})
714  end {address_to_register};

715  procedure generate_address;                                       {ZH}
716  var opc: integer;
717  begin address_to_register;
718    if (id div d16) mod 2 = 1

```

```

719     then {formal} fill_result_list(18{TFA},0)
720     else begin opc:= 14{TRAD};
721         if (id div d15) mod 2 = 0 then opc:= opc + 1{TRAS};
722         if (id div d19) mod 2 = 1 then opc:= opc + 2{TIAD or TIAS};
723         fill_result_list(opc,0)
724     end
725 end {generate_address};

726 procedure reservation_of_arrays;                                {KN}
727 begin if vlam <> 0
728     then begin vlam:= 0;
729         if store[tlsc-1] = 161{block-begin marker}
730         then rlaa:= nlib + store[tlsc-2]
731         else rlaa:= nlib + store[tlsc-3];
732         rlab:= nlib + nlsc;
733         while rlab <> rlaa do
734             begin id:= store[rlab-1];
735                 if (id >= d26) and (id < d25 + d26)
736                 then begin {value array:}
737                     address_to_register;
738                     if (id div d19) mod 2 = 0
739                     then fill_result_list(92{RVA},0)
740                     else fill_result_list(93{IVA},0);
741                     store[rlab-1]:= (id div d15) * d15 - d16 + pnlv;
742                     pnlv:= pnlv + 8 * 32 {at most 5 indices}
743                 end;
744                 if store[rlab-2] mod d3 = 0
745                 then rlab:= rlab - 2 else rlab:= rlab - 3
746             end;
747             rlab:= nlib + nlsc;
748             while rlab <> rlaa do
749                 begin if store[rlab-1] >= d26
750                     then begin id:= store[rlab-1] - d26;
751                         if id < d25
752                         then begin address_to_register;
753                             fill_result_list(95{VAP},0)
754                         end
755                         else begin id:= id - d25;
756                             address_to_register;
757                             fill_result_list(94{LAP},0)
758                         end
759                     end;
760                 if store[rlab-2] mod d3 = 0
761                 then rlab:= rlab - 2 else rlab:= rlab - 3
762             end;

```

```

763         if nflag <> 0
764             then id:= store[nlib+nid]
765         end
766 end {reservation_of_arrays};

767 procedure procedure_statement;                                {LH}
768 begin if eflag = 0 then reservation_of_arrays;
769     if nid > nlscof
770     then begin if fflag = 0 then test_first_occurrence;
771         address_to_register
772     end
773     else begin fill_t_list(store[nlib+nid] mod d12);
774         if dl = 98{()}
775         then begin eflag:= 1; goto 9801 end
776     end
777 end {procedure_statement};

778 procedure production_transmark;                                {ZL}
779 begin fill_result_list(9+2*fflag-eflag,0)
780 end {production_transmark};

781 procedure production_of_object_program(opht: integer);        {ZS}
782 var operator,block_number: integer;
783 begin oh:= opht;
784     if nflag <> 0
785     then begin nflag:= 0; aflag:= 0;
786         if pflag = 0
787         then if jflag = 0
788             then begin address_to_register;
789                 if oh > (store[tlsc-1] div d8) mod 16
790                 then operator:= 315{5*63}
791                 else begin operator:= store[tlsc-1] mod d8;
792                     if (operator <= 63) or (operator > 67)
793                     then operator:= 315{5*63}
794                     else begin tlsc:= tlsc - 1;
795                         operator:= 5 * operator
796                     end
797                 end;
798             if fflag = 0
799             then begin if id div d15 mod 2 = 0
800                 then operator:= operator + 1;
801                 if id div d19 mod 2 <> 0
802                 then operator:= operator + 2;
803                 fill_result_list(operator-284,0)
804             end

```

```

805         else fill_result_list(operator-280,0)
806         end
807     else if fflag = 0
808         then begin block_number:= id div d19 mod d5;
809             if block_number <> bn
810             then begin fill_result_list
811                 (0,71827456+block_number);
812                 fill_result_list(28{GTA},0)
813             end;
814             test_first_occurrence;
815             if id div d24 mod 4 = 2
816             then fill_result_list(2,88080384+id mod d15)
817                 {2T 'address'}
818             else fill_result_list(1,88604672+id mod d15)
819                 {2T 'address' A}
820             end
821         else begin address_to_register;
822             fill_result_list(35{TFR},0)
823         end
824     else begin procedure_statement;
825         if nid > nlsco
826         then begin fill_result_list(0,4718592{2A 0 A});
827             production_transmark
828         end
829     end
830 end
831 else if aflag <> 0
832 then begin aflag:= 0; fill_result_list(58{TAR},0) end;
833 while oh <= store[tlsc-1] div d8 mod 16 do
834 begin tlsc:= tlsc - 1; operator:= store[tlsc] mod d8;
835     if (operator > 63) and (operator<= 80)
836     then fill_result_list(operator-5,0)
837     else if operator = 132 {NEG}
838     then fill_result_list(57{NEG},0)
839     else if (operator < 132) and (operator > 127)
840     then begin {ST,STA,STP,STAP}
841         if operator > 129
842         then begin {STP,STAP}
843             tlsc:= tlsc - 1;
844             fill_result_list(0,71827456+store[tlsc]{2B 'BN' A})
845         end;
846         fill_result_list(operator-43,0)
847     end
848     else {special function}
849     if (operator > 127) and (operator <= 141)

```

```

850     then fill_result_list(operator-57,0)
851     else if (operator > 141) and (operator <= 151)
852     then fill_result_list(operator-40,0)
853     else stop(22)
854     end
855 end {production_of_object_program};

856 function thenelse: boolean;                                {ZN}
857 begin if (store[tlsc-1] mod 255 = 83{then})
858     or (store[tlsc-1] mod 255 = 84{else})
859     then begin tlsc:= tlsc - 2;
860         fill_future_list(flib+store[tlsc],rlsc);
861         unload_t_list_element(eflag);
862         thenelse:= true
863     end
864     else thenelse:= false
865 end {thenelse};

866 procedure empty_t_list_through_thenelse;                    {FR}
867 begin oflag:= 1;
868     repeat production_of_object_program(1)
869     until not thenelse
870 end {empty_t_list_through_thenelse};

871 function do_in_t_list: boolean;                              {ER}
872 begin if store[tlsc-1] mod 255 = 86
873     then begin tlsc:= tlsc - 5;
874         nlsc:= store[tlsc+2]; bn:= bn - 1;
875         fill_future_list(flib+store[tlsc+1],rlsc+1);
876         fill_result_list(1,88604672{2T OX0 A}+store[tlsc]);
877         do_in_t_list:= true
878     end
879     else do_in_t_list:= false
880 end {do_in_t_list};

881 procedure look_for_name;                                     {HZ}
882 label 1,2;
883 var i,w: integer;
884 begin i:= nlib + nlsc;
885 1: w:= store[i-2];
886     if w = inw
887     then if w mod 8 = 0
888         then {at most 4 letters/digits} goto 2
889         else {more than 4 letters/digits}
890             if store[i-3] = fnw then goto 2;

```

```

891     if w mod 8 = 0 then i:= i - 2 else i:= i - 3;
892     if i > nlib then goto 1;
893     stop(7);
894 2: nid:= i - nlib - 1; id:= store[i-1];
895     pflag:= id div d18 mod 2;
896     jflag:= id div d17 mod 2;
897     fflag:= id div d16 mod 2
898 end {look_for_name};

899 procedure look_for_constant;                                {FW}
900 var i: integer;
901 begin if klib + klsc + dflag >= nlib
902     then begin {move name list}
903         if nlib + nlsc + 16 >= plib then stop(5);
904         for i:= nlsc - 1 downto 0 do
905             store[nlib+i+16]:= store[nlib+i];
906         nlib:= nlib + 16
907         end;
908     if dflag = 0
909     then begin {search integer constant}
910         store[klib+klsc]:= inw;
911         i:= 0;
912         while store[klib+i] <> inw do i:= i + 1
913         end
914     else begin {search floating constant}
915         store[klib+klsc]:= fnw; store[klib+klsc+1]:= inw;
916         i:= 0;
917         while (store[klib+i] <> fnw)
918             or (store[klib+i+1] <> inw) do i:= i + 1
919         end;
920     if i = klsc
921     then {first occurrence} klsc:= klsc + dflag + 1;
922     id:= 3 * d24 + i;
923     if dflag = 0 then id:= id + d19;
924     jflag:= 0; pflag:= 0; fflag:= 0
925 end {look_for_constant};

926 begin {body of main scan}                                    {EL}
927 1: read_until_next_delimiter;
928 2: if nflag <> 0
929     then if kflag = 0
930         then look_for_name
931         else look_for_constant
932     else begin jflag:= 0; pflag:= 0; fflag:= 0 end;
933 3: if dl <= 65 then goto 64; {+,-}                            {EH}

```

```

934     if dl <= 68 then goto 66; {*,/,_:=}
935     if dl <= 69 then goto 69; {|^}
936     if dl <= 75 then goto 70; {<,_<=,_>,|=}
937     if dl <= 80 then goto 76; {~,^,'=>,_=}
```

938	case dl of	
939	81: goto 81; {goto}	{KR}
940	82: goto 82; {if}	{EY}
941	83: goto 83; {then}	{EN}
942	84: goto 84; {else}	{FZ}
943	85: goto 85; {for}	{FE}
944	86: goto 86; {do}	{FL}
945	87: goto 87; {,}	{EK}
946	90: goto 90; {:}	{FN}
947	91: goto 91; {;}	{FS}
948	92: goto 92; {:=}	{EZ}
949	94: goto 94; {step}	{FH}
950	95: goto 95; {until}	{FK}
951	96: goto 96; {while}	{FF}
952	98: goto 98; {(}	{EW}
953	99: goto 99; {)}	{EU}
954	100: goto 100; {[}	{EE}
955	101: goto 101; {]}	{EF}
956	102: goto 102; { <}	{KS}
957	104: goto 104; {begin}	{LZ}
958	105: goto 105; {end}	{FS}
959	106: goto 106; {own}	{KH}
960	107: goto 107; {Boolean}	{KZ}
961	108: goto 108; {integer}	{KZ}
962	109: goto 109; {real}	{KE}
963	110: goto 110; {array}	{KF}
964	111: goto 111; {switch}	{HE}
965	112: goto 112; {procedure}	{HY}
966	end {case};	

```

967 64: {+,-} {ES}
968     if oflag = 0
969     then begin production_of_object_program(9);
970             fill_t_list_with_delimiter
971             end
972     else if dl = 65{-}
973     then begin oh:= 10; dl:= 132{NEG};
974             fill_t_list_with_delimiter
975             end;
976     goto 1;
```

```

977 66: {*,/,_:=} {ET}
978     production_of_object_program(10);
979     fill_t_list_with_delimiter;
980     goto 1;

981 69: {|^} {KT}
982     production_of_object_program(11);
983     fill_t_list_with_delimiter;
984     goto 1;

985 70: {<,_<,_>,|=} {KK}
986     oflag:= 1;
987     production_of_object_program(8);
988     fill_t_list_with_delimiter;
989     goto 1;

990 76: {~,^,‘,=>,_= } {KL}
991     if dl = 76{~}
992     then begin oh:= 83-dl; goto 8202 end;
993     production_of_object_program(83-dl);
994     fill_t_list_with_delimiter;
995     goto 1;

996 81: {goto} {KR}
997     reservation_of_arrays; goto 1;

998 82: {if} {EY}
999     if eflag = 0 then reservation_of_arrays;
1000     fill_t_list(eflag); eflag:= 1;
1001 8201: oh:= 0;
1002 8202: fill_t_list_with_delimiter;
1003     oflag:= 1; goto 1;

1004 83: {then} {EN}
1005     repeat production_of_object_program(1) until not thenelse;
1006     tlsc:= tlsc - 1; eflag:= store[tlsc-1];
1007     fill_result_list(30{CAC},0);
1008     fill_result_list(2,88178688+flsc) {N 2T 'flsc'};
1009 8301: fill_t_list(flsc); flsc:= flsc + 1;
1010     goto 8201;

1011 84: {else} {FZ}
1012     production_of_object_program(1);
1013     if store[tlsc-1] mod d8 = 84{else}
1014     then if thenelse then goto 84;

```

```

1015 8401: if do_in_t_list then goto 8401;
1016     if store[tlsc-1] = 161 {block-begin marker}
1017     then begin tlsc:= tlsc - 3;
1018             nlsc:= store[tlsc+1];
1019             fill_future_list(flib+store[tlsc],rlsc+1);
1020             fill_result_list(12{RET},0);
1021             bn:= bn - 1; goto 8401
1022     end;
1023     fill_result_list(2,88080384+flsc) {2T 'flsc'};
1024     if thenelse {finds 'then'!}
1025     then tlsc:= tlsc + 1 {keep eflag in t_list};
1026     goto 8301;

1027 85:  {for}                                     {FE}
1028     reservation_of_arrays;
1029     fill_result_list(2,88080384+flsc) {2T 'flsc'};
1030     fora:= flsc; flsc:= flsc + 1;
1031     fill_t_list(rlsc);
1032     vflag:= 1; bn:= bn + 1;
1033 8501: oh:= 0; fill_t_list_with_delimiter;
1034     goto 1;

1035 86:  {do}                                     {FL}
1036     empty_t_list_through_thenelse;
1037     goto 8701; {execute part of DDEL ,}
1038 8601: {returned from DDEL ,}
1039     vflag:= 0; tlsc:= tlsc - 1;
1040     fill_result_list(2,20971520+flsc) {2S 'flsc'};
1041     fill_t_list(flsc); flsc:= flsc + 1;
1042     fill_result_list(27{FOR8},0);
1043     fill_future_list(flib+fora,rlsc);
1044     fill_result_list(19{FOR0},0);
1045     fill_result_list(1,88604672{2T 0X0 A}+store[tlsc-2]);
1046     fill_future_list(flib+forc,rlsc);
1047     eflag:= 0; intro_new_block1;
1048     goto 8501;

1049 87:  {,}                                     {EK}
1050     oflag:= 1;
1051     if iflag = 1
1052     then begin {subscript separator;}
1053             repeat production_of_object_program(1)
1054             until not thenelse;
1055             goto 1
1056     end;

```

```

1057     if vflag = 0 then goto 8702;
1058     {for-list separator;}
1059     repeat production_of_object_program(1)
1060     until not thenelse;
1061 8701: if store[tlsc-1] mod d8 = 85{for}
1062     then fill_result_list(21{for2},0)
1063     else begin tlsc:= tlsc - 1;
1064             if store[tlsc] mod d8 = 96{while}
1065             then fill_result_list(23{for4},0)
1066             else fill_result_list(26{for7},0)
1067             end;
1068     if dl = 86{do} then goto 8601;
1069     goto 1;
1070 8702: if mflag = 0 then goto 8705;
1071     {actual parameter separator;}
1072     if store[tlsc-1] mod d8 = 87{,}
1073     then if aflag = 0
1074         then if (store[tlsc-2] = rlsc)
1075             and (fflag = 0) and (jflag = 0) and (nflag = 1)
1076             then begin if nid > nlscop
1077                 then begin if (pflag = 1) and (fflag = 0)
1078                     then {non-formal procedure;}
1079                     test_first_occurrence;
1080                     {PORD construction;}
1081                     if (id div d15) mod 2 = 0
1082                     then begin {static addressing}
1083                         pstb:= ((id div d24) mod d2) * d24
1084                             + id mod d15;
1085                         if (id div d24) mod d2 = 2
1086                         then pstb:= pstb + d17
1087                         end
1088                     else begin{dynamic addressing}
1089                         pstb:= d16 + (id mod d5) * d22
1090                             + (id div d5) mod d10;
1091                         if (id div d16) mod 2 = 1
1092                         then begin store[tlsc-2]:= pstb + d17;
1093                             goto 8704
1094                         end
1095                     end;
1096                     if (id div d18) mod 2 = 1
1097                     then store[tlsc-2]:= pstb + d20
1098                     else if (id div d19) mod 2 = 1
1099                     then store[tlsc-2]:= pstb + d19
1100                     else store[tlsc-2]:= pstb;
1101                     goto 8704

```

```

1102             end
1103             else begin fill_result_list(98{TFP},0);
1104                 goto 8703
1105             end
1106         end
1107         else goto 8703
1108     else begin {completion of implicit subroutine;}
1109         store[tlsc-2]:= store[tlsc-2] + d19 + d20 + d24;
1110         fill_result_list(13{EIS},0); goto 8704
1111     end;
1112 8703: {completion of implicit subroutine;}
1113     repeat production_of_object_program(1)
1114     until not (thenelse or do_in_t_list);
1115     store[tlsc-2]:= store[tlsc-2] + d20 + d24;
1116     fill_result_list(13{EIS},0);
1117 8704: if dl = 87{,} then goto 9804 {prepare next parameter};
1118     {production of PORs:}
1119     psta:= 0; unload_t_list_element(pstb);
1120     while pstb mod d8 = 87{,} do
1121     begin psta:= psta + 1; unload_t_list_element(pstb);
1122         if pstb div d16 mod 2 = 0
1123         then fill_result_list(pstb div d24, pstb mod d24)
1124         else fill_result_list(0,pstb);
1125         unload_t_list_element(pstb)
1126     end;
1127     tlsc:= tlsc - 1;
1128     fill_future_list(flib+store[tlsc],rlsc);
1129     fill_result_list(0,4718592+psta) {2A 'psta' A};
1130     bn:= bn - 1;
1131     unload_t_list_element(fflag); unload_t_list_element(eflag);
1132     production_transmark;
1133     aflag:= 0;
1134     unload_t_list_element(mflag); unload_t_list_element(vflag);
1135     unload_t_list_element(iflag); goto 1;
1136 8705: empty_t_list_through_thenelse;
1137     if sflag = 0 then {array declaration} goto 1;
1138     {switch declaration;}
1139     oh:= 0; dl:= 160;
1140     fill_t_list(rlsc); fill_t_list_with_delimiter; goto 1;

1141 90: {:}
1142     if jflag = 0
1143     then begin {array declaration}
1144         ic:= ic + 1;
1145         empty_t_list_through_thenelse

```

{FN}

```

1146         end
1147     else begin {label declaration}
1148         reservation_of_arrays;
1149         label_declaration
1150     end;
1151     goto 1;

1152 91: goto 105{end};

1153 92: {:=}                                     {EZ}
1154     reservation_of_arrays;
1155     dl:= 128{ST}; oflag:= 1;
1156     if vflag = 0
1157     then begin if sflag = 0
1158         then begin {assignment statement}
1159             if eflag = 0
1160             then eflag:= 1
1161             else dl:= 129{STA};
1162             oh:= 2;
1163             if pflag = 0
1164             then begin {assignment to variable}
1165                 if nflag <> 0
1166                 then {assignment to scalar} generate_address;
1167             end
1168             else begin {assignment to function identifier}
1169                 dl:= dl + 2{STP or STAP};
1170                 fill_t_list((id div d19) mod d5{bn from id})
1171             end;
1172             fill_t_list_with_delimiter
1173         end
1174         else begin {switch declaration}
1175             fill_result_list(2,88080384+flsc) {2T 'flsc'};
1176             fill_t_list(flsc); flsc:= flsc + 1;
1177             fill_t_list(nid);
1178             oh:= 0; fill_t_list_with_delimiter;
1179             dl:= 160;
1180             fill_t_list(rlsc); fill_t_list_with_delimiter
1181         end
1182     end
1183     else begin {for statement}
1184         eflag:= 1;
1185         if nflag <> 0 then {simple variable} generate_address;
1186         fill_result_list(20{FOR1},0);
1187         forc:= flsc;
1188         fill_result_list(2,88080384+flsc) {2T 'flsc'};

```

```

1189         flsc:= flsc + 1;
1190         fill_future_list(flib+fora,rlsc);
1191         fill_result_list(0,4718592{2A 0 A});
1192         fora:= flsc;
1193         fill_result_list(2,71303168+flsc) {2B 'flsc};
1194         flsc:= flsc + 1;
1195         fill_result_list(9{ETMP},0)
1196     end;
1197     goto 1;

1198 94: {step}                                     {FH}
1199     empty_t_list_through_thenelse;
1200     fill_result_list(24{FOR5},0);
1201     goto 1;

1202 95: {until}                                    {FK}
1203     empty_t_list_through_thenelse;
1204     fill_result_list(25{FOR6},0);
1205     goto 8501;

1206 96: {while}                                    {FF}
1207     empty_t_list_through_thenelse;
1208     fill_result_list(22{FOR3},0);
1209     goto 8501;

1210 98:  {()                                       {EW}
1211         oflag:= 1;
1212         if pflag = 1 then goto 9803;
1213 9801: {parenthesis in expression:}
1214         fill_t_list(mflag);
1215         mflag:= 0;
1216 9802: oh:= 0; fill_t_list_with_delimiter;
1217         goto 1;
1218 9803: {begin of parameter list:}
1219         procedure_statement;
1220         fill_result_list(2,88080384+flsc) {2T 'flsc'};
1221         fill_t_list(iflag); fill_t_list(vflag);
1222         fill_t_list(mflag); fill_t_list(eflag);
1223         fill_t_list(fflag); fill_t_list(flsc);
1224         iflag:= 0; vflag:= 0; mflag:= 1; eflag:= 1;
1225         flsc:= flsc + 1; oh:= 0; bn:= bn + 1;
1226         fill_t_list_with_delimiter;
1227         dl:= 87{,};
1228 9804: {prepare parsing of actual parameter:}
1229         fill_t_list(rlsc);

```

```

1230         aflag:= 0; goto 9802;

1231     99: {} }                                     {EU}
1232         if mflag = 1 then goto 8702;
1233         repeat production_of_object_program(1)
1234         until not thenelse;
1235         tlsc:= tlsc - 1; unload_t_list_element(mflag);
1236         goto 1;

1237     100: {[ ] }                                  {EE}
1238         if eflag = 0 then reservation_of_arrays;
1239         oflag:= 1; oh:= 0;
1240         fill_t_list(eflag); fill_t_list(iflag);
1241         fill_t_list(mflag); fill_t_list(fflag);
1242         fill_t_list(jflag); fill_t_list(nid);
1243         eflag:= 1; iflag:= 1; mflag:= 0;
1244         fill_t_list_with_delimiter;
1245         if jflag = 0 then generate_address {of storage function};
1246         goto 1;

1247     101: {[ ] }                                  {EF}
1248         repeat production_of_object_program(1)
1249         until not thenelse;
1250         tlsc:= tlsc - 1;
1251         if iflag = 0
1252         then begin {array declaration:}
1253             fill_result_list(0,21495808+aic{2S 'aic' A});
1254             fill_result_list(90{RSF}+ibd,0) {RSF or ISF};
1255             arrb:= d15 + d25 + d26;
1256             if ibd = 1 then arrb:= arrb + d19;
1257             arra:= nlib + nlsc;
1258             repeat store[arra-1]:= arrb + pnlv;
1259                 if store[arra-2] mod d3 = 0
1260                 then arra:= arra - 2 else arra:= arra - 3;
1261                 pnlv:= pnlv + (ic + 3) * d5; aic:= aic - 1
1262             until aic = 0;
1263             read_until_next_delimiter;
1264             if dl <> 91 then goto 1103;
1265             eflag:= 0; goto 1
1266         end;
1267         unload_t_list_element(nid); unload_t_list_element(jflag);
1268         unload_t_list_element(fflag); unload_t_list_element(mflag);
1269         unload_t_list_element(iflag); unload_t_list_element(eflag);
1270         if jflag = 0
1271         then begin {subscripted variable:}

```

```

1272         aflag:= 1; fill_result_list(56{IND},0);
1273         goto 1
1274     end;
1275     {switch designator;}
1276     nflag:= 1; fill_result_list(29{SSI},0);
1277     read_next_symbol;
1278     id:= store[nlib+nid];
1279     pflag:= 0; goto 3;

1280 102: {|<}                                     {KS}
1281     qc:= 1; qb:= 0; qa:= 1;
1282     repeat read_next_symbol;
1283         if dl = 102{|<} then qc:= qc + 1;
1284         if dl = 103{|>} then qc:= qc - 1;
1285         if qc > 0
1286             then begin qb:= qb + dl * qa; qa:= qa * d8;
1287                 if qa = d24
1288                     then begin fill_result_list(0,qb); qb:= 0; qa:= 1 end
1289             end
1290     until qc = 0;
1291     fill_result_list(0,qb+255{end marker}*qa);
1292     oflag:= 0; goto 1;

1293 104: {begin}                                     {LZ}
1294     if store[tlsc-1] <> 161 {block-begin marker}
1295     then reservation_of_arrays;
1296     goto 8501;

1297 105: {end}                                       {FS}
1298     reservation_of_arrays;
1299     repeat empty_t_list_through_thenelse
1300     until not do_in_t_list;
1301     if sflag = 0
1302     then begin if store[tlsc-1] = 161 {blok-begin marker}
1303             then begin tlsc:= tlsc - 3;
1304                     nlsc:= store[tlsc+1];
1305                     fill_future_list(flib+store[tlsc],rlsc+1);
1306                     fill_result_list(12{RET},0);
1307                     bn:= bn - 1;
1308                     goto 105
1309             end
1310     end
1311     else begin {end of switch declaration}
1312             sflag:= 0;
1313             repeat tlsc:= tlsc - 2;

```

```

1314         fill_result_list(1,88604672+store[tlsc])
1315         {2T 'stacked RLSC' A}
1316         until store[tlsc-1] <> 160{switch comma};
1317         tlsc:= tlsc - 1; unload_t_list_element(nid);
1318         label_declaration;
1319         fill_result_list(0,85983232+48) {1T 16X1};
1320         tlsc:= tlsc - 1;
1321         fill_future_list(flib+store[tlsc],rlsc)
1322     end;
1323     eflag:= 0;
1324     if dl <> 105{end} then goto 1;
1325     tlsc:= tlsc - 1;
1326     if tlsc = tlib + 1 then goto 1052;
1327     repeat read_next_symbol
1328     until (dl = 91{;}) or (dl = 84{else}) or (dl = 105{end});
1329     jflag:= 0; pflag:= 0; fflag:= 0; nflag:= 0;
1330     goto 2;

1331 106: {own}                                {KH}
1332     new_block_by_declaration;
1333     read_next_symbol;
1334     if dl = 109{real} then ibd:= 0 else ibd:= 1;
1335     read_until_next_delimiter;
1336     if nflag = 0 then goto 1102;
1337     goto 1082;

1338 107: {Boolean}                            {KZ}
1339     goto 108{integer};

1340 108: {integer}                            {KZ}
1341     ibd:= 1;
1342     new_block_by_declaration;
1343     read_until_next_delimiter;
1344 1081: if nflag = 0
1345     then begin if dl = 110{array} then goto 1101;
1346             goto 112{procedure}
1347     end;
1348     {scalar:}
1349     if bn <> 0 then goto 1083;
1350 1082: {static addressing}
1351     id:= gvc;
1352     if ibd = 1
1353     then begin id:= id + d19; gvc:= gvc + 1 end
1354     else gvc:= gvc + 2;
1355     fill_name_list;

```

```

1356     if dl = 87{,}
1357     then begin read_until_next_delimiter;
1358             goto 1082
1359     end;
1360     goto 1;
1361 1083: {dynamic addressing}
1362     id:= pnlv + d15;
1363     if ibd = 1
1364     then begin id:= id + d19;
1365             pnlv:= pnlv + 32; lvc:= lvc + 1
1366     end
1367     else begin pnlv:= pnlv + 2 * 32; lvc:= lvc + 2 end;
1368     fill_name_list;
1369     if dl = 87{,}
1370     then begin read_until_next_delimiter;
1371             goto 1083
1372     end;
1373     read_until_next_delimiter;
1374     if (dl <= 106{own}) or (dl > 109{real})
1375     then begin reservation_of_local_variables;
1376             goto 2
1377     end;
1378     if dl = 109{real} then ibd:= 0 else ibd:= 1;
1379     read_until_next_delimiter;
1380     if nflag = 1 then goto 1083 {more scalars};
1381     reservation_of_local_variables;
1382     if dl = 110{array} then goto 1101;
1383     goto 3;

1384 109: {real}                                     {KE}
1385     ibd:= 0;
1386     new_block_by_declaration;
1387     read_until_next_delimiter;
1388     if nflag = 1 then goto 1081;
1389     goto 2;

1390 110: {array}                                     {KF}
1391     ibd:= 0;
1392     new_block_by_declaration;
1393 1101: if bn <> 0 then goto 1103;
1394 1102: {static bounds, constants only:}
1395     id:= 3 * d24;
1396     if ibd <> 0 then id:= id + d19;
1397     repeat arra:= nlsc; arrb:= tlsc;
1398     repeat {read identifier list:}

```

```

1399         read_until_next_delimiter; fill_name_list
1400 until dl = 100{[]};
1401 arrc:= 0;
1402 fill_t_list(2-ibd); {delta[0]}
1403 repeat {read bound-pair list:}
1404     {lower bound:}
1405     read_until_next_delimiter;
1406     if dl <> 90 {:}
1407     then if dl = 64{+}
1408         then begin read_until_next_delimiter;
1409             arrd:= inw
1410         end
1411         else begin read_until_next_delimiter;
1412             arrd:= - inw
1413         end
1414     else arrd:= inw;
1415     arrc:= arrc - (arrd * store[tlsc-1]) mod d26;
1416     {upper bound:}
1417     read_until_next_delimiter;
1418     if nflag = 0
1419     then if dl = 65{-}
1420         then begin read_until_next_delimiter;
1421             arrd:= - inw - arrd
1422         end
1423         else begin read_until_next_delimiter;
1424             arrd:= inw - arrd
1425         end
1426     else arrd:= inw - arrd;
1427     if dl = 101{[]}
1428     then fill_t_list(- ((arrd + 1) * store[tlsc-1]) mod d26)
1429     else fill_t_list(((arrd + 1) * store[tlsc-1]) mod d26)
1430 until dl = 101{[]};
1431 arrd:= nlsc;
1432 repeat {construction of storage function in constant list:}
1433     store[nlib+arrd-1]:= store[nlib+arrd-1] + klsc;
1434     fill_constant_list(gvc); fill_constant_list(gvc+arrc);
1435     tlsc:= arrb;
1436     repeat fill_constant_list(store[tlsc]);
1437         tlsc:= tlsc + 1
1438     until store[tlsc-1] <= 0;
1439     gvc:= gvc - store[tlsc-1]; tlsc:= arrb;
1440     if store[nlib+arrd-2] mod d3 = 0
1441     then arrd:= arrd - 2 else arrd:= arrd - 3
1442 until arrd = arra;
1443 read_until_next_delimiter

```

```

1444     until dl <> 87{,};
1445     goto 91{;};
1446 1103: {dynamic bounds,arithmetic expressions;}
1447     ic:= 0; aic:= 0; id:= 0;
1448     repeat aic:= aic + 1;
1449         read_until_next_delimiter;
1450         fill_name_list
1451     until dl <> 87{,};
1452     eflag:= 1; oflag:= 1;
1453     goto 8501;

1454 111: {switch}                                     {HE}
1455     reservation_of_arrays;
1456     sflag:= 1;
1457     new_block_by_declaration;
1458     goto 1;

1459 112: {procedure}                                   {HY}
1460     reservation_of_arrays;
1461     new_block_by_declaration;
1462     fill_result_list(2,88080384+flsc) {2T 'flsc'};
1463     fill_t_list(flsc); flsc:= flsc + 1;
1464     read_until_next_delimiter; look_for_name;
1465     label_declaration; intro_new_block;
1466     new_block_by_declaration1;
1467     if dl = 91{;} then goto 1;
1468     {formal parameter list;}
1469     repeat read_until_next_delimiter; id:= pnlv + d15 + d16;
1470         fill_name_list; pnlv:= pnlv + 2 * d5 {reservation PARD}
1471     until dl <> 87;
1472     read_until_next_delimiter; {for ; after }}
1473 1121: read_until_next_delimiter;
1474     if nflag = 1 then goto 2;
1475     if dl = 104{begin} then goto 3;
1476     if dl <> 115{value} then goto 1123 {specification part};
1477     {value part;}
1478     spe:= d26; {value flag}
1479 1122: repeat read_until_next_delimiter; look_for_name;
1480     store[nlib+nid]:= store[nlib+nid] + spe
1481     until dl <> 87;
1482     goto 1121;
1483 1123: {specification part;}
1484     if (dl = 113{string}) or (dl = 110{array})
1485     then begin spe:= 0; goto 1122 end;
1486     if (dl = 114{label}) or (dl = 111{switch})

```

```

1487     then begin spe:= d17; goto 1122 end;
1488     if dl = 112{procedure}
1489     then begin spe:= d18; goto 1122 end;
1490     if dl = 109{real}
1491     then spe:= 0 else spe:= d19;
1492     if (dl <= 106) or (dl > 109) then goto 3; {if,for,goto}
1493     read_until_next_delimiter; {for delimiter following real/integer/boolean}
1494     if dl = 112{procedure}
1495     then begin spe:= d18; goto 1122 end;
1496     if dl = 110{array} then goto 1122;
1497 1124: look_for_name; store[nlib+nid]:= store[nlib+nid] + spe;
1498     if store[nlib+nid] >= d26
1499     then begin id:= store[nlib+nid] - d26;
1500             id:= (id div d17) * d17 + id mod d16;
1501             store[nlib+nid]:= id;
1502             address_to_register; {generates 2S 'PARD position' A}
1503             if spe = 0
1504             then fill_result_list(14{TRAD},0)
1505             else fill_result_list(16{TIAD},0);
1506             address_to_register; {generates 2S 'PARD position' A}
1507             fill_result_list(35{TFR},0);
1508             fill_result_list(85{ST},0)
1509     end;
1510     if dl = 87{,}
1511     then begin read_until_next_delimiter;
1512             goto 1124
1513     end;
1514     goto 1121;

1515 1052:
1516 end {main_scan};

1517 procedure program_loader;                                {RZ}
1518 var i,j,ll,list_address,id,mcp_count,crfa: integer;
1519     heptade_count,parity_word,read_location,stock: integer;
1520     from_store: 0..1;
1521     use: boolean;

1522 function logical_sum(n,m: integer): integer;
1523 {emulation of a machine instruction}
1524 var i,w: integer;
1525 begin w:= 0;
1526     for i:= 0 to 26 do
1527     begin w:= w div 2;

```

```

1528     if n mod 2 = m mod 2 then w:= w + d26;
1529     n:= n div 2; m := m div 2
1530     end;
1531     logical_sum:= w
1532 end {logical_sum};

1533 procedure complete_bitstock;                                {RW}
1534 var i,w: integer;
1535 begin while bitcount > 0 {i.e., at most 20 bits in stock} do
1536     begin heptade_count:= heptade_count + 1;
1537     case from_store of
1538     0: {bit string read from store:}
1539         begin if heptade_count > 0
1540             then begin bitcount:= bitcount + 1;
1541                 heptade_count:= - 3;
1542                 read_location:= read_location - 1;
1543                 stock:= store[read_location];
1544                 w:= stock div d21;
1545                 stock:= (stock mod d21) * 64
1546             end
1547             else begin w:= stock div d20;
1548                 stock:= (stock mod d20) * 128
1549             end
1550         end;
1551     1: {bit string read from tape:}
1552         begin read(lib_tape,w);
1553             if heptade_count > 0
1554             then begin {test parity of the previous 4 heptades}
1555                 bitcount:= bitcount + 1;
1556                 parity_word:=
1557                     logical_sum(parity_word,parity_word div d4)
1558                     mod d4;
1559                 if parity_word in [0,3,5,6,9,10,12,15]
1560                 then stop(105);
1561                 heptade_count:= -3; parity_word:= w;
1562                 w:= w div 2
1563             end
1564             else parity_word:= logical_sum(parity_word,w)
1565             end
1566         end {case};
1567         for i:= 1 to bitcount - 1 do w:= 2 * w;
1568         bitstock:= bitstock + w; bitcount:= bitcount - 7
1569     end {while}
1570 end {complete_bitstock};

```

```

1571 function read_bit_string(n: integer): integer;           {RW}
1572 var i,w: integer;
1573 begin w:= 0;
1574   for i:= 1 to n do
1575     begin w:= 2 * w + bitstock div d26;
1576       bitstock:= (bitstock mod d26) * 2
1577     end;
1578     read_bit_string:= w; bitcount:= bitcount + n;
1579     complete_bitstock
1580 end {read_bit_string};

1581 procedure prepare_read_bit_string1;
1582 var i: integer;
1583 begin for i:= 1 to 27 - bitcount do bitstock:= 2 * bitstock;
1584   bitcount:= 21 - bitcount; heptade_count:= 0;
1585   from_store:= 0; complete_bitstock
1586 end {prepare_read_bit_string1};

1587 procedure prepare_read_bit_string2;
1588 begin bitstock:= 0; bitcount:= 21; heptade_count:= 0;
1589   from_store:= 0; complete_bitstock;
1590   repeat until read_bit_string(1) = 1
1591 end {prepare_read_bit_string2};

1592 procedure prepare_read_bit_string3;
1593 var w: integer;
1594 begin from_store:= 1; bitstock:= 0; bitcount:= 21;
1595   repeat read(lib_tape,w) until w <> 0;
1596   if w <> 30 {D} then stop(106);
1597   heptade_count:= 0; parity_word:= 1;
1598   complete_bitstock;
1599   repeat until read_bit_string(1) = 1
1600 end {prepare_read_bit_string3};

1601 function address_decoding: integer;                       {RY}
1602 var w,a,n: integer;
1603 begin w:= bitstock;
1604   if w < d26 {code starts with 0}
1605   then begin {0}      n:= 1; a:= 0; w:= 2 * w end
1606   else begin {1xxxxx} n:= 6; a:= (w div d21) mod d5;
1607     w:= (w mod d21) * d6
1608   end;
1609   if w < d25 {00}
1610   then begin {00} n:= n + 2; a:= 32 * a + 0; w:= w * 4 end else
1611   if w < d26 {01}

```

```

1612     then begin {01xx} n:= n + 4; a:= 32 * a + w div d23;
1613           if a mod d5 < 6
1614             then {010x} a:= a - 3 else {011x} a:= a - 2;
1615             w:= (w mod d23) * d4
1616           end
1617     else begin {1xxxxx} n:= n + 6;
1618           a:= a * 32 + (w div d21) mod d5;
1619           w:= (w mod d21) * d6
1620           end;
1621     if w < d25 {00}
1622     then begin {00} n:= n + 2; a:= 32 * a + 1 end else
1623     if w < d26 {01}
1624     then begin {01x} n:= n + 3; a := 32 * a + w div d24 end
1625     else begin {1xxxxx} n:= n + 6;
1626           a:= 32 * a + (w div d21) mod d5
1627           end;
1628     w:= read_bit_string(n); address_decoding:= a
1629     end {address_decoding};

1630 function read_mask: integer;                                {RN}
1631 var c: 0 .. 19;
1632 begin
1633   if bitstock < d26 {code starts with 0}
1634   then {0x} c:= read_bit_string(2) else
1635   if bitstock < d26 + d25 {01}
1636   then {10x} c:= read_bit_string(3) - 2
1637   else {11xxxx} c:= read_bit_string(6) - 44;
1638   case c of
1639     0: read_mask:= 656; {0, 2S 0 A }
1640     1: read_mask:= 14480; {3, 2B 0 A }
1641     2: read_mask:= 10880; {2, 2T 0 XO }
1642     3: read_mask:= 2192; {0, 2B 0 A }
1643     4: read_mask:= 144; {0, 2A 0 A }
1644     5: read_mask:= 10368; {2, 2B 0 XO }
1645     6: read_mask:= 6800; {1, 2T 0 A }
1646     7: read_mask:= 0; {0, 0A 0 XO }
1647     8: read_mask:= 12304; {3, 0A 0 A }
1648     9: read_mask:= 10883; {2, N 2T 0 XO }
1649     10: read_mask:= 6288; {1, 2B 0 A }
1650     11: read_mask:= 4128; {1, 0A 0 XO B }
1651     12: read_mask:= 8832; {2, 2S 0 XO }
1652     13: read_mask:= 146; {0, Y 2A 0 A }
1653     14: read_mask:= 256; {0, 4A 0 XO }
1654     15: read_mask:= 134; {0, Y 2A 0 XO P}
1655     16: read_mask:= 402; {0, Y 6A 0 A }

```

```

1656         17: read_mask:= 4144; {1, 0A 0 X0 C }
1657         18: read_mask:= 16; {0, 0A 0 A }
1658         19: read_mask:= address_decoding
1659     end {case}
1660 end {read_mask};

1661 function read_binary_word: integer; {RF}
1662 var w: integer; opc: 0 .. 3;
1663 begin if bitstock < d26 {code starts with 0}
1664     then begin {OPC >= 8}
1665         if bitstock < d25 {00}
1666             then if bitstock < d24 {000}
1667                 then w:= 4 {code is 000x}
1668                 else w:= 5 {code is 001xx}
1669             else if bitstock < d25 + d24 {010}
1670                 then if bitstock < d25 + d23 {0100}
1671                     then w:= 6 {0100xx}
1672                     else w:= 7 {0101xxx}
1673                 else w:= 10 {011xxxxxxx};
1674             w:= read_bit_string(w);
1675             if w < 2 {000x} then {no change} else
1676             if w < 8 {001xx} then w:= w - 2 else
1677             if w < 24 {010xx} then w:= w - 10 else
1678             if w < 48 {0101xxx} then w:= w - 30
1679             else {011xxxxxxx} w:= w - 366;
1680             read_binary_word:= opc_table[w]
1681         end {0}
1682     else begin w:= read_bit_string(1);
1683             w:= read_mask; opc:= w div d12;
1684             w:= (w mod d12) * d15 + address_decoding;
1685             case opc of
1686                 0: ;
1687                 1: w:= w + list_address;
1688                 2: begin if w div d17 mod 2 = 1 {d17 = 1}
1689                     then w:= w - d17
1690                     else w:= w + d19;
1691                     w:= w - w mod d15 + store[flib + w mod d15]
1692                 end;
1693                 3: if klib = crfb
1694                     then w:= w - w mod d15 + store[mlib+w mod d15]
1695                     else w:= w + klib
1696                 end {case};
1697             read_binary_word:= w
1698         end {1}
1699 end {read_binary_word};

```

```

1700 procedure test_bit_stock;                                {RH}
1701 begin if bitstock <> 63 * d21 then stop(107)
1702 end {test_bit_stock};

1703 procedure typ_address(a: integer);                       {RT}
1704 begin writeln(output);
1705   write(output,a div 1024:2,'',(a mod 1024) div 32:2,' ',a mod 32:2)
1706 end {typ_address};

1707 procedure read_list;                                     {RL}
1708 var i,j,w: integer;
1709 begin for i:= ll - 1 downto 0 do
1710   begin w:= read_binary_word;
1711     if list_address + i <= flib + flsc
1712     then begin {shift FLI downwards}
1713       if flib <= read_location
1714       then stop(98);
1715       for j:= 0 to flsc - 1 do
1716         store[read_location+j]:= store[flib+j];
1717         flib:= read_location
1718       end;
1719       store[list_address+i]:= w
1720     end {for i};
1721     test_bit_stock;
1722 end {read_list};

1723 function read_crf_item: integer;                         {RS}
1724 begin if crfa mod 2 = 0
1725   then read_crf_item:= store[crfa div 2] div d13
1726   else read_crf_item:= store[crfa div 2] mod d13;
1727   crfa:= crfa + 1
1728 end {read_crf_item};

1729 begin {of program loader}
1730   rlib:= (klie - rlsc - klsc) div 32 * 32;
1731   {increment entries in future list:}
1732   for i:= 0 to flsc - 1 do store[flib+i]:= store[flib+i] + rlib;
1733   {move KLI to final position:}
1734   for i:= klsc - 1 downto 0 do store[rlib+rlsc+i]:= store[klib+i];
1735   klib:= rlib + rlsc;
1736   {prepare mcp-need analysis:}
1737   mcpe:= rlib; mcp_count:= 0;
1738   for i:= 0 to 127 do store[mlib+i]:= 0;
1739   {determine primary need of MCP's from name list:}

```

```

1740     i:= nlsc0;
1741     while i > nlscop do
1742     begin id:= store[nlib+i-1];
1743         if store[nlib+i-2] mod d3 = 0
1744         then {at most 4 letter/digit identifier} i:= i - 2
1745         else {at least 5 letters or digits} i:= i - 3;
1746         if (id div d15) mod 2 = 0
1747         then begin {MCP is used} mcp_count:= mcp_count + 1;
1748                 store[mlib+(store[flib+id mod d15]-rlib) mod d15]:=
1749                 - (flib + id mod d15)
1750             end
1751     end;
1752     {determine secondary need using the cross-reference list:}
1753     crfa:= 2 * crfb;
1754     ll:= read_crf_item {for MCP length};
1755     while ll <> 7680 {end marker} do
1756     begin i:= read_crf_item {for MCP number};
1757         use:= (store[mlib+i] <> 0);
1758         j:= read_crf_item {for number of MCP needing the current one};
1759         while j <> 7680 {end marker} do
1760         begin use:= use or (store[mlib+j] <> 0); j:= read_crf_item end;
1761         if use
1762         then begin mcpe:= mcpe - ll;
1763                 if mcpe <= mcpb then stop(25);
1764                 if store[mlib+i] < 0
1765                 then {primary need} store[-store[mlib+i]]:= mcpe
1766                 else {only secondary need} mcp_count:= mcp_count + 1;
1767                 store[mlib+i]:= mcpe
1768             end;
1769         ll:= read_crf_item
1770     end;
1771     {load result list RLI:}
1772     ll:= rlsc; read_location:= rnsb;
1773     prepare_read_bit_string1;
1774     list_address:= rlib; read_list;
1775     if store[rlib] <> opc_table[89{START}] then stop(101);
1776     typ_address(rlib);
1777     {copy MLI:}
1778     for i:= 0 to 127 do store[crfb+i]:= store[mlib+i];
1779     klib:= crfb; flsc:= 0;
1780     {load MCP's from store:}
1781     prepare_read_bit_string2;
1782     ll:= read_bit_string(13) {for length or end marker};
1783     while ll < 7680 do
1784     begin i:= read_bit_string(13) {for MCP number};

```

```

1785     list_address:= store[crfb+i];
1786     if list_address <> 0
1787     then begin read_list; test_bit_stock;
1788             mcp_count:= mcp_count - 1;
1789             store[crfb+i]:= 0
1790         end
1791     else repeat read_location:= read_location - 1
1792             until store[read_location] = 63 * d21;
1793     prepare_read_bit_string2; ll:= read_bit_string(13)
1794 end;
1795 {load MCP's from tape;}
1796 reset(lib_tape);
1797 while mcp_count <> 0 do
1798     begin writeln(output);
1799         writeln(output,'load (next) library tape into the tape reader');
1800         prepare_read_bit_string3;
1801         ll:= read_bit_string(13) {for length or end marker};
1802         while ll < 7680 do
1803             begin i:= read_bit_string(13) {for MCP number};
1804                 list_address:= store[crfb+i];
1805                 if list_address <> 0
1806                 then begin read_list; test_bit_stock;
1807                         mcp_count:= mcp_count - 1;
1808                         store[crfb+i]:= 0
1809                     end
1810                 else repeat repeat read(lib_tape,ll) until ll = 0;
1811                         read(lib_tape,ll)
1812                     until ll = 0;
1813                 prepare_read_bit_string3; ll:= read_bit_string(13)
1814             end
1815         end;
1816 {program loading completed;}
1817     typ_address(mcpe)
1818 end {program_loader};

1819 {main program}

1820 begin
1821 {initialization of word_del_table}                                {HT}
1822     word_del_table[10]:= 15086; word_del_table[11]:= 43;
1823     word_del_table[12]:= 1; word_del_table[13]:= 86;
1824     word_del_table[14]:= 13353; word_del_table[15]:= 10517;
1825     word_del_table[16]:= 81; word_del_table[17]:= 10624;
1826     word_del_table[18]:= 44; word_del_table[19]:= 0;

```

```

1827 word_del_table[20]:= 0; word_del_table[21]:= 10866;
1828 word_del_table[22]:= 0; word_del_table[23]:= 0;
1829 word_del_table[24]:= 106; word_del_table[25]:= 112;
1830 word_del_table[26]:= 0; word_del_table[27]:= 14957;
1831 word_del_table[28]:= 2; word_del_table[29]:= 2;
1832 word_del_table[30]:= 95; word_del_table[31]:= 115;
1833 word_del_table[32]:= 14304; word_del_table[33]:= 0;
1834 word_del_table[34]:= 0; word_del_table[35]:= 0;
1835 word_del_table[36]:= 0; word_del_table[37]:= 0;
1836 word_del_table[38]:= 107;

1837 {initialization of flex_table} {LK}
1838 flex_table[ 0]:= -2; flex_table[ 1]:= 19969; flex_table[ 2]:= 16898;
1839 flex_table[ 3]:= -0; flex_table[ 4]:= 18436; flex_table[ 5]:= -0;
1840 flex_table[ 6]:= -0; flex_table[ 7]:= 25863; flex_table[ 8]:= 25096;
1841 flex_table[ 9]:= -0; flex_table[10]:= -0; flex_table[11]:= -1;
1842 flex_table[12]:= -0; flex_table[13]:= -1; flex_table[14]:= 41635;
1843 flex_table[15]:= -0; flex_table[16]:= 31611; flex_table[17]:= -0;
1844 flex_table[18]:= -0; flex_table[19]:= 17155; flex_table[20]:= -0;
1845 flex_table[21]:= 23301; flex_table[22]:= 25606; flex_table[23]:= -0;
1846 flex_table[24]:= -0; flex_table[25]:= 25353; flex_table[26]:= 30583;
1847 flex_table[27]:= -0; flex_table[28]:= -1; flex_table[29]:= -0;
1848 flex_table[30]:= -0; flex_table[31]:= -1; flex_table[32]:= 19712;
1849 flex_table[33]:= -0; flex_table[34]:= -0; flex_table[35]:= 14365;
1850 flex_table[36]:= -0; flex_table[37]:= 14879; flex_table[38]:= 15136;
1851 flex_table[39]:= -0; flex_table[40]:= -0; flex_table[41]:= 15907;
1852 flex_table[42]:= -1; flex_table[43]:= -0; flex_table[44]:= -1;
1853 flex_table[45]:= -0; flex_table[46]:= -0; flex_table[47]:= -1;
1854 flex_table[48]:= -0; flex_table[49]:= 17994; flex_table[50]:= 14108;
1855 flex_table[51]:= -0; flex_table[52]:= 14622; flex_table[53]:= -0;
1856 flex_table[54]:= -0; flex_table[55]:= 15393; flex_table[56]:= 15650;
1857 flex_table[57]:= -0; flex_table[58]:= -0; flex_table[59]:= 30809;
1858 flex_table[60]:= -0; flex_table[61]:= -1; flex_table[62]:= 30326;
1859 flex_table[63]:= -0; flex_table[64]:= 19521; flex_table[65]:= -0;
1860 flex_table[66]:= -0; flex_table[67]:= 12309; flex_table[68]:= -0;
1861 flex_table[69]:= 12823; flex_table[70]:= 13080; flex_table[71]:= -0;
1862 flex_table[72]:= -0; flex_table[73]:= 13851; flex_table[74]:= -1;
1863 flex_table[75]:= -0; flex_table[76]:= -1; flex_table[77]:= -0;
1864 flex_table[78]:= -0; flex_table[79]:= -1; flex_table[80]:= -0;
1865 flex_table[81]:= 11795; flex_table[82]:= 12052; flex_table[83]:= -0;
1866 flex_table[84]:= 12566; flex_table[85]:= -0; flex_table[86]:= -0;
1867 flex_table[87]:= 13337; flex_table[88]:= 13594; flex_table[89]:= -0;
1868 flex_table[90]:= -0; flex_table[91]:= 31319; flex_table[92]:= -0;
1869 flex_table[93]:= -1; flex_table[94]:= -1; flex_table[95]:= -0;
1870 flex_table[96]:= -0; flex_table[97]:= 9482; flex_table[98]:= 9739;

```

```

1871 flex_table[ 99]:= -0; flex_table[100]:= 10253; flex_table[101]:= -0;
1872 flex_table[102]:= -0; flex_table[103]:= 11024; flex_table[104]:= 11281;
1873 flex_table[105]:= -0; flex_table[106]:= -0; flex_table[107]:= 31832;
1874 flex_table[108]:= -0; flex_table[109]:= -1; flex_table[110]:= -1;
1875 flex_table[111]:= -0; flex_table[112]:= 31040; flex_table[113]:= -0;
1876 flex_table[114]:= -0; flex_table[115]:= 9996; flex_table[116]:= -0;
1877 flex_table[117]:= 10510; flex_table[118]:= 10767; flex_table[119]:= -0;
1878 flex_table[120]:= -0; flex_table[121]:= 11538; flex_table[122]:= -2;
1879 flex_table[123]:= -0; flex_table[124]:= -2; flex_table[125]:= -0;
1880 flex_table[126]:= -0; flex_table[127]:= -2;

1881 {preparation of prescan} {LE}
1882 rns_state:= virginal; scan:= 1;
1883 read_until_next_delimiter;

1884 prescan; {HK}

1885 {writeln;
1886 for bn:= plib to plie do writeln(bn:5,store[bn]:10);
1887 writeln;}

1888 {preparation of main scan:} {HL}
1889 rns_state:= virginal; scan:= - 1;
1890 iflag:= 0; mflag:= 0; vflag:= 0; bn:= 0; aflag:= 0; sflag:= 0;
1891 eflag:= 0; rlsc:= 0; flsc:= 0; klsc:= 0; vlam:= 0;
1892 flib:= rnsb + 1; klib:= flib + 16; nlib:= klib + 16;
1893 if nlib + nlsc0 >= plib then stop(25);
1894 nlsc:= nlsc0; tlsc:= tlib; gvc:= gvc0;
1895 fill_t_list(161);
1896 {prefill of name list:}
1897 store[nlib + 0]:= 27598040;
1898 store[nlib + 1]:= 265358; {read}
1899 store[nlib + 2]:= 134217727 - 6;
1900 store[nlib + 3]:= 61580507;
1901 store[nlib + 4]:= 265359; {print}
1902 store[nlib + 5]:= 134217727 - 53284863;
1903 store[nlib + 6]:= 265360; {TAB}
1904 store[nlib + 7]:= 134217727 - 19668591;
1905 store[nlib + 8]:= 265361; {NLCR}
1906 store[nlib + 9]:= 134217727 - 0;
1907 store[nlib + 10]:= 134217727 - 46937177;
1908 store[nlib + 11]:= 265363; {SPACE}
1909 store[nlib + 12]:= 53230304;
1910 store[nlib + 13]:= 265364; {stop}
1911 store[nlib + 14]:= 59085824;

```

```

1912   store[nlib + 15] := 265349;           {abs}
1913   store[nlib + 16] := 48768224;
1914   store[nlib + 17] := 265350;           {sign}
1915   store[nlib + 18] := 61715680;
1916   store[nlib + 19] := 265351;           {sqrt}
1917   store[nlib + 20] := 48838656;
1918   store[nlib + 21] := 265352;           {sin}
1919   store[nlib + 22] := 59512832;
1920   store[nlib + 23] := 265353;           {cos}
1921   store[nlib + 24] := 48922624;
1922   store[nlib + 25] := 265355;           {ln}
1923   store[nlib + 26] := 53517312;
1924   store[nlib + 27] := 265356;           {exp}
1925   store[nlib + 28] := 134217727 -      289;
1926   store[nlib + 29] := 29964985;
1927   store[nlib + 30] := 265357;           {entier}

1928   store[nlib + 31] := 134217727 - 29561343;
1929   store[nlib + 32] := 294912;           {SUM}
1930   store[nlib + 33] := 134217727 - 14789691;
1931   store[nlib + 34] := 134217727 - 15115337;
1932   store[nlib + 35] := 294913;           {PRINTTEXT}
1933   store[nlib + 36] := 134217727 - 27986615;
1934   store[nlib + 37] := 294914;           {EVEN}
1935   store[nlib + 38] := 134217727 -      325;
1936   store[nlib + 39] := 21928153;
1937   store[nlib + 40] := 294915;           {arctan}
1938   store[nlib + 41] := 134217727 - 15081135;
1939   store[nlib + 42] := 294917;           {FLOT}
1940   store[nlib + 43] := 134217727 - 14787759;
1941   store[nlib + 44] := 294918;           {FIXT}
1942   store[nlib + 45] := 134217727 -      3610;
1943   store[nlib + 46] := 134217727 - 38441163;
1944   store[nlib + 47] := 294936;           {ABSFIXT}

1945   intro_new_block2;
1946   bitcount:= 0; bitstock:= 0; rnsb:= bim;
1947   fill_result_list(96{START},0);
1948   pos:= 0;
1949   main_scan;                               {EL}
1950   fill_result_list(97{STOP},0);

1951   {writeln; writeln('FLI:');
1952   for bn:= 0 to flsc-1 do
1953   writeln(bn:5,store[flib+bn]:10);}

```

```

1954 {writeln; writeln('KLI:');
1955 for bn:= 0 to klsc-1 do
1956   writeln(bn:5,store[klib+bn]:10,
1957           (store[klib+bn] mod 134217728) div 16777216 : 10,
1958           (store[klib+bn] mod 16777216) div 2097152 : 2,
1959           (store[klib+bn] mod 2097152) div 524288 : 3,
1960           (store[klib+bn] mod 524288) div 131072 : 2,
1961           (store[klib+bn] mod 131072) div 32768 : 2,
1962           (store[klib+bn] mod 32768) div 1024 : 4,
1963           (store[klib+bn] mod 1024) div 32 : 3,
1964           (store[klib+bn] mod 32) div 1 : 3);}

1965 {preparation of program loader}
1966   opc_table[ 0]:= 33; opc_table[ 1]:= 34; opc_table[ 2]:= 16;
1967   opc_table[ 3]:= 56; opc_table[ 4]:= 58; opc_table[ 5]:= 85;
1968   opc_table[ 6]:= 9;  opc_table[ 7]:= 14; opc_table[ 8]:= 18;
1969   opc_table[ 9]:= 30; opc_table[10]:= 13; opc_table[11]:= 17;
1970   opc_table[12]:= 19; opc_table[13]:= 20; opc_table[14]:= 31;
1971   opc_table[15]:= 35; opc_table[16]:= 39; opc_table[17]:= 61;
1972   opc_table[18]:= 8;  opc_table[19]:= 10; opc_table[20]:= 11;
1973   opc_table[21]:= 12; opc_table[22]:= 15;
1974   for ii:= 23 to 31 do opc_table[ii]:= ii - 2;
1975   opc_table[32]:= 32; opc_table[33]:= 36; opc_table[34]:= 37;
1976   opc_table[35]:= 38;
1977   for ii:= 36 to 51 do opc_table[ii]:= ii + 4;
1978   opc_table[52]:= 57; opc_table[53]:= 59; opc_table[54]:= 60;
1979   for ii:= 55 to 102 do opc_table[ii]:= ii + 7;

1980   store[crfb+ 0]:= 30 * d13 + 0; store[crfb+ 1]:= 7680 * d13 + 20;
1981   store[crfb+ 2]:= 1 * d13 + 7680; store[crfb+ 3]:= 12 * d13 + 2;
1982   store[crfb+ 4]:= 7680 * d13 + 63; store[crfb+ 5]:= 3 * d13 + 7680;
1983   store[crfb+ 6]:= 15 * d13 + 4; store[crfb+ 7]:= 3 * d13 + 7680;
1984   store[crfb+ 8]:= 100 * d13 + 5; store[crfb+ 9]:= 7680 * d13 + 134;
1985   store[crfb+10]:= 6 * d13 + 24; store[crfb+11]:= 7680 * d13 + 21;
1986   store[crfb+12]:= 24 * d13 + 7680; store[crfb+13]:= 7680 * d13 + 7680;

1987   store[mcpb]:= 63 * d21; store[mcpb+1]:= 63 * d21;

1988   program_loader;

1989   writeln(output); writeln(output); writeln(output);
1990   for ii:= mcpe to rlib + rlsc + klsc - 1 do
1991     writeln(output,ii:5,store[ii]:9)

```

1992 end.