BRUGERVEJLEDNING FOR COMAL-80 COMPILER. (VERSION 1.2)

af Peter Ferdinand.

Indholdsfortegnelse: side

0. Forord.

Dette programsystem kaldes for en oversætter eller en 'compiler', som det hedder i datasproget. Den benævnes i det følgende som en COMAL80-compiler. COMAL80-compileren kan oversætte et COMAL80 program til en intern kode som i det følgende bliver kaldt for p-kode. Denne kode er meget nær ved egentlig maskinkode, så det betyder at det oversatte program kommer til at køre meget hurtigere og bliver langt mere kompakt end det oprindelige COMAL-80 program. P-koden kan køres på et afviklingssystem, som i det følgende vil blive benævnt PCOMAL. Denne kørselsmåde har en række fordele for den erfarne bruger af COMAL-80. Man kan sige at COMAL-80 sproget med denne compiler bliver voksent. Det vil for den professionelle programmør fremover blive et værktøj som bør overvejes på lige fod med andre programmeringssprog som PASCAL og C.

COMAL80-compileren bliver desuden leveret i en version som beregner med 13 cifres nøjagtighed ved alle operationer med decimaltal.

Programmeringssproget COMAL-80 har nogle klare fordele fremfor andre pogrammeringssprog, som betyder at det er særdeles velegenet til selv vanskelige programmeringsopgaver. Det er ikke her pladsen detaljeret, at komme ind på disse fordele, men det skal blot nævnes at COMAL-80 har formået at kombinere BASIC's nemme fejlretningsmuligheder med PASCAL's programstruktur. Når der hertil lægges at strengfaciliteterne og filfaciliterne i COMAL-80 er langt bedre end i PASCAL, så er der hermed skabt forudsætninger for et sprog som bør få stor udbredelse også til proffesionelt brug. Med COMAL80-compileren bliver programmørens værktøj komplet.

COMAL80-compileren vil betyde en væsentlig forøgelse af afviklingshastigheden, en væsentlig formindskelse af programmet, en uafhængighed af COMAL-80 fortolkeren samt en sikkerhed for at programmet ikke kan ændres af en bruger.

Indenfor EDB litteraturen er der foreslået mange forskellige metoder til at afgøre en datamaskines effektivitet og hastighed. Det er ikke alle metoder som er lige hensigtsmæssige og de fleste vil ofte lide af den skavank, at de kun måler en enkelt del af et programsystems ydeevne. En af de klassiske metoder til afprøvning af et datasystem er at oversætte og afvikle Eratosthenes' si. Dette program er blevet afprøvet med COMAL80-compileren og samtidigt med COMAL-80 fortolkeren. Resultatet er vist nedenfor. Programmet ser således ud:

```
0010 //PROGRAM PRIMES
0018 SIZ#:=5461
0020 DIM FLAGS#(0:SIZ#)
0040 PRINT "10 ITERATIONS"
0050 FOR ITER#:=1 TO 10 DO
0060   COUNT#:=0
0070   FOR I#:=0 TO SIZ# DO
0080     FLAGS#(I#):=TRUE
0090   NEXT I#
0100   FOR I#:=0 TO SIZ# DO
0110     IF FLAGS#(I#) THEN
```

```
0120        PRIME#:=I#+I#+3
0130        K#:=I#+PRIME#
0140        WHILE K#(=SIZ# DO
0150          FLAGS#(K#):=FALSE
0160          K#:+PRIME#
0170        ENDWHILE
0180        COUNT#:+1
0190      ENDIF
0192    NEXT I#
0200  NEXT ITER#
0210  PRINT COUNT#;" PRIMES"
0220  END
```

	program i bytes	variable i bytes	system i Kbytes	hastighed i sekunder
COMAL-80	447	138	35	1065
PCOMAL	314	28	12	430

Hastigheden er målt ved 10 gennemløb på en 2.5 mhz Z80 maskine. 'System' er størrelsen af det afviklingssystem som er nødvendigt for at køre programmet og som altid skal være til stede samtidigt med programmet. 'Variable' er den plads som er afsat i lageret til variable eksklusivt den store matrix ,FLAGS# på 4561 * 2 bytes, som er den samme i begge programmer. Ved oversættelsen til PCOMAL fremkommer der en hastighedsforøgelse på 2.5 gang og en formindskelse af programmet med 42 %. Når det dertil lægges at afviklingsmodulet kun fylder 1/3 del, så er der ganske store fordele ved at oversætte programmet.

COMAL80-compileren kan oversætte alle programelementer fra COMAL-80 således som sproget er defineret i METANIC's version. Dog er der ganske få undtagelser, som alene skyldes de særlige betingelser, hvorunder en oversætter arbejder til forskel fra en fortolker. Det betyder, at man kan teste og udvikle et program ved at anvende fortolkeren, mens det færdige program så kan oversættes til p-kode og det kan derefter kan afvikles uafhængigt af COMAL-80 fortolkeren. Det vil i de fleste tilfælde betyde en forøgelse af kørselshastigheden af programmet på ca 2.5 gange afhængigt af hvor meget ind- og udlæsning der sker i programmet. Et program der anvender diskstation og linieskriver meget vil sandsynligvis kun få en lille forøgelse af kørselshastigheden.

Forudsætningen for at kunne anvende denne manual og programsystemet er, at man har et godt kendskab til COMAL-80 . Manualen vil udelukkende beskrive de ændringer, der forekommer i oversætteren i forhold til fortolkeren.

Det er nødvendigt at have adgang til en datamaskine med 64 kbytes lager, to diskette stationer og en COMAL-80 fortolker i swapping udgaven.

COMAL80-compileren leveres på en enkelt diskette som indeholder alle de programmer som er nødvendige for oversættelsen og afviklingen af de oversatte programmer.

1. Kort gennemgang af en oversættelse.

Lad os nu forudsætte at der er blevet udviklet et program som man af den ene elle den anden grund ønsker at få oversat til PCOMAL. Programmet er blevet gennemafprøvet under COMAL-80 i en sådan grad at man er sikker på at alle fejl er blevet opdagede og er blevet rettede. Det er nu muligt at gennemføre en oversættelse af dette program. Vi vil i det følgende kalde programmet for 'TEST' for en nemheds skyld. Dette COMAL-80 program skal gemmes på en diskette som en ASCII fil med typebetegnelsen .CML. Vort program vil derfor hedde TEST.CML

Disketten med betegnelsen COMAL80-compiler indsættes i diskettedrevet til venstre på datamaskinen mærket med A:

Programmet som vi ønsker at oversætte skal så ligge i diskettestationen til højre på panelet. Den er mærket med B:.

Disketten i B: med programmet der skal oversættes bør have rigelig plads til rådighed, da der i løbet af oversættelsen bliver oprettet fire midlertidige filer på denne diskette. Disse filer kan være af en ret anseelig størrelse.

Selve brugerprogrammet vil ikke ligge i lageret under oversættelsen, men vil blive hentet frem fra diskette efterhånden som oversættelsen skrider frem. Det betyder at der kan oversættes programmer som er langt større end den plads der er til rådighed i lageret. Da oversætteren arbejder rekursivt vil det betyde at komplicerede aritmetriske udtryk vil bruge temmeligt meget plads i lageret.

Vi forestiller os, at vi har lavet et COMAL-80 program, som f.eks. kan se således ud:

```
0010 // testprogram
0020 //
0030 I#:=1; STOPVÆRDI#:=10; TAB:=10
0040 WHILE I#<=STOPVÆRDI# DO
0050   I#:+1
0060   EXEC PRTPROC(I#)
0070 ENDWHILE
0080 PRINT "PROGRAM SLUT"
0090 //
0100 PROC PRTPROC(NUM#)
0110   PRINT "TRIN: ";NUM#,SIN(NUM#),COS(NUM#)
0120 ENDPROC PRTPROC
```

Dette program er udviklet under COMAL-80. Derefter ønsker vi at få det oversat til p-kode for at programmet kan afvikles uafhængigt af COMAL-80 fortolkeren. Når programmet er færdigt gemmes det på disketten med kommandoen

***LIST DK1:TEST**

Det betyder at programmet gemmes som en tekstfil i såkaldt ASCII format.

Disketten mærket med COMAL80-COMPILER sættes i diskstation A:, som ligger til venstre på panelet. Når operativsystemet er hentet op i lageret, og det sædvanlige klartegn er udskrevet:

A)PCOMAL COMPILE

PCOMAL systemet kaldes og compilermodulet lægges op i lageret. Kørslen startes og det ser således ud:

```
PCOMAL (V1.2) by Peter Ferdinand, (C) BOGIKA, Ikast, Denmark.
```

COMAL80-compileren kaldes som ovennævnt og hovedmenuen kommer på skærmen.

```
**************************************************************

          COMAL-80  COMPILER (version 1.2)
          by Peter Ferdinand.
          copyright (c) 1983  BOGIKA, Ikast, Denmark.

**************************************************************

  - compilation of a program                          (depress C)
  - printout of the code of a compiled program        (depress P)
  - STOP for execution                                (depress S)

  - parameters for the compilation:
     - PARAM: print of p-codes          (yes)         (depress T)
     - PARAM: generate NEWLINE p-codes  (yes)         (depress N)
     - PARAM: generate LINK module      (no )         (depress L)
Select desired function---------------------------------------)C

Type the name of the program (if nothing is typed, program
DK1:TEST is used) :

Where do you want the listing - on the printer    (depress P)
                              - or on the screen  (depress S) P
```

C vælges fra menuen og oversættelsen begynder. Der spørges først om filnavnet på COMAL-80 programmet. I dette tilfælde svares der blot ved at trykke 'return', fordi TEST allerede er blevet anvendt en gang før. Derefter spørges om udskrift ønskes til lineskriver (P) eller dataskærm (S). Herefter starter oversættelsen af programmet. Det sker igennem tre trin:

1) Leksikal analyse.
Programmet oversættes til intern kode og alle procedure - og funktionsnavne identificeres. Alle procedurer, funktioner og variable udskrives ved afslutningen af fasen.

2) Compileringsfasen.
Programmet oversættes til p-kode linie for linie. Det er muligt at få udskrevet programlinier og p-koder under oversættelsen. Det vælges i hovedmenuen.

3) Asembleringsfasen.
Alle de løse programmoduler fra de to foregående faser bliver samlet til et modul, som udskrives på diskette. Hvis der er externe procedurer eller funktioner som kaldes, så vil de blive angivet, således at disse kan lænkes sammen med hovedprogrammet. Ved afslutningen af denne fase, så vil hovedmenuen blive kaldt igen. Hvis der er blevet kaldt stadardfunktioner fra programmet som skal de lænkes sammen med hovedprogrammet i denne fase. Det biblioteksprogram som indeholder alle standardfunktioner og procedurer kaldes for COMLIB og ligger sammen med compileren på DK0: Under lænkningsfasen udskriver programmet følgende spørgsmål:

LIBRARY FILE:

Det er brugerens opgave at angive den biblioteksfil som kan løse de eksterne procedurekald i programmet. Som regel gøres det ved at bruge standardbiblioteksfilen COMLIB.

Hvis alt er sket som det er beskrevet, så burde oversættelsen være være gennemført uden fejl. Hvis der har været fejl i programmet, så vil oversættelsen stoppe øjeblikeligt med angivelse af en fejlkode, som kan findes i afsnit 3.2. Der vendes tilbage til hovedmenuen. Man bør være opmærksom på at syntaksfejl kun kan opstå, hvis programmet ikke har været afprøvet tilstrækkeligt under COMAL-80.

Derimod kan der opstå fejl som skyldes de bestemte forhold hvorunder en oversætter arbejder. F.eks. fejl i definitionen af variable, eller fejl hvis der er brugt programelementer af speciel BASIC type (ON .... GOTO, GOSUB ...., GOTO linienummer, RETURN). Disse forskelle er nærmere beskrevet i afsnit 2.

Det er muligt at få udskrevet det oversatte program i p-kode format. Det gøres ligeledes fra hovedmenuen. Tryk P.

Programmet er nu oversat og kan afvikles uden brug af COMAL-80 fortolkeren under CP/M operativsystemet.

A)PCOMAL B:TEST

Herefter afvikles programmet TEST.

2. Forskelle mellem COMAL-80 og COMAL-COMPIL.

Der er visse forskelle mellem de to systemer selvom grundstammen er den samme. Det er dels forskelle der skyldes at det ene system er fortolkende mens det andet er oversættende, og dels forskelle der skyldes at de sidste BASIC ting er pillet ud af COMAL80-compileren.

2.1 Programelementer som ikke findes i COMAL80-compileren.

De sidste BASIC levn er pillet ud af oversætteren. Det gælder alle referencer til linienumre. GOTO linienummer virker ikke og vil give en fejlmeddelelse. Denne sætningstype kan i stedet erstattes med GOTO label, hvor label er defineret i en LABEL sætning senere eller tidligere i programmet.

De andre programelementer som ikke findes er:

ON ..... GOTO/GOSUB nummer, GOSUB nummer, RETURN, GOTO nummer.

I oversætteren kan der anvendes variabel- og procedurenavne som er af enhver længde, men det er kun de første otte tegn som bliver anvendt. Det betyder at navne som DISKSEKTORET og DISKSEKTORTO vil blive opfattet som det samme navn.

Funktionen ERRTEXT$ findes af gode grunde ikke i PCOMAL, da fejltekster ikke vil blive udskrevet under afviklingen.

Når der kaldes maskinprogramdele fra COMAL-80 programmet, så må man sikre sig at Z-80 registrene HL', DE', IX og IY bliver tilbageleveret i samme stand som de blev modtaget.

Sætningen END skal være den sidste sætning i et program. Den angiver den fysiske slutning på programmet. Dog behøver den ikke at være med.

2.2 Forskelle i variables rækkevidde (deres 'scope')

Den vigtigste ændring i forhold til COMAL-80 fortolkeren er den måde, hvorpå variables rækkevidde defineres (deres 'scope').

I COMAL-80 bliver variables rækkevidde bestemt under afviklingen af et program, det betyder at alle variable fundamentalt er globale med mindre de defineres i en lukket procedure eller i en funktion, hvor de så er lokale. De lokale variable kan dog gøres globale ved hjælp af sætningen GLOBAL. Det er altsammen udmærket indtil man kalder en åben procedure fra en lukket procedure. Så bliver alle variable pludseligt lokale til den lukkede procedure. Det betyder reelt, at man ved blot noget komplicerede programstrukturer må vælge enten at lukke alle procedurer eller at have dem åbne allesammen. Det er temmeligt uhensigtsmæssigt, hvis man udvikler blot noget komplicerede programmer.

For at komme over dette problem - og for at tilnærme rækkeviddebegrebet til normal PASCAL tradition - så vil variables

rækkevidde i COMAL80-compileren bestemmes i oversættelsesfasen.

Det betyder at en variabel som fysisk bliver defineret (får tildelt en værdi) i starten af et program vil leve som en global variabel resten af programmet igennem:

```
0010 A#:=100
0020 -------
0030 PROC P1
0040   PRINT A#
0050 ENDPROC P1
0060 EXEC P1
0070 -------
```

A# vil udskrives med 100 og vil altid forblive en global variabel.

Det eneste der kan gøre en global variabel til en lokal er at bruge den som parameter i et procedurekald:

```
0010 A#:=100
0020 PROC P1(A#)
0030   PRINT A#
0040 ENDPROC P1
0050 EXEC P1(500)
0060 ---------
```

Her vil udskriften blive 500.
Hvis variablen blive defineret (tildelt en værdi) indeni en procedure, så vil denne variabel kun eksistere som lokal inden i denne procedure og ikke udenfor.

```
0010 PROC P1
0020   A#:=200
0030 ENDPROC P1
0040 PRINT A#
```

Her vil der blive udskrevet en fejl ved oversættelsen af programmet, da A# ikke er defineret i hovedprogrammet og derfor kun eksisterer som en lokal variabel.
Konsekvensen er, at man må sørge for at alle variable som man ønsker skal være globale, må defineres før de anvendes. Det skal senest ske før det første underprogram kald, hvor de anvendes som globale variable.

Det samme gør sig gældende med dimensionerede variable. Hvis de dimensioneres i hovedprogrammet, så er de definerede som globale, men når de dimensioneres i et underprogram, så er de definerede som lokale.

2.3 Inklusion af programdele.

Det er muligt at inkludere programdele fra en COMAL-80 tekstfil i sit program. Tekstfilen må gemmes med LIST og den må ikke afsluttes med en END sætning. Den indlæses i programmet i oversættelsesfasen ved at angive en udvidelse til REM sætningen:

```
0010 //$INCL filnavn     (f.eks. DK1:REFLIB)
```

hvor filnavn er navnet på en textfil med COMAL-80 kode. Det er vigtigt at sætningen ser fuldstændig ud som ovennævnte, da der ellers ikke vil ske noget (sætningen opfattes så som en REM). Det betyder at der også skal forekomme kun et blanktegn efter kommandoen INCL.

2.4 Externe procedurer og funktioner.

Det er muligt at kalde externe procedurer og funktioner, som er blevet oversat ved en tidligere lejlighed. Man kan på denne måde lave sit eget programbibliotek med oversatte programmer, som så kan lænkes sammen med et hovedprogram. Den externe procedure eller funktion skal defineres i hovedprogrammet, og det gøres således:

```
0010 //$EXTER PROC P4(A#,REF B)
0029 //$EXTER DEF FNPOP#(ADR#)
```

Disse externe procedurer kan nu kaldes på samme måde som alle andre procedurer der er defineret i programmet. Man bør dog være omhyggelig med at antallet og typen af parametre svarer ganske nøje til proceduren i det externe modul. Der sker ikke en typecheck under afviklingen og lænkningen af programmet.
De to procedurer kan så kaldes på følgende måde:

```
0080 EXEC P4(23,ASD)

0100 A#:=FNPOP#(234)
```

Se nærmere eksemplet i afsnit 4.

2.5 Filbehandling.

Ved selve oversættelsen er der ingen forskelle på anvendelsen af filer mellem COMAL-80 og COMAL80-compiler. Specifikationen af filer syntaktisk er fuldstændigt sammenfaldende mellem de to systemer.

Hvis anvendelsen af filer udelukkende sker indenfor et enkelt af systemerne så vil brugerne ikke mærke nogen forskelle overhovedet i organiseringen af filer på disketten. D.v.s. at en fil oprettet indenfor PCOMAL systemet uden problemer kan læses og anvendes af et andet program som også bliver afviklet under PCOMAL systemet. Denne anvendelse er fuldstændigt som specificeret i COMAL-80 brugervejledningen.

Problemer kan opstå når man ønsker at benytte filer under PCOMAL, som er oprettet og skrevet under COMAL-80. Det vil kunne lade sig gøre i visse tilfælde, men i andre vil man enten få en fejlmeddelelse eller man vil få fejlagtige resultater. Nedenfor er vist i hvilke det vil kunne lade sig gøre umiddelbart og i hvilke det vil lade sig gøre ved mindre ændringer af programmet.

Filer oprettet og skrevet med COMAL-80.	Filer læses af COMAL80 programmer under PCOMAL.

Filer med SEKVENTIEL/DIREKTE tilgang - med ASCII tegn

F.eks.

```
PRINT FILE 0: A, A#, A$          INPUT FILE 0: A, A#, A$
```

I dette tilfælde er anvendelsen helt ensartet og filer oprettet
under COMAL-80 kan uden videre læses af PCOMAL.

filer med SEKVENTIEL/DIREKTE tilgang - med binære tegn

F.eks.

```
WRITE FILE 0: A, A#, A$          READ FILE 0: A, A#, A$
```

I dette tilfælde er anvendelsen også helt ensartet og filer oprettet under COMAL-80 kan uden videre læses af PCOMAL.

SEKVENTIEL/DIREKTE tilgang - binære tegn - hele matricer

F.eks.

```
DIM A#(20,20)
MAT A#=0
WRITE FILE 0: A#
```

Denne type kan ikke umiddbart læses fra PCOMAL. Man må indlæse enkeltelementer separat.

F.eks.

```
DIM A#(20,20)
FOR I#=1 TO 20
  FOR J#=1 TO 20
    READ FILE 0: A#(I#,J#)
  NEXT J#
NEXT I#
```

3. Nærmere gennemgang af oversættelsen.

I dette afsnit vil det i detaljer blive gennemgået, hvorledes oversættelsen typisk kan forløbe for et program. Vi vil bruge program TEST fra indledningen, som det gennemgående eksempel. Oversættelsen kan startes med tre forskellige parametre.

1. (T) 'Print of p-codes' (se afsnit 5.)
Der udskrives p-coder på enten skærm eller på lineskriver efterhånden som oversættelsen skrider frem. Hvis p-coder ikke ønskes, så vil kun kildeprogrammet blive udskrevet.

2. (N) 'Generate NEWLINE p-codes'
Der udskrives en p-code med et linenummer for hver ny COMAL linie som behandles af oversætteren. Der kan spares ikke så lidt plads ved at udelade disse koder i det færdigt oversatte program. Fordelen ved at medtage disse koder er at man ved programafviklingen får oplyst i hvilken linie programmet er stoppet ved en fejl.
Det er kun muligt at afbryde det færdige program under afviklingen, hvis NEWLINE p-koderne er medtaget. Rent praktisk sker det således at PCOMAL afviklingssystemet spørger om der ligger et ESC-tegn på tastaturet, hvis det er tilfældet, så afbrydes afviklingen. Tegnet vil samtidigt blive lagt ned i lageradresse 256.

3. (L) 'Generate LINK module' (se afsnit 4.)
Oversættelsen vil fremstille et biblioteksmodul, som senere kan lænkes sammen med et hovedmodul. Et biblioteksmodul er magen til et ikke assembleret hovedmodul, der ikke indeholder globale variable overhovedet. Der må altså ikke forekomme globale variable i et biblioteksmodul.

3.1 Leksikal analyse.

I denne fase oversættes COMAL programmet til intern kode (som ikke er p-kode). Alle variable og underprogrammer identificeres og bliver udskrevet på enten skærm eller printer. DATA-sætninger og andre konstanter bliver udskildt fra programmet. Følgende udskrift kan forekomme:

```
LEXICAL ANALYSIS OF COMAL-80 PROGRAM FROM FILE: DK1:TEST.CML

FIRST LOOP
..............
SECOND LOOP


******   TABLE OF VARIABLES  ******
 NO  VAR       TYP  I NO  VAR       TYP  I NO  VAR       TYP  I
-------------------I-------------------I-------------------I
 31  NUM        V   I 41  STOPVÆRD   V   I 14  I          V   I
```

```
******   TABLE OF PROCEDURES   ******
 NO  VAR        TYP  I NO  VAR        TYP  I NO  VAR        TYP  I
-----------------------I-----------------------I-----------------------I
 95  PRTPROC     RP  I 78  SIN         RX  I 62  COS         RX  I
```

Herefter udskrives samtlige biblioteksprocedurer.

Denne fase gennemløber programmet i to løkker. Disse udskrives på terminal/linieskriver efterhånden som løkkerne begynder. Derefter udskrives der succesivt et punkt (.) for hver programlinie som indlæses.

Der er benyttet tre variable NUM, I og STOPVÆRD. Det tre variable har typen V, som betyder heltal. Den næste tabel viser alle de procedurer som er anvendt i programmet. I dette tilfælde er der en brugerdefineret procedure i programmet (PRTPROC) og to externe procedurer(SIN og COS) som er af typen REAL.

Der kan forekomme følgende typer af variable:

V	Simple og indicerede heltalsvariable.
R	Simple og indicerede reelle variable.
S	Indicerede streng variable.

Der kan forekomme disse typer af underprogrammer:

VX	Standard extern funktion, heltalsresultat.
RX	- do - reelt resultat.
SX	- do - streng resultat.
VF	Funktion med heltalsresultat.
RF	Funktion med reelt resultat.
RP	Brugerdefineret procedure.
RL	Navn på en label
RS	Standardsætning som ligner et procedurekald uden anvendelse af paranteser. F.eks. INIT "DK1:"

De numre der er angivet har intet med variblens placering at gøre, men er udelukkende en intern nummerering.

3.2 Oversættelsesfasen.

Oversættelsesfasen oversætter COMAL-80 programmet fra den interne kode til egentlig p-kode. Disse p-koder kan udskrives samtidigt med at programmet oversætter. Der kan i denne fase forekomme en række fejlkoder. Udskriften fra oversættelsen kan f.eks. se således ud, hvis udskriften af p-koder er slået fra.

```
COMPILATION OF COMAL-80 PROGRAM FROM FILE:      DK1:TEST.CML

  10     12  // testprogram
  20     15  //
  30     18  I#:=1; STOPVÆRDI#:=10; TAB:=10
  40     44  WHILE I#(=STOPVÆRDI# DO
  50     60    I#:+1
  60     71    EXEC PRTPROC(I#)
  70     80  ENDWHILE
  80     89  PRINT "PROGRAM SLUT"
  90     96  //
 100     99  PROC PRTPROC(NUM#)
 110    107    PRINT "TRIN: ";NUM#,SIN(NUM#),COS(NUM#)
 120    145  ENDPROC PRTPROC

codelgt: 158   constantlgt: 22   Variabellgt:12  datalgt :0

*****  NORMAL END OF COMPILATION  *****
```

Den første kolonne i programlinierne er linenummeret i COMAL-80 programmet, mens den anden kolonne er det antal bytes som hidtil er blevet brugt til p-koder. Hvilket ikke er det samme som antallet af p-koder, da disse kan være længere end en byte. Der afsluttes med at angive længden i bytes af de forskellige programdele.

I begyndelsen af programmet er der hopordrer til interne såvel som eksterne procedurer. Disse adresser vil blive løst i den efterfølgende fase kaldet lænkning.

Fejlkoderne fra denne fase er noget primitive. De består fundamentalt kun af et nummer og linenummeret på den sidst indlæste linie, hvilket ikke behøver at være den linie hvori fejlen er opstået. Det har ikke været anset for nødvendigt at udvikle fejlsøgningen mere, da selve programudviklingen altid vil ske i COMAL-80 systemet, hvor fejlfindingen er meget fin. Oversættelsen vil stoppe når blot en fejl bliver fundet.

Fejlkode	Beskrivelse
1	STAK OVERLØB
2	STAK UNDERLØB
3	STAK OVERLØB I ARGUMENTSTAK VED PROCKALD
4	TYPEKONFLIKT I UDTRYK
5	TYPEKONFILKT I UDTRYK (IKKE STRENGUDTRYK)
6	IDENTIFIER IKKE DEFINERET
7	INGEN VENSTRE PARANTES I FAKTOR
8	SYMBOL IKKE KENDT I FAKTOR
9	TYPEKONFLIKT I TERM
10	UKENDT OPERATOR I UDTRYK
11	UDEFINERET TYPE I TILDELINGSSÆTNING
12	TYPEN IKKE KENDT FOR TILDELINGSSÆTNINGEN
13	:= MANGLER
14	'THEN' MANGLER I IF SÆTNING
15	'DO' MANGLER I WHILE SÆTNING
16	PROC/DEF IKKE DEFINERET

```
17        VENSTRE PARANTES MANGLER I 'EXEC'
18        PROCEDURENAVN I 'EXEC' IKKE DEFINERET
19        TYPEKONFLIKT VED PROCEDUREKALD
20        HVERKEN KONSTANT ELLER VARIABELNAVN I KALD
21        HØJRE PARANTES MANGLER
22        GALT ANTAL AF PARAMETRE I PROCEDUREKALD
23        IDENTIFIER FORVENTET I INPUTSÆTNING
24        IKKE DEFINERET VARIABEL BRUGT I INPUTSÆTNING
25        IDENTIFIER FORVENTET I DIM-SÆTNING
26        HELTALSKONSTANT FORVENTET
27        VENTSRE PARANTES MANGLER I DIM-SÆTNING
28        'OF' MANGLER I STRENG DIMENSIONERING
29        HELTALSKONSTANT FORVENTET I STRENGDIM.
30        IDENTIFIER FORVENTET I FOR- SÆTNING
31        TO/DOWNTO FORVENTET I FOR-SÆTNING
32        TYPEKONKLIKT I FOR-SÆTNING
33        'DO' FORVENTET I FOR-SÆTNING
34        'OF' FORVENTET I CASE-SÆTNING
35        'WHEN' FORVENTET EFTER CASE-SÆTNING
36        ENDCASE MANGLER
37        FEJL I SÆTNING - FEJLSÆTNING
38        IDENTIFIER FORVENTET I PROC/FUNC-SÆTNING
39                  -  DO  -
40        PARAMETER DEKLARATION GAL
41        MANGLER VENSTRE PARANTES I PROC/FUNC
42        MANGLER IDENTIFIER I ENDPROC/ENDDEF
43        GALT SKILLETEGN I INDEX
44        VENSTRE PARATES MANGLER I INDEX
45        NUMERISK KONSTANT FORVENTET I DIM
46        ':' MANGLER I INPUT/READ SÆTNING
47        VENSTRE PARATES MANGLER I TAB
48        HØJRE PARANTES I TAB MANGLER
49        GALT SKILLETEGN I PRINT-SÆTNING
50        ':' MANGLER I INPUT-SÆTNING
51        ':=' MANGLER I TAB-SÆTNING
55        INGEN IDENTIFIER I LABEL-SÆTNING
56        INGEN LABEL-IDENTIFIER I GOTO-SÆTNING
57        ':' MANGLER I PRINT USING
58        ',' MANGLER I RND(  ,  ) FUNKTION
59        UDEFINERET INDENTIFIER I GLOBAL-SÆTNING
60        IKKE EN INDICERET VARIABEL I MAT-SÆTNING
61        FEJL I TRAP-SÆTNING
62        TYPEKONFLIKT I ARITMETRIK UDTRYK
63        PLADS TIL VARIABLE ER OPBRUGT
```

3.3 Assemblering og lænkning.

I denne sidste fase samles alle de enkelte stumper sammen til et enkelt modul. Adressen på de enkelte programdele udskrives. Det angives ligeledes, hvis der findes externe procedurer eller funktioner som skal sammenlænkes med hovedmodulet. Det kan f.eks. se således ud:

```
ASSEMBLING AND LINKING OF COMAL-80 PROGRAM :DK1:TEST.CML

DATAADR: 160   CONSTANTADR: 160   TOTALLGT: 182

  - unsolved external references:
COS             3  I SIN             6  I

  - internal procedures and functions:
PRTPROC       107  I

LIBRARY FILE: COMLIB


DATAADR: 190   CONSTANTADR: 190   TOTALLGT: 212

  - unsolved external references:
NONE

  - internal procedures and functions:
PRTPROC       107  I SIN           159  I COS           174  I
```

I lænkningsfasen bliver programmet ved med at spørge om nye biblioteksmoduler så længe der endnu er uløste eksterne referencer. I dette tilfælde var der to eksterne funktioner SIN og COS, som var kaldt fra hovedprogrammet. De blev sammenlænket med hovedprogrammet ved at kalde standard biblioteksfilen (COMLIB). Da det var sket var der ikke flere referencer at løse.

Under lænkningen gives der oplysninger om programmets totale størrelse i bytes. Desuden oplyses det, hvor arealet for DATA sætninger starter og hvor arealet for programmets konstanter starter. I et biblioteksmodul er det muligt at definere konstanter, men ikke datasætninger. De adresser som er angivet efter navnet på proceduren betyder for externe procedurer den adresse, hvorfra proceduren kaldes, mens adressen for interne procedurer er den aktuelle adresse i programmet.

Det bør bemærkes at der ikke sker et typecheck ved sammenlænkningen af programmer og bibliotek.

4. Fremstilling af biblioteksmodul

Et biblioteksmodul er en række procedurer og funktioner som lever deres selvstændige liv. De oversættes en gang for alle og kan derefter anvendes af alle programmer fremover. En sådan biblioteksmodul kan opfattes som en række af småprogrammer som et hovedprogram frit kan vælge imellem. Biblioteksmodulet kan senere sammenlænkes med et hovedmodul som så kan afvikles umiddelbart under PCOMAL. Standardbiblioteksprogrammet COMLIB er bl.a. et sådan biblioteksmodul.

Der er specielle regler for oversættelsen af biblioteksmoduler.

Der må ikke forekomme globale variable i et biblioteksmodul. Det betyder at man bør lukke alle procedurer med ordren CLOSED, og at sætningen GLOBAL ikke kan anvendes. Ligeledes må der ikke forekomme programlinier som ligger udenfor proceduredefinitionerne.

Der må ikke forekomme DATA sætninger i et biblioteksmodul, men gerne andre konstanter i tildelingssætninger.

Hvis der indenfor biblioteksmodulet findes procedurer som kalder andre procedurer indenfor det samme biblioteksmodul, så skal de kaldte procedurer defineres før de bliver kaldt. Hvis det ikke sker så er man nødsaget til at lænke det samme modul to eller flere gange til hovedmodulet.

Nedenfor vises et eksempel på en række procedurer som vi ønsker at lave til en biblioteksfil. Programmet ser således ud når det bliver oversat af COMAL-COMPIL:

```
COMPILATION OF COMAL-80 PROGRAM FROM FILE:      DK1:REFLIB.CML

  10    18  // PROCEDURES AND GENERATION OF LIBRARYMODULE
  20    21  // REFERENCE
  22    24  //
  30    27  PROC REFER5(REF I#) CLOSED
  40    36    I#:=999
  50    49  ENDPROC REFER5
  60    56  //
  70    59  PROC REFER4(REF I#) CLOSED
  80    62    EXEC REFER5(I#)
  90    72  ENDPROC REFER4
 100    81  //
 110    84  PROC REFER3(REF I#) CLOSED
 120    87    EXEC REFER4(I#)
 130    97  ENDPROC REFER3
 140   106  //
 150   109  PROC REFER2(REF I#) CLOSED
 160   112    EXEC REFER3(I#)
 170   122  ENDPROC REFER2
 180   131  //
 190   134  PROC REFER1(REF I#) CLOSED
 200   137    EXEC REFER2(I#)
 210   147  ENDPROC REFER1
 220   156  //

codelgt: 161   constantlgt: 0   variablelgt:8  datalgt :0
```

```
*****  NORMAL END OF COMPILATION  *****

CREATE LIBRARY OF COMAL-80 PROGRAMS FROM: DK1:REFLIB.CML
************************************************************

DATAADR: 163   CONSTANTADR: 163   TOTALLGT: 163
```

Dette program er nu gemt på diskette DK1: under navnet REFLIB. Det er et bibliotheksprogram som kan lænkes til ethvert hovedmodul. Denne sammenlænkning er vist i det næste eksempel. De procedurer som skal kaldes må defineres i programmet som eksterne procedure ved anvendelsen af sætningen :

```
        //$EXTER PROC NAME(ARG1, REF ARG2)

COMPILATION OF COMAL-80 PROGRAM FROM FILE: DK1:EXECREF.CML

  10      9  //$EXTER PROC REFER1(REF I#)
  20     12  //$EXTER PROC REFER2(REF I#)
  30     15  //
  40     18  FOR VAR1#:=1 TO 25 DO
  50     46    VAR2#:=0
  60     55    EXEC REFER1(VAR2#)
  62     62    PRINT VAR2#
  70     69    EXEC REFER2(VAR2#)
  72     81    PRINT VAR2#
  80     88  NEXT VAR1#
  90    107  END

codelgt: 112   constantlgt: 0   variables:12  datalgt :0

*****  NORMAL END OF COMPILATION  *****

ASSEMBLING AND LINKING OF COMAL-80 PROGRAM :DK1:EXECREF.CML

DATAADR: 114   CONSTANTADR: 114   TOTALLGT: 114

  - unsolved external references:
REFER2          3  I REFER1          6  I

  - internal procedures and functions:
NONE

LIBRARY FILE: DK1:REFLIB

DATAADR: 210   CONSTANTADR: 210   TOTALLGT: 210

  - unsolved external references:
NONE

  - internal procedures and functions:
REFER1        113  I REFER2        132  I REFER3        151  I
REFER4        170  I REFER5        189  I
```

5. Udskrivning af p-koder.

Dette afsnit kan overspringes, da det kun beskriver den interne kode som benyttes af afviklingssystemet. Forståelsen af denne kode er ikke nødvendig for at anvende systemet i sin helhed.

Her vises en oversigt over, hvorledes lageret er disponeret under afviklingen af de oversatte programmer.

adresse i lageret	indholdet
øverste lageradresse	
	CP/M operativsystemet
GLOBALADR:	----------
	globale variable
BASEADR:	----------
	lokale variable og stak (vokser nedad)
	(vokser opad)
HEAPADR:	midlertidige konstanter og strengvariable
HEAPBASE:	----------
	konstanter
CONSTANTADR:	----------
	datakonstanter og pegepinde til konstanter.
DATAADR:	----------
	P-kodeprogram
PROGRAMADR:	----------
	P-COMAL afviklingssystem.
nederste lageradresse	

P-koderne kan udskrives enten på lineskriver eller på skærmen.
Det vælges i hovedmenuen. Det kan f.eks. se således ud:

```
COMPILATION OF COMAL-80 PROGRAM FROM FILE:   DK1:TEST.CML

                        0     5  JMP      0
COS                     3     5  JMP      -62
SIN                     6     5  JMP      -78
PRTPROC                 9     5  JMP      -95
                                        ADD AT 0  CHANGED TO 12
  10     12  // testprogram
                       12     2  NEW      10
  20     15  //
                       15     2  NEW      20
  30     18  I#:=1; STOPVÆRDI#:=10; TAB:=10
                       18     2  NEW      30
                       21     4  ENT      6
                       24     5  JMP      27
** INIT **8
                       27     0  GLO      -8
                       30     9  LCOV     1
                       33    12  STOV
** INIT **10
                       34     0  GLO      -10
                       37     9  LCOV     10
                       40    12  STOV
                       41     9  LCOV     10
  40     44  WHILE I#(=STOPVÆRDI# DO
                       44     2  NEW      40
                       47   120  TABB
                       48     0  GLO      -8
                       51    15  LODV
                       52     0  GLO      -10
                       55    15  LODV
                       56    60  LEV
                       57     6  JPC      0
  50     60     I#:+1
                       60     2  NEW      50
                       63     0  GLO      -8
                       66    81  COPV
                       67    15  LODV
                       68     9  LCOV     1
  60     71     EXEC PRTPROC(I#)
                       71     2  NEW      60
                       74    24  ADDV
                       75    12  STOV
                       76     0  GLO      -8
                       79    15  LODV
  70     80  ENDWHILE
                       80     2  NEW      70
                       83     7  CAL      9
                       86     5  JMP      48
                                    ADD AT 57  CHANGED TO 89
  80     89  PRINT "PROGRAM SLUT"
                       89     2  NEW      80
                       92   110  CON
                       93    10  LCOS     1
```

```
 90     96  //
                      96     2  NEW      90
100     99  PROC PRTPROC(NUM#)
                      99     2  NEW      100
                     102    76  OUTS
                     103   101  NLN
                                   ADD AT 21  CHANGED TO 12
                     104     5  JMP      0
                                   ADD AT 9  CHANGED TO 107
110    107    PRINT "TRIN: ";NUM#,SIN(NUM#),COS(NUM#)
                     107     2  NEW      110
                     110     4  ENT      6
                     113   110  CON
                     114    10  LCOS     15
                     117    76  OUTS
                     118     1  LOC      2
                     121    15  LODV
                     122    75  OUTV
                     123   102  NTB
                     124     8  INT      4
                     127     1  LOC      2
                     130    15  LODV
                     131    99  FLT
                     132     7  CAL      6
                     135    77  OUTR
                     136   102  NTB
                     137     8  INT      4
                     140     1  LOC      2
                     143    15  LODV
                     144    99  FLT
120    145  ENDPROC PRTPROC
                     145     2  NEW      120
                     148     7  CAL      3
                     151    77  OUTR
                     152   101  NLN
                                   ADD AT 110  CHANGED TO 8
                     153     3  RTN      2
                                   ADD AT 104  CHANGED TO 156
                     156   115  STP

codelgt: 157   constantlgt: 22   variablelgt:12  datalgt :0

*****  NORMAL END OF COMPILATION  *****
```

Det følgende er en oversigt over alle de p-koder som er defineret i systemet og som forstås af PCOMAL afviklingssystemet. P-koderne kan enten være en byte eller tre bytes lange. De første 12 er af typen tre bytes, mens resten kun er en byte lange. Hvis p-koden ender på V så er det en heltalskode (to bytes), hvis den ender på R er det en reel kode (OTTE bytes) og hvis den ender på S, så er den en streng koder, hvor længden ligger i lageret. SP betyder stakpointer, ^SP betyder første element på stakken, GLOBAL betyder displacement fra værdien af GLOBAL, Base betyder displacement fra værdien af BASE, som er pegepinden til variabeldelen af den nuværende procedure, Program betyder displacement fra værdien af programtælleren. Constant betyder displacement fra værdien af constant adressen. Disse P-koder er meget lig de koder som Brinch Hansen har benyttet til konstruktionen af sin sekventielle PASCAL compiler. (Brinch Hansen: The Architecture of Concurrent Programs, New Yersey 1977). Grundlaget er omvendt polsk notation, hvor der benyttes en stak til alle operationer.

F.EKS. vil en sætning som oversættes til omvendt polsk notation se ud på denne måde:

```
VAR1#:=(VAR2# * 100) + VAR3#

GLO      VAR1     adressen på VAR1 på STAK
GLO      VAR2     adressen på VAR2 på STAK
LODV              VÆRDIEN PÅ STAK
LCOV     100      heltalskonstant på STAK
MULV              multiplikation af to øverste stak
GLO      VAR3     adressen på VAR# på stakken
LODV              værdien på stak
ADDV              addition af to øverste stak
STOV              gemmer øverste element i adressen
                  på næste.
```

p-kode		evt.argument	beskrivelse
0	GLO	-Global	Globale adresse
1	LOC	-Base	Lokale adresse
2	NEW	LINIENR	Linienummeret gemmes
3	RTN	-PARAM	Retur fra procedure, param i bytes
4	ENT	VARS	Indgang til procedure, lokale vars
5	JMP	Program	Programhop
6	JPC	Program	Betinget programhop, hvis ^St falsk
7	CAL	Program	Procedurekald
8	INT	BYTES	St forøges med værdien af BYTES
9	LCOV	CONSTVAL	Value of CONST lægges på St
10	LCOS	Constant	Strengadr lægges på St
11	LCOR	Constant	Reelt tal lægges på St
12	STOV		^St gemmes i variablen, heltal
13	STOS		- , streng
14	STOR		- , reelt tal
15	LODV		Variablen på ^St, heltal
16	LODS		- , streng
17	LODR		- , reelt tal
18	DIMV		Dimensionering af indiceret variabel
19	DIMS		Dimensionering af indiceret streng
20	DIMR		Dimensionering af indiceret reelt

21	NEGV	Negate ^St, heltal
22	STUS	Gem 'print using' buffer
23	NEGR	Negate ^St, real
24	ADDV	Adder to øverste elementer af St
25	ADDS	Sammenlæg to strenge til en streng
26	ADDR	adder to reele tal, resultat til ^St
27	SUBV	Subtraher to heltal, result. til ^St
29	SUBR	Subtraher to reelle tal
30	MULV	Multiplicer to heltal
32	MULR	Multiplicer to reelle tal
36	MODV	Modulo af to heltal
38	MODR	Modulo af to reelle tal
39	DIVV	Heltalsdivision
40	NLIN	indlæser indtil end-line mødes
41	DIVR	Division af to reelle tal
42	CODE	P-koder slutter
43	CONI	INPUT fra standard consol
44	POWR	^St+1 opløftet i ^St potens, reelle
45	EQV	^St+1 = ^St resultat ^St, heltal
46	EQS	– , streng
47	EQR	– , reelt
48	NEV	^St+1 ()^St resultat ^St, heltal
49	NES	– , streng
50	NER	– , reelt
51	LTV	^St+1 (^St resultat ^St , heltal
52	LTS	– , streng
53	LTR	– , reelt
54	GEV	^St+1)=^St resultat ^St , heltal
55	GES	– , streng
56	GER	– , reelt
57	GTV	^St+1) ^St resultat ^St , heltal
58	GTS	– , streng
59	GTR	– , reelt
60	LEV	^St+1(=^St resultat ^St , heltal
61	LES	– , streng
62	LER	– , reelt
63	ORV	^St+1 or ^St result. ^St, heltal
65	ORR	– , reelt
66	ANDV	^St+1 and^St sesult. ^St, heltal
67	TABB	sætter TAB værdien
68	ANDR	– , reelt
69	INV	input heltal til ^St, ASCII
70	INS	input streng til ^St, ASCII
71	INR	input reelt til ^St , ASCII
72	INBV	input heltal til ^St, binær
73	INBS	input streng til ^St, binær
74	INBR	input reelt til ^St , binær
75	OUTV	output heltal fra ^St, ASCII
76	OUTS	output streng fra ^St, ASCII
77	OUTR	output reelt fra ^St, ASCII
78	OUBV	output heltal fra ^St, binær
79	OUBS	output streng fra ^St, binær
80	OUBR	output reelt fra ^St, binær
81	COPV	copy ^St til ^St-1, heltal
82	RND2	random funktion med to argumenter.
83	COPR	– , reelt
84	INCV	incremetere ^St, heltal
85	NOTR	logisk NOT operation, reelt
86	INCR	– , reelt
87	DECV	decrementere ^St, heltal

89	DECR	- , reelt
90	DCOV	næste dataelem ^St, heltal
91	DCOS	- , streng
92	DCOR	- , reelt
93	INXV	beregning af index, heltal
95	INXR	beregning af index, reelt
96	IND	indirekte adressering
97	LODD	delsteng til ^St
98	STOD	^St til delstreng
99	FLT	heltal til reelt tal
100	NOT	not ^St
101	NLN	'eol' for output
102	NTB	'nexttab' for output
103	ORD	'open' for 'read'
104	OWR	'open' for 'write'
105	ORND	'open' for 'random' tilgang
106	TAB	tabulator for output
107	CLA	'close' alle
108	CLF	'close' bestemt fil
109	FIL	filkode for de næste i/o operationer
110	CON	'output' til consol
111	SEL	select output
112	RST	'restore' data til start værdi
113	RLA	'restore' data til label
114	IN	^St IN ^St+1, strengoperation
115	STP	stop
116	DEL	delete file
117	CHA	chain file
118	CNV	reelt tal til heltal
119	HADS	adr. på strengresultat i 'heap',str
120	HADR	- ,reelt
121	OUUV	print using, heltal
122	OUUS	print using, streng
123	OUUR	print using, reelt
124	RFIL	randomfil nummer og recordnr.
125	MATV	matrixtildeling, heltal
126	MATS	- , streng
127	MATR	- , reelt
128	EOF	returnerer EOF kode
129	RND	tilfældigt tal
130	RAND	randomize
131	TERR	TRAP err, ^St
132	TESC	TRAP esc, ^St
133	STOB	Gem byte i ^St
134	FESC	ESC funktionskald
135	FERR	ERR -
136	LODB	hent byte fra ^St
137	SIN	sinusfunktion
138	COS	cosinus funktion
139	TAN	tangentfunktion
140	ATN	arc. tangent
141	LOG	naturlige logaritme
142	EXP	exponent
143	SQR	kvadratrod
144	INIT	initialise filsystem
145	CLR	blank skærm
146	CURS	cursor adresse
147	REN	'rename' af fil
148	INP	input fra port
149	OUTP	'output' til port

150	SYS	kald af maskinprogram
151	UNIT	set udfil til UNIT
152	GETU	returnerer UNIT
153	OUMV	'output' matrix, heltal
154	OUMS	– , streng
155	OUMR	– , reelt
156	INMV	'input' matrix , heltal
157	INMS	– , streng
158	INMR	– , reelt
159	INTR	heltalsdel af reelt tal
160	VALR	convertering af streng til reelt tal.
161	VALV	– heltal
162	STR	convertering at tal til streng
163	SPC	antal blanktegn angives
164	ABSV	absolut værdi af heltal
165	SGN	fortegnsfunktion
166	ABSR	absolut værdi af reelt tal

En nærmere definition af disse koder er det ikke stedet til her. Der er blot tænkt på de brugere som er særligt interesse-rede.

6. Afviklingssystemet.

Selve afviklingen af programmet sker uden at anvende COMAL-80. Det sker ved at kalde en p-kode fortolker, som efterhånden vil fortolke alle p-koder i programmet. Programmet der skal fortolkes skrives som parameter til p-kodefortolkeren, således:

```
A>PCOMAL  B:TEST
```

PCOMAL skal ligge på disketten i station A:, mens selve programmet der skal afvikles ligger i station B:. Der sker nu en kørsel af programmet TEST.
Følgende fejlkoder kan forekomme i afviklingssystemet:

1	al hukommelse opbrugt
2	ingen program af det navn
3	fil kan ikke åbnes
4	ikke EOF i p-kodefil
5	ikke gyldig p-kode
6	aritmetrisk over- eller underløb(w)
7	division med 0, heltal(w)
8	indexfejl
9	PCOMAL versioner uforenelige
10	en fil er allerede åbnet der
11	substreng tildeling fejlagtig
12	fejl i argument
13	fejl i filtype
14	fejl i filnavn
15	filtypen er gal
16	fil allerede åben her
17	typekonflikt i READ/DATA
18	fejl i filno
19	gal filtype
23	læsefil findes ikke
24	diskplads opbrugt ved oprettelse af fil
25	fil kan ikke lukkes
26	fil kan ikke læses
27	fil findes allerede med det navn
28	recordlængde i randomfil er fejlagtig
29	ikke en randomfil
30	fejl i sektornummer for randomfil
32	recordlængde er gal
40	division med 0, reelt
41	underløb, reelt tal
42	overløb, reelt tal
43	ulovligt argument
44	ulovligt argument
45	fejl i logaritmeberegning
46	fejl i SQR argument
47	gal filtype ved læsefil
48	fil kan ikke lukkes
49	'end of file' fundet
50	tabværdien gal i print
51	angivelse af type i UNIT gal
52	fejl i GETUNIT

7. Oversigt over alle programmer på disketten.

Da der er tale om et større programsystem, så er der hermed en oversigt over alle programmer som hører med til systemet.

PROGRAMFILER	FUNKTION
COMPILE	OPRETTER: COMOCOMM.DAT (Datafil parametre) Styreprogram for afviklingen af oversættelsen. FILNAVN opgives. Hovedmenu.
COMOLEXI	OPRETTER: 'filnavn'.LEX (intern kode) 'filnavn'.CST (konstanter og procedurer) ANVENDER: COMOLEXI.HLP (PROCS, NAVNE M.M.) COMOCOMM.DAT 'filnavn'.CML (kildekode) Oversætter et COMAL program til intern kode, identificerer variable og procedurer og lagre konstanter.
COMOCOMP	OPRETTER: 'filnavn'.PCD (P-kode) ANVENDER: COMOMN.MN (P-kode navne) 'filnavn'.CML 'filnavn'.CST 'filnavn'.LEX Oversætter den interne kode til egentlig P-kode. P-koden skrives på 'filnavn'.PCD.
COMOLIB	OPRETTER: libnavn.PSY (PROCS lænkemodul) libnavn.PLB (P-kode lænkemodul) ANVENDER: 'filnavn'.CST 'filnavn'.PCD Opretter et lænkemodul fra et program. Det kan senere lænkes sammen med et hovedmodul.

COMOASM

OPRETTER: 'filnavn'.COD (færdig P-kode)

ANVENDER: libnavn.PSY (lænkemodul PROCEDURE)
libnavn.PLB (lænkemodul P-koder)
'filnavn'.CST
'filnavn'.PCD

Samler alle de løse tråde til et enkelt modul som udskrives på fil. Standard biblioteksfilen anvendes som regel: COMLIB

COMOLIST

ANVENDER: 'filnavn'.COD

Programmet udskriver P-koder og programlinier, hvis 'newline' koder er angivet ved oversættelsen.

8. Litteraturfortegnelse

METANIC COMAL-80, user's manual
1981, Metanic, Denmark.

Poul Østergård,
Programmering i COMAL-80,
Teknisk Forlag, 1982.

David Gries,
Compiler Construction for Digital Computers,
1971, John Wiley & Sons, USA.

Thom Hogan,
Osborne CP/M User Guide,
1982, McGraw-Hill, USA.

Per Brinch Hansen,
The Architecture of Concurrent Programs,
1977, Prentice Hall, USA.

Nicklaus Wirth,
Algorithms + Data structures = Programs,
1976, Prentice Hall, USA.

Blaise W. Liffick (editor),
The BYTE Book of Pascal,1979,
A collection of articles from BYTE magazine,
Byte Publications, USA.

Roy Atherton,
Structured Programming With Comal,
Wiley & Sons, England, 1982.