H. Pakker

H.1 Introduktion

Pakker består af en række procedurer og funktioner, der egentlig er **maskinkode-underprogrammer**; men som fra RcComal80 ser ud og kan benyttes på præcis samme måde, som almindelige RcComal80 procedurer og funktioner.

Der er en række fordele ved at bruge pakker skrevet i maskinkode. Dels er de hurtige at udføre, og dels giver de adgang til flere faciliteter. Desuden giver de mulighed for at lave biblioteker og for at udnytte lageret bedre. Disse fordele uddybes i afsnittene H.1.1-H.1.4.

Til gengæld er det sværere at skrive og afteste maskinkode, og pakker skrevet i maskinkode vil derfor næppe blive skrevet af den "almindelige" RcComal80 bruger. Men enhver RcComal80 bruger vil kunne benytte pakker, skrevet af andre, i egne programmer. Bemærk: Man kan ikke rette i pakker fra RcComal80 - man kan kun bruge dem.

Dette bilag beskriver, hvorledes man bruger pakker, mens bilag I beskriver, hvorledes man programmerer pakker.

H.1.1 Hastighed

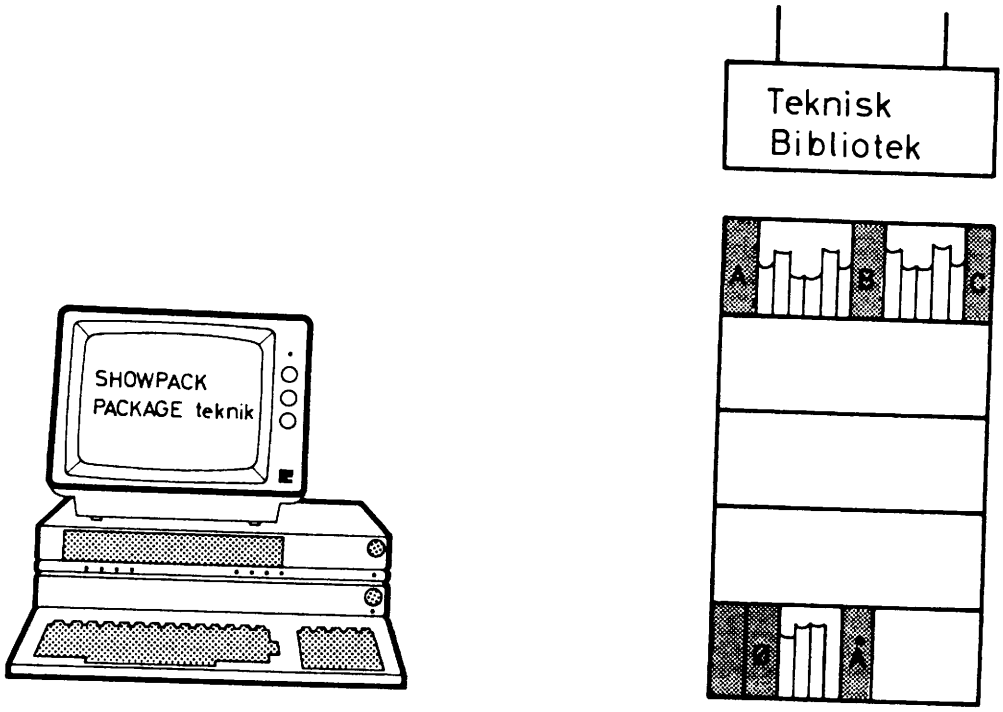
Maskinkode er **hurtigere** at udføre end RcComal80, fordi maskinkode er skrevet i maskinens eget sprog, og derfor kan udføres omgående, hvorimod et RcComal80-program fortolkes af RcComal80-fortolkeren.

H.1.2 Flere faciliteter

Pakker skrevet i assembler giver mulighed for at udnytte maskinens faciliteter fuldt ud, f.eks. ved at lave systemkald til det underliggende styresystem eller ved at kunne addressere hele lageret.

H.1.3 Biblioteker

Ligesom et fagbibliotek f.eks. opstiller og inddeler sine bogsamlinger ud fra hensigtsmæssige temaer, kan man ved hjælp af pakker så at sige skabe sig et **pakkebibliotek**, der under forskellige temaer rummer en række samlinger af nyttige maskinkode-underprogrammer,der kan lånes og benyttes fra ethvert RcComal80 program.



Pakker i RcComal80 er udviklet for at gøre underprogrammer (dvs. procedurer og funktioner) endnu mere anvendelige: En stor opgave kan løses på en overskuelig måde ved brug af underprogrammer: Når opgaven skal løses, kan man opdele den i en række mindre og mere overskuelige opgaver. Derefter kan man benytte procedurer eller funktioner, som underprogrammer, der løser delopgaverne.

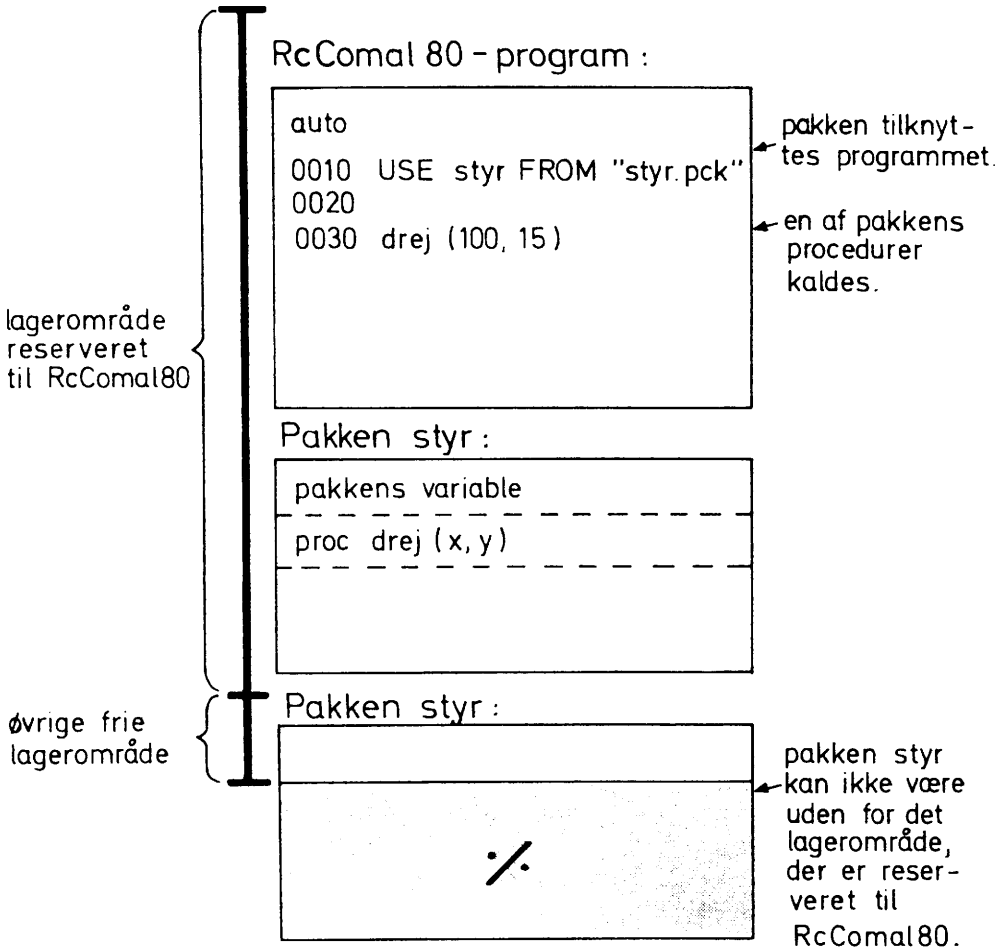
Genbrug af underprogrammer er således meget udbredt, og derfor er det en god ide, at samle underprogrammer pakkevis i et bibliotek. En pakke er altså ofte en temapakke, der under samme tema indeholder en samling underprogrammer, der løser typiske delopgaver inden for et bestemt område - f.eks. en matematikpakke, der kan løse forskellige typer ligninger, eller en mekanik-styrepakke, der kan udføre forskellige former for styring af en motor.

Det er v.hj.a. procedurer og funktioner i RcComal80 muligt at opnå en del af de fordele, der er ved biblioteker.

H.1.4 Bedre udnyttelse af lageret

Endelig giver pakker en mulighed for at udnytte PICCOLINE med et stort lager endnu bedre, når man udfører RcComal80 programmer. RcComal80 finder nemlig selv ud af, om der er plads til at placere pakken uden for det lagerområde, som er reserveret til brugerens RcComal80-program. Hvis det ikke er tilfældet, placeres pakken inden for dette lagerområde.

Desuden er det muligt at have faciliteter i RcComal80, som kun fylder i lageret, når brugeren ønsker de skal være tilstede.



H.2 Brug af pakker

For at kunne bruge pakker, er der indført 4 nye nøgleord i RcComa180, nemlig

USE...FROM...
SHOWPACK
LISTPACK
DISCARD

I dette afsnit vil vi se nærmere på, hvordan disse nøgleord bruges.

H.2.1 Tilgængelighed og brug

En pakke, der skal bruges i et RcComa180-program, gøres tilgængelig ved at give en USE-sætning eller en USE-kommando til den fil, der indeholder pakken. Filen med pakken hentes ind i programlageret, når programmet udføres (ved RUN) eller når USE gives som kommando. Bemærk: Man kan bruge (USE) lige så mange pakker, som der er plads til i lageret. En pakke kan dog kun USE's, hvis den ligger på en fil på disketten.

Når en pakke er gjort tilgængelig, kan man benytte alle de underprogrammer i form af procedurer og funktioner, som pakken stiller til rådighed for RcComa180 brugere.

Eksempel
Vi forestiller os, at der på disketten findes en fil: **COPY603.PCK**
På filen ligger en pakke **copy603**, der indeholder to maskinkode-underprogrammer, som kan udskrive hhv. hele eller en del af det nuværende grafik-skærmbillede på en Rc603-skriver.

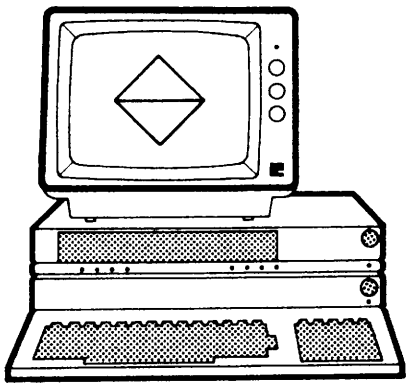
Hvis man vil gøre copy603-pakken tilgængelig, kan man f.eks. udføre følgende program (ved at give kommandoen RUN til programmet). I dette program er der også et eksempel på, hvordan en pakke-procedure (skærmkopi og skærmudsnit) kaldes. Det sker på præcis samme måde, som kald af en almindelig procedure eller funktion i RcComa180.

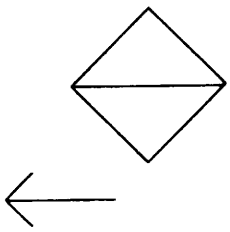
Bemærk, at programmet kun kan udføres, hvis det grafiske styresystem (GRAPHICS) er hentet ind i maskinen.

```
0010 USE copy603 FROM "copy603.pck" // pakken gøres tilgængelig
0020
0030 PROC trekant(x,y)
0040   DRAW 2*x,0
0050   DRAW -x,2*y
0060   DRAW -x,-2*y
0070 ENDPROC trekant
0080
0090 OPEN GRAPHICS
0100 WINDOW 0,100,0,100
0110
```

```
0120 MOVETO 40,46
0130 xkoor:= 20
0140 ykoor:= 20
0150 trekant(xkoor,ykoor)
0160 ykoor:= -ykoor
0170 trekant(xkoor,ykoor)
0180
0190 skærmkopi // udfør pakkeproceduren skærmkopi
0200 skærmudsnit(200,100,200,80,10) // udfør pakkenproceduren
0210                                 // skærmudsnit
```

Skærmbilledet under udførelsen af programmet ses på følgende figur, som også viser, hvad henholdsvis skærmkopi og skærmudsnit tegner på skriveren:





H.2.2 Oversigt over pakkens indhold

Efter brug af USE, er pakkens maskinkode-underprogrammer tilgængelige, og man kan få en liste over dem ved at bruge kommandoen:

```
LISTPACK copy603
```

Kommandoen vil give følgende udskrift:

```
PACKAGE copy603
  PROC skærmkopi
  PROC skærmudsnit(startx,starty,antalx,antaly,indryk)
ENDPACKAGE copy603
```

H.2.3 Oversigt over indlæste pakker

Hvis man vil have en oversigt over, hvilke pakker der for tiden er i programlageret, kan man give kommandoen:

```
SHOWPACK
```

I vores tilfælde vil resultatet så være:

```
PACKAGE copy603
```

Ved hjælp af kommandoerne SHOWPACK og LISTPACK kan man således få en fuldstændig oversigt over de biblioteker med maskinkode-underprogrammer, der er i programlageret på et givet tidspunkt.

Bemærk: Selv om en pakke er i programlageret, er det ikke sikkert, at den er tilgængelig. Her gælder de samme regler, som ved brug af den eksisterende kommando CON.

H.2.4 Fjernelse af pakker

Hvis man ønsker at fjerne pakkerne fra programlageret, bruges kommandoen

```
DISCARD
```

H.3 Virkefeltsregler

Dette afsnit beskæftiger sig med, hvor variabelnavne, der hører til pakker, kan ses og benyttes. I RcComa180 definerer hovedprogrammet og hver af de lukkede procedurer et virkefelt - dvs. et afgrænset område for synlighed og brug af variable.

Virkefeltsreglerne er kun specielle med hensyn til pakker, hvis man benytter CLOSED, EXTERNAL, eller hvis der er navnesammenfald på forskellige pakkers underprogrammer. Det anbefales derfor, at man **ikke læser afsnittet** før end man har prøvet at bruge pakker.

Vi vil se nærmere på følgende spørgsmål:

1) hvor i et program kan man angive, hvilke pakker programmet benytter.
2) hvilke navne er synlige inden for hvilke dele af RcComa180 programmerne.
3) hvordan kan man angive, at man ønsker at få fat i en bestemt procedure eller funktion, når der findes andre med samme navn.

ad 1) Det er USE-sætninger, der specificerer, hvilke pakker programmet benytter.
En USE-sætning må optræde alle steder i et RcComa180 program, undtagen inden for en ekstern procedure eller en ekstern funktion.

ad 2) Her gælder samme regel som for alle andre variable:

Uanset hvor en USE-sætning optræder i programteksten vil pakkenavnet og navnene på pakkens underprogrammer (procedurer og funktioner) blive kendte inden for det virkefelt, hvor USE-sætningen er placeret.

Eksempel
Pakken **stack** indeholder følgende:

```
LISTPACK stack

PACKAGE stack
  PROC push(x)
  FUNC pop
ENDPACK stack
```

Hvis vi så laver et RcComa180 program, der ser ud som følger:

```
0010 PROC testproc closed
0020   USE stack FROM "stack.pck"
0030   print pop
0040 ENDPROC testproc
0050
```

```
0060 a := 3
0070 push(a)
0080 print "resultat af programmet er : "
0090 testproc
```

Så fås udskriften:

```
AT 0070
*** Variabel ikke erklæret
```

Det sker, fordi pakken stack og dermed procedure push og function pop kun er synlig indenfor virkefeltet, der er afgrænset af procedure testproc.

ad 3) Hvis en pakke-procedure eller en pakke-funktion har samme navn som en anden variabel, kan den alligevel godt benyttes. Man benytter den så ved at foranstille dens navn med pakkens navn og et punktum.

Eksempel

Vi har en pakke ved navn **stack**, der indeholder følgende:

```
PACKAGE stack
  PROC push(x)
  FUNC pop
ENDPACK stack
```

Hvis vi har et program med en procedure pop, kan vi benytte proceduren pop fra pakken stack ved at skrive stack.pop:

```
0010 PROC pop
0020   PRINT "det er noget jazz"
0030 ENDPROC pop
0040
0050 USE stack FROM "stack.pck"
0060
0070 a := 7
0080 push(a)
0090 pop
0100 print "resultatet er ";
0110 print stack.pop
```

Når programmet udføres - ved brug af RUN kommandoen - vil det give udskriften:

```
det er noget jazz
resultatet er 7
```

H.4 DISCARD

RcComa180 kommando

Format
DISCARD

Anvendelse
Kommandoen fjerner alle pakker fra programlageret.

Bemærkning
1. Systemet udfører automatisk en DISCARD, hvis kommandoen NEW bruges.

Eksempel

```
SHOWPACK
  PACKAGE copy603

DISCARD

SHOWPACK
```

H.5 LISTPACK

RcComa180 kommando

Format
LISTPACK pakkenavn

pakkenavn: Navn på en bestemt pakke.

Anvendelse
Kommandoen udskriver en liste over, hvilke procedurer og funktioner, pakken stiller til rådighed, samt hvilke parametre disse procedurer og funktioner kræver.

Bemærkning
1. Kommandoen kræver, at pakken befinder sig i programlageret (dvs. at dens navn bliver vist, når kommandoen SHOWPACK benyttes).

Eksempel

```
SHOWPACK
  PACKAGE bitop
  PACKAGE copy603

LISTPACK bitop
  PACKAGE bitop
    FUNC bitand(arg1,arg2)
    FUNC bitor(arg1,arg2)
    FUNC bitxor(arg1,arg2)
    FUNC bitnot(arg1)
    FUNC bitshl(arg1,noofbits)
    FUNC bitshr(arg1,noofbits)
  ENDPACKAGE bitop
```

H.6 NEW

RcComa180 sætning/kommando

Format
NEW [filnavn]

filnavn: navnet på en fil angivet i anførselstegn.

Anvendelse
Sætningen/kommandoen bruges til at slette program, pakker og data i maskinens lager, samt lukke eventuelle åbne filer.

Bemærkning
1. NEW anvendes, når brugeren ønsker at indtaste et nyt program, og derfor vil have slettet det nuværende programindhold, de nuværende pakker og de nuværende data.

2. MARGIN- og ZONE-værdierne sættes til 0 ved NEW.

3. Alle datastrømme, der er blevet åbnet af programmet, lukkes ved NEW.
   Filer, der er blevet åbnet af pakker, bliver ikke lukket ved NEW.
   Alt hvad der i øvrigt er blevet ændret ved af pakker, sker der intet ved.

4. Angives filnavn vil filnavnet blive udskrevet i statuslinien, ellers vil den indeholde navnet
   1/SAVE.$$$ på PICCOLO, og
   01/Save.Csv på PARTNER og PICCOLINE.

H.7 RUN

RcComa180 kommando

Format
RUN [filnavn]

filnavn: Navn på en fil, angivet i anførselstegn.

Anvendelse
Kommando, der starter udførelsen af det program, der ligger i programlageret, eller LOAD'er og udfører et program, der er SAVE'd på en diskettefil.
Inden selve udførelsen af programmet påbegyndes, hentes de pakker, der skal bruges. Pakkerne hentes fra disketten (dog kun hvis de ikke allerede er i programlageret).
Alle pakker, der bruges fra programmet (se USE), bliver initialiseret.

Bemærkning
1. Hvis der kun er pakker og ikke noget program i programlageret, sker der intet ved RUN-kommandoen.

2. Datalageret slettes, når RUN-kommandoen bruges.

3. Programudførelsen starter i programlinien med det laveste linienummer.

4. Programnavnet angivet ved filnavn skal være gemt med kommandoen SAVE.

5. Ønskes udskriften standset midlertidigt, trykkes på mellemrumstasten. Ved næste tryk startes der atter.

H.8 SHOWPACK

RcComa180 kommando

Format
SHOWPACK

Anvendelse
Kommandoen angiver, hvilke pakker der befinder sig i programlageret.

Eksempel

```
SHOWPACK
  PACKAGE copy603
  PACKAGE stack
```

H.9 USE-FROM

RcComa180 sætning/kommando

Format
USE navn [FROM filnavn] { , navn [FROM filnavn] } ...

navn: Navn på pakken (max 16 tegn).
filnavn: Navn på den fil, hvori pakken er gemt.
Navnet angives i anførselstegn.

Anvendelse
Sætningen/kommandoen benyttes til at specificere, hvilke pakker programmet benytter. Hvis 'FROM filnavn' udelades, benyttes pakkenavnet som filnavn.

Bemærkning

1. Hvis der ikke angives nogen type i filnavnet, tilføjes automatisk .PCK til navnet på Partner og PICCOLINE.

2. Afgives USE som kommando, hentes pakken ind i programlageret (såfremt den ikke findes der i forvejen) og pakken initialiseres. Under initialiseringen kan der evt. være kommunikation med brugeren. I så fald vil det være beskrevet i den dokumentation, der hører til pakken.
Ved initialiseringen bliver procedurer og funktioner i pakken kendte, så de kan udføres på kommando-niveau.
Navne, som var kendte før, er ikke længere kendte.

3. Gives USE som sætning, sker der følgende ved RUN-kommandoen:
Pakken hentes ind i programlageret og initialiseres, inden den egentlige programudførelse starter.
Under initialiseringen kan der evt. være kommunikation med brugeren. I så fald vil det være beskrevet i den dokumentation, der hører til pakken.
Ved initialiseringen bliver procedurer og funktioner i pakken kendte, så de kan udføres som sætninger i programmet og på kommando-niveau.
Under programudførelsen kan pakkens underprogrammer benyttes på samme måde, som hvis de var erklærede som lukkede procedurer/funktioner på det sted, hvor USE-sætningen er.
Pakker og underprogrammer, der var kendt før RUN-kommandoen, er ikke længere kendte.

4. Der kan kun være 1 pakke på hver fil.

5. I en pakke på en fil, er der altid beskrevet, hvad pakken hedder. Parameteren navn i USE-sætningen/ -kommandoen, skal være lig med dette navn. Navnet kan findes i den dokumentation, der hører til pakken.

6. Der må gerne bruges flere pakker fra samme program. Den eneste begrænsning på antallet af pakker, som et program kan bruge, er, at der skal være plads til pakkerne i lageret. Dvs. enten udenfor det lager, der bruges af RcComa180, eller inde i det lager, som RcComa180-fortolkeren bruger til Coma180-programmet.

7. Ofte kan FROM filnavn udelades, idet filen har samme navn som pakken og med filtype PCK.

Eksempel

```
USE program
kommando("graphics")
```

Dette bevirker, at grafik-styresystemet installeres inde fra RcComa180, hvis der er plads nok.

H.10 Kommando- eller udførelsesfejl

Ud over de allerede eksisterende fejlmeddelelser, er der blevet tilføjet følgende i forbindelse med indførelsen af pakker:

0150: FEJL I PAKKENAVN
Pakkenavn svarer ikke til det navn, der er i filen, f.eks:

```
0010 USE pop FROM "stack.pck"
```

0151: PAKKE ER FOR STOR
Pakken er større end 32 Kbyte, og kan derfor ikke hentes ind i lageret.

0152: FIL INDEHOLDER IKKE PAKKE
Den fil, der er angivet som indeholdende en pakke, har et format der viser, at den ikke kan indeholde en pakke.

0153: IKKE PLADS TIL PAKKE
Den fil, der indeholder pakken, er så stor, at den ikke kan være i det resterende af maskinens lager.

I. PROGRAMMERING AF PAKKER TIL RCCOMAL80

Pakker til RcComal80 skal skrives i assembler, dvs. ASM86 eller RASM86. Det er derfor nødvendigt at have kendskab til et af disse assemblersprog, hvis du vil forstå dette afsnit.

Der skal gøres opmærksom på at programmering i assembler giver adgang til alle faciliteter i maskinen såsom styresystemkald og alle lagerceller, hvilket gør det muligt "at slå benene væk under sig selv", således at maskinen må genstartes. Det er i modsætning til programmering i RcComal80, hvor du befinder dig i beskyttede omgivelser og bliver advaret, hvis du forsøger at lave noget galt. Det er sværere at finde fejl i et assembler-program, fordi programmet ofte fortsætter efter en fejl, så det er umuligt at se, hvad der går galt.

Hvis du vil programmere i assembler på en PICCOLINE, er det nødvendigt at anskaffe CCP/M-86 Manual Set (SW1699D), som bl.a. indeholder "Programmer's Utilities Guide", som dokumenterer ASM86. Desuden indeholder manualsættet "Programmer's Reference Guide", som beskriver de forskellige styresystemkald m.m., hvis du får brug for nogle af dem i dine pakker. Det kan også være nødvendigt at anskaffe "PICCOLINE Programmer's Guide" (SW1498D). Det skal bemærkes, at alle disse manualer er på engelsk.

De eksempler, der omtales i dette bilag findes alle som kildetekster (filtype A86) på disk 4/4 i PICCOLINE SW1400 release 3.0 eller på distributionsdisketten for RcComal80 release 2.0. Dette gør det nemmere at lave sine egne pakker, idet det er muligt at genbruge en del fra disse pakker, specielt vedrørende hoved på en pakke og overførsel af parametre (se I.4 og I.7).

I.1 Format for en pakke

Der er nogle specielle formater, som et assemblerprogram skal overholde, for at det kan bruges som en pakke af RcComal80:

- forrest i codesegmentet skal der defineres et hoved, der angiver:
  - pakkens navn
  - antal procedurer og funktioner, der skal kendes fra RcComal80
  - definition af procedure- og funktionshoveder for disse

  Formatet beskrives nærmere under afsnittet med eksempler (afsnit I.4).

- forrest i datasegmentet skal der stå:

```
ORG 100H
RW 3
```

  Den første linie angiver, at de første 256 (100 hexadecimalt) bytes ikke må bruges (de skal bruges til Base Page). Den anden linie reserverer 3 ord, som bruges, når en assembler-rutine kaldes.

- alle procedurer og funktioner, der angives i hovedet (se ovenfor), skal afsluttes med et RETF (return far), og stakkens størrelse skal være den samme som på kaldtidspunktet (dvs. antal PUSH og CALL skal være det samme som antal POP og RET).

- der skal være en initialiseringsrutine (angives i hovedet). Denne rutine kaldes automatisk, når det angives at en pakke skal benyttes (se USE i afsnittet om brug af pakker). Denne rutine skal også afsluttes med et RETF. Initialiseringsrutinen benyttes til at initialisere eventuelle fælles variable.

I.2 Ved kald af pakkerutine

Når en af pakkens procedurer eller funktioner kaldes skiftes DS til pakkens datasegment. Hvis pakken har et staksegment, skiftes SS til dette, ellers bruges det samme staksegment som RcComa180 bruger.

Når en af procedurerne eller funktionerne kaldes er stakkens udseende (Bemærk: når stakken vokser bliver stakpointeren mindre):

adresse relativt til staktoppen (i bytes)	indhold
0	offset for returadresse
2	segment for returadresse
4	offset for tabel af RcComa180-rutiner
6	segment for tabel af RcComa180-rutiner
8	segment for parametre
10	offset for evt. første parameter
12	offset for evt. anden parameter
14	...

De to øverste ord skal bruges, når der skal returneres fra pakken (ved RETF) (se også næste afsnit).

De to næste ord indeholder adgang til forskellige RcComa180-rutiner, som bl.a. gør det muligt at konvertere tal på RcComa180-format til 16 bits heltal og omvendt. Disse rutiner er nærmere beskrevet i afsnit I.11.

De resterende ord gør det muligt at få fat i parametrene (se I.4 og I.7).

I.3 Betingelser ved afslutning

Når et kald afsluttes ved et RETF, er der tre betingelser, der skal være opfyldt:

- DS-registeret skal indeholde data-segmentet for pakken

- retur-adressen skal stå som de to øverste ord på stakken, dvs. pakken skal sørge for at tage lige så meget af stakken, som den lægger på stakken.

- hvis det er et funktionskald, der afsluttes, skal register AX indeholde segmentet for resultatet, og register CX skal indeholde offset for resultatet. Desuden skal DX indholde længden af resultatet, hvis det er en strengfunktion.

I.4 Eksempler

Dette afsnit indeholder to eksempler på pakker: PROGRAM og BITOP. Assemblerkoden for pakkerne ligger på filerne PROGRAM.A86 og BITOP.A86 på disk 4/4 i PICCOLINE SW1400 release 3.0 eller på distributionsdisketten for RcComa180 release 2.0.

I.4.1 PROGRAM

Pakken PROGRAM indeholder kun en procedure kommando, som laver et kald af det program, som strengparameteren comstr$ angiver. Efter kaldet af dette program returneres til RcComa180. F.eks. kan dette bruges til at starte grafikken inde fra RcComa180 ved følgende to linier:

```
USE program
kommando("graphics")
```

Hovedet for pakken PROGRAM er:

```
CSEG
PACKAGE     DB      7, 'PROGRAM'          ; pakke program
INIT        DW      OFFSET FINITO         ;
PACKTYPE    DB      'ass'                 ;
PROC        DB      1                     ;
FUNC        DB      0                     ;

    DB      8,'KOMMANDO'          ; procedure kommando(comstr$);
    DW      OFFSET XCOM           ;
    DB      1                     ;         1 parameter
    DB      6, 'COMSTR'           ;
    DB      2,0                   ;         simple_string

DB  'ass'                 ; markerer afslutning af pakkens hoved
```

Den anden linie definerer pakkens navn: navnet består af 7 tegn, som er PROGRAM.

Den tredie linie angiver offset for initialiseringsrutinen. Her hedder offset FINITO, som er en rutine, der bare indeholder RETF. Der er altså ingen variable, der skal initialiseres. Bemærk, der bruges DW, idet et offset fylder to bytes.

I den fjerde linie angives ved DB 'ass', at det er en pakke skrevet i assembler. Bemærk her skal antallet af tegn ikke angives. DB 'ass' bruges også ved markering af afslutningen på hovedet (se sidste linie).

I linie 5 og 6 angives, hvor mange procedurer og funktioner, der defineres i denne pakke (her 1 procedure og ingen funktioner).

Derefter følger definitionerne af de enkelte procedurer og funktioner.

For hver procedure angives:

- navnet (DB 8,'KOMMANDO')
- offset for proceduren. Bemærk der bruges DW
- antal parametre (her 1)
- de enkelte parametre

En parameter beskrives ved to linier. Først navnet (DB 6,'COMSTR') og dernæst typen af parameteren (DB 2,0). Det første tal angiver, hvad slags parameter det er:

0 svarer til en numerisk parameter
1 svarer til en numerisk REF parameter
2 svarer til en strengparameter
3 svarer til en REF strengparameter

Det andet tal angiver dimensionen af parameteren. Når det første tal er 0 eller 2 (dvs. det ikke er en REF parameter), skal dimensionen være 0.

En funktion har næsten samme format, se næste eksempel i afsnit I.4.2.

Selve koden for proceduren kommando består af seks dele:

1) sæt prioriteten af dette program til 197

2) hent parameteren fra RcComa180's datasegment over i pakkens datasegment

3) udfør programmet som strengen angiver

4) tilknyt dataskærmen til dette program, når programmet er færdigt

5) sæt prioriteten til 200

6) programmer funktionstasterne, som de skal være i RcComa180, da det udførte program kan have ændret funktionstasterne.

Her er det kun punkt 2 og 6, der specielt har noget med pakker at gøre; de andre punkter er styresystemkald. Vi vil derfor se nærmere på på koden for punkt 2 og 6. Hele eksemplet findes på filen PROGRAM.A86, og den oversatte version af pakken på filen PROGRAM.PCK.

Punkt 2 flytter strengen fra RcComal's datasegment over i pakkens datasegment. I RcComal ligger en streng på følgende format (hver linie gælder for en byte, dvs. længderne fylder to bytes (et ord) og de enkelte tegn en byte):

maksimale længde
aktuelle længde
første tegn
andet tegn
...
...
sidste tegn

Koden er som følger:

```
MOV     GEMP,SP             ;
MOV     BP,SP               ;
MOV     AX,DS               ;
MOV     ES,AX               ;
MOV     DI,OFFSET CLBUF1    ;
PUSH    DS                  ;
MOV     DS,SS:8ÆBPÅ         ; get_segment_for_parameter;
MOV     SI,SS:10ÆBPÅ        ; get_address_for_parameter;
INC     SI                  ; skip_max_length_of_parameter;
INC     SI                  ;
LODS    AX                  ; get_length_of_parameter;
MOV     CX,AX               ;
REP     MOVSB               ; move_parameter_to_clbuf;
MOV     AL,0                ;
STOSB                       ; move_zero_to_clbuf;
POP     DS                  ;
```

Først gemmes stakpointeren (SP) i en variabel (gemp) i pakkens datasegment, så er det nemt senere at genetablere denne staksituation. Når vi skal flytte en sekvens af bytes (som i dette tilfælde) er det nemmes at benytte REP MOVSB, hvor

DS indeholder det segment, som de bytes, der skal flyttes, ligger i
SI indeholder offset for første byte, der skal flyttes, i det segment DS indeholder
ES indeholder det segment, hvor bytene skal flyttes hen
DI indeholder offset, hvor den første af de flyttede byte skal ligge
CX indeholder antal bytes, der skal flyttes

Efter REP MOVSB indsættes et 0 efter strengen i pakkens datasegment. Det sker, fordi det kræves, at strengen slutter med et 0, når den skal fortolkes som et programnavn.

Programmering af funktionstaster svarende til RcComa180 foretages ved rutine nr. 31 i den række af rutiner, som RcComa180 stiller til rådighed for assemblerpakkerne (se afsnit I.11). Inden kaldet skal DS sættes til RcComa180's datasegment og BX sættes til rutinenummeret (31). Selve kaldet laves som et CALLF SS:DWORD PTR 4 ÆBPÅ, dvs. et call far til den adresse som det tredie og fjerde ord på stakken angiver (se afsnittet om stakken på kaldtidspunktet).

I.4.2 BITOP

Pakken BITOP indeholder 6 funktioner, der kan lave bitoperationer: and, or, xor, not, shl og shr.

Pakkens hoved fortæller at der er 6 funktioner og ingen procedurer og starter således:

```
CSEG                                        ; Code
PACKAGE         DB          5,'BITOP'       ;
INIT            DW          OFFSET FINITO   ;
PACKTYPE        DB          'ASS'           ;
PROC            DB          0               ;
FUNC            DB          6               ;
```

Formatet for funktionen bitand er som følger:

```
    DB          6,'BITAND'          ; FUNC bitand(arg1,arg2)
    DB          0                   ; reel funktion
    DW          OFFSET XBITAND      ; adresse på funktionen
    DB          2                   ; 2 parametre
    DB          4,'ARG1'            ; navn for første parameter
    DB          0,0                 ; type for første parameter
    DB          4,'ARG2'            ; navn for anden parameter
    DB          0,0                 ; type for anden parameter
```

I forhold til procedurer er der indført en linie efter navnet på funktionen, som angiver om det er en reel funktion (0) eller en strengfunktion (1).

Efter hovederne for funktionerne følger en linie:

```
    DB          'ass'
```

der markerer afslutningen på hovedet for pakken.

Parameterværdierne ligger på RcComa180 format, men ved hjælp af RcComa180-rutine nr. 1 (se afsnit I.11) kan de konverteres til 16 bits heltal (GETONEPAR i BITOP.A86):

```
MOV       BP,GEMP              ;
PUSH      DS                   ;
MOV       DS,SS:8ÆBPÅ          ; data_segment := COMAL's
                               ;                 data_segment;
MOV       SI,SS:10ÆBPÅ         ; (* første param adresse *)
MOV       BX,1                 ;
CALLF     SS:DWORD PTR 4ÆBPÅ   ; bitrep1 :=
                               ;      fix(første parameter);
POP       DS                   ; data_segment := pakkens
                               ;                 data_segment;
OR        AX,AX                ; IF fejl ved konvertering THEN
JNZ       ERR                  ;         error(ill_integer);
MOV       BITREP1,BX           ;
```

Her bruges variablen GEMP til at indeholde værdien af stakpointeren ved kaldet af funktionen. Efter CALLF, som udfører konverteringen, indeholder BX det 16 bits heltal og AX en eventuel fejlkode ved konverteringen. Den anden parameter hentes på samme måde, idet 10 ændres til 12 i linie 4.

Resultatet af funktionen konverteres fra 16 bits heltal til RcComa180 format ved rutine nr. 0 (PUTRES i BITOP.A86):

```
MOV       SI,BX                ; (* resultat *)
MOV       BP,GEMP              ;
PUSH      DS                   ;
MOV       DS,SS:8ÆBPÅ          ; data_segment := COMAL's
                               ;                 data_segment;
MOV       BX,0                 ;
CALLF     SS:DWORD PTR 4ÆBPÅ   ;
MOV       CX,SI                ; value := float(resultat);
MOV       AX,DS                ; segment for value;
POP       DS                   ; data_segment := pakkens
                               ;                 data_segment;
```

Derefter sættes AX til at indeholde segment og CX offset for, hvor resultatet ligger.

I.5 Oversættelse af pakker

Pakker skrevet i ASM86 skal oversættes med ASM86, og derefter skal der genereres en CMD-fil v.hj.a. GENCMD. Til sidst bør du omdøbe CMD-filen til at have filtypen PCK.

Til at gøre alt dette findes en SUBMIT-fil PCK.SUB, så en oversættelse af assemblerprogrammet PROGRAM.A86 sker ved

```
SUBMIT PCK PROGRAM
```

Pakker kan også oversættes med RASM86, og der kan genereres en CMD-fil med LINK86, hvis du har disse to programmer til rådighed.

ASM86 og GENCMD ligger på en af distributionsdisketterne for maskinen. RASM86 og LINK86 er en del af SW1603 (DR assembler plus tools).

I.6 Aftestning af pakker

En pakke aftestes nemmest ved først at programmere den som et assemblerprogram, som oversættes ved ASM86 og GENCMD (se forrige afsnit). CMD-filen kan så udføres direkte. Udførelsen kan debugges v.hj.a. DDT86 eller SID86, hvorved det er nemmere at finde fejl.

Når assemblerkoden fungerer korrekt indsættes hovedet i codesegmentet, og datasegmentet udvides med det nødvendige. Desuden skal der indsættes RETF for hver af de rutiner, der angives i hovedet. Koden skal også ændres til at overføre eventuelle parametre.

Det er nemmest at teste, at hovedet er korrekt ved fra RcComal80 at udføre en USE-kommando efterfulgt af en LISTPACK-kommando, som viser de procedurer og funktioner, der er til rådighed fra pakken.

I.7 Parametre

Når en procedure eller funktion har parametre, er det en god ide at have en variabel i pakkens datasegment, som peger på toppen af stakken (som den er lige efter kaldet af proceduren eller funktionen). Så kan parameterværdierne findes uanset om pakken lægger noget på stakken. Dette fremgår også af de to eksempler, der er omtalt tidligere (afsnit I.4).

I.7.1 Reelle parametre

Reelle parametre fylder 8 bytes på RcComal80-format. I samlingen af RcComal80-rutiner, der kan benyttes fra assemblerpakker, findes der rutiner til at regne med reelle tal på RcComal80-format og rutiner til at konvertere mellem RcComal80-format og 16 bits heltal (se afsnit I.11). Normalt anvendes 16 bits heltal i assembler-programmering, idet tallene så kan være i et register. Det giver en mindre nøjagtighed end i RcComal80 (13 betydende cifre). Se endvidere eksemplet BITOP (afsnit I.4.1).

I.7.2 Streng parametre

Streng parametre er beskrevet under eksemplet PROGRAM (afsnit I.4.1).

I.7.3 Tabel parametre

Tabel parametre må kun være REF-parametre (se næste afsnit). Værdien af en tabel parameter indeholder både en beskrivelse af data-formatet (en såkaldt dope-vektor) og de aktuelle data:

reference til dope-vektor
reference til data

Begge referencer fylder et ord (to bytes).

En dope-vektor har følgende format:

no. of dimensions
max index n'te dimension
min index n'te dimension
"size" n'te dimension
...
...
max index første dimension
min index første dimension
"size" første dimension

Alle felter undtagen det første fylder et ord (to bytes). no. of dimensions fylder kun en byte.

Se eksemplet nedenfor vedrørende "size".

Data for en tabel ligger grupperet på følgende måde: først ligger alle tabelindgange, hvor første index er "min index for første dimension", dernæst alle tabelindgange hvor første index er "min index for første dimension" + 1, o.s.v. og til sidst alle tabelindgange hvor første index er "max index for første dimension".

Indenfor hver gruppe er de grupperet efter andet index, hvis der er flere dimensioner. O.s.v.

Eksempel:

```
DIM a(1:7, 1:5)          // taltabel med to dimensioner
```

Data for a (hvert element fylder 8 bytes, da det er en taltabel):

```
a(1,1)
a(1,2)
a(1,3)
a(1,4)
a(1,5)
a(2,1)
a(2,2)
...
...
a(7,4)
a(7,5)
```

Dope-vektor for a

2	antal dimensioner
5	max index anden dimension
1	min index anden dimension
8	"size" for elementer
7	max index første dimension
1	min index første dimension
40	"size" for gruppe af elementer med samme første index

"Size" for første dimension er 40, da gruppen af elementer med ens første index er 5 elementer a 8 bytes lig med 40 bytes. Det betyder, at gruppen af elementer med første index lig 4 starter (4-1)*40 = 120 bytes efter starten på data.

"Size" for anden dimension er 8, da elementerne indenfor en gruppe med samme første index fylder 8 bytes, da de er tal.

Teksttabeller ligger på samme måde, men længderne af basis-elementerne er ikke 8, men længden af basisstrengene, som angives i DIM-sætningen.

I.7.4 REF-parametre

Pakken skal selv sørge for at ændre parameter-værdierne for REF-parametre. Det gælder også, hvis du bruger en af RcComa180-rutinerne (se afsnit I.11): de ændrer aldrig på værdierne af argumenterne.

I.8 Fejlhåndtering

Pakken skal selv tage sig af fejlhåndtering og sørge for eventuelle fejludskrifter, og brugeren af pakken skal sørge for, at RcComal80-programmet undersøger, om der er opstået fejl under udførelsen af en pakkeprocedure eller - funktion. Det skal fremgå af beskrivelsen af pakken, hvordan pakken leverer fejlmarkeringer tilbage (det kan f.eks. ske gennem en REF-parameter).

I.9 Gode råd og advarsler

I.9.1 Stak

Når en pakkeprocedure eller -funktion kaldes vil den bruge samme assemblerstak, som den RcComal80 bruger, hvis der ikke er defineret et staksegment i pakken. Hvis der bruges meget stakplads, kan det blive nødvendigt, at pakken definerer sit eget staksegment. RcComal80-fortolkeren sørger så for at skifte til dette staksegment ved kaldet og skifte tilbage til RcComal80's staksegment efter returneringen til RcComal80.

I.9.2 Markørplacering

RcComal80 holder hele tiden rede på, hvor markøren er placeret på skærmen. Hvis pakken ændrer på markørplaceringen ved udskrift på eller indlæsning fra skærmen, vil markørens placering være ændret, når der returneres til RcComal80; ofte til at stå oven i det, der er skrevet på skærmen.

Under alle omstændigheder bør det fremgå af dokumentationen for pakken, hvis problemet kan forekomme.

I.9.3 Filer

Hvis et pakke-rutine åbner en fil, skal denne fil lukkes, inden RcComal80-programmet afslutter sin udførelse. Det skal fremgå af dokumentationen, hvis brugeren skal huske at kalde en pakke-rutine, inden RcComal80-programmet afsluttes.

I.9.4 Tegnsæt

Hvis en pakke-rutine ændrer maskinens tegnsæt, skal RcComal80-programmet være programmeret sådan, at man ved afslutningen af programudførelsen får det oprindelige tegnsæt igen.

I.10 Dokumentation

Til en pakke skal der laves den nødvendige dokumentation, så en bruger ikke er i tvivl, hvordan pakkens rutiner virker, hvilke fejlmuligheder der er, og hvordan de markeres.

I.11 RcComal80-rutiner

Dette afsnit beskriver de RcComal80-rutiner, som det er muligt at kalde fra en assemblerpakke, se afsnittet om stakkens udseende ved kald (I.3) og afsnittet med eksempler (I.4). Rutinerne kaldes ved et CALLF med det dobbeltord på stakken, som indeholder adressen på indgangen til disse rutiner.

Ved kaldet skal BX indeholde nummeret på rutinen. Hvis rutinen har parametre skal SI pege på første parameter og DI pege på den eventuelle anden parameter. Efter udførelsen vil SI pege på resultatet, hvis der er noget, og AX vil indeholde en fejlkode, hvis der er opstået en fejl. Det ovennævnte gælder dog ikke ved rutine 0, 1 og 2, som er konverteringsrutinerne mellem formatet på tal i RcComal80 og 16 bits heltal (se nedenfor).

Fejlkoderne er nærmere beskrevet i RcComal80 manualen appendiks C.

Rutine nr og navn		Beskrivelse
0	FLOAT	Konverterer det 16 bits heltal, som SI indeholder til RcComal80 format. Efter kaldet vil SI pege på værdien på RcComal80 format.
1	FIX	Konverterer det tal på RcComal80 format, som SI peger på til 16 bits heltal. Efter kaldet vil BX indeholde det 16 bits heltal, og AX kan indeholde fejlkoden 101, som betyder at tallet er for stort eller for lille til at blive konverteret.
2	STR	Konverterer det tal SI peger på til en streng af tegn, så tallet let kan skrives ud. Efter kaldet indeholder BX længden af strengen og DX peger på det første tegn.
3	ADD	Adderer de to tal på RcComal80 format, som SI og DI peger på. Tallene, som SI og DI peger på ændres ikke, men SI ændres så den efter kaldet peger på et tal, der er summen af de to tal. AX kan indeholde fejlkoden 106, hvis der er arithmetisk overløb.
4	SUB	Subtraktion, registre bruges som ved ADD. Bemærk SI er første parameter.
5	MULT	Multiplikation, registre bruges som ved ADD.
6	DIVI	Division, registre bruges som ved ADD. Bemærk SI er første parameter. Desuden er der mulighed for fejlkode 104 (Division med 0).
7	MOD	Heltalsmodulus, registre bruges som ved ADD. Bemærk SI er første parameter.

8	DIV	Heltalsdivision, registre bruges som ved ADD. Bemærk SI er første parameter. Desuden er der mulighed for fejlkode 104 (Division med 0).
9	POWER	Potensopløftning, registre bruges som ved ADD. Bemærk SI er første parameter.
10	EQ	Sammenligning, registre bruges som ved ADD. Efter kaldet peger SI på tallet 1, hvis de to tal er ens, ellers peger SI på tallet 0.
11	NE	Ikke-ens, ellers som EQ.
12	LT	Mindre-end, ellers som EQ.
13	LE	Mindre-end-eller-lig-med, ellers som EQ.
14	GT	Større-end, ellers som EQ.
15	GE	Større-end-eller-lig-med, ellers som EQ.
16	OR	Logisk or, registre bruges som ved ADD. Efter kaldet peger SI på tallet 1, hvis bare et af tallene er forskellige fra 0, ellers peger SI på tallet 0.
17	AND	Logisk and, registre bruges som ved ADD. Efter kaldet peger SI på tallet 1, hvis begge tal er forskellige fra 0, ellers peger SI på tallet 0.
18	NOT	Logisk not. Ved kaldet peger SI på parameteren. Efter kaldet peger SI på tallet 1, hvis parameteren er lig 0, ellers peger SI på tallet 0.
19	ABS	Absolut værdi. SI peger på parameteren. Efter kaldet peger SI på den absolutte værdi.
20	SIGN	Fortegnsværdi. SI peger på parameteren. Efter kaldet peger SI på tallet 1, hvis parameteren er positiv, tallet 0 hvis parameteren er 0, og tallet -1 hvis parameteren er negativ.
21	NEG	Negation, ellers som ABS.
22	INT	Den hele del, det nærmeste heltal der er mindre end eller lig parameteren. Ellers som ABS.
23	SQR	Kvadratrod, eller som ABS. Kan returnere fejlkode 103 (Kvadratrod af negativt tal).
24	EXP	Den naturlige exponentialfunktion, ellers som ABS.
25	LOG	Den naturlige logaritmefunktion, ellers som ABS. Kan returnere fejlkode 102 (Log til negativt tal).
26	TAN	Tangensfunktion, ellers som ABS. Regner med radianer.

27	COS	Cosinusfunktion, ellers som TAN.
28	SIN	Sinusfunktion, ellers som TAN.
29	ATN	Arctangensfunktion (den omvendte funktion til tangens), ellers som TAN.
30	PI	Efter kaldet peger SI på tallet pi. Ingen parametre.
31	FUNKKEY	Programmerer funktionstaster, som de skal være i RcComa180.

Bemærk: ved funktioner vil de tal, som SI og DI peger på, ikke blive ændret, men SI vil blive ændret til at pege på det sted, hvor resultatet ligger.

Ved kald af flere funktioner efter hinanden kan det være nødvendigt at gemme et mellemresultat ved at flytte værdien over i pakkens datasegment, da det ellers kan blive ødelagt ved næste funktionskald. Når så værdien skal flyttes tilbage til RcComa180's datasegment, er der plads på adressen, der har offset 8 mindre end den adresse, der returneres i SI fra de forskellige rutiner, bortset fra rutine nr. 1, 30 og 31.

Eksempler

sin(par1)+cos(par2) kan udregnes således

```
SI:= adresse på par1
kald sin   (rutine nr 28)
flyt resultatet over i pakkens datasegment
SI:= adresse på par2
kald cos   (rutine nr 27)
flyt resultatet af sin-kaldet fra pakkens datasegment
  til RcComa180's datasegment til en adresse der har et
  offset, der er 8 mindre end offset, hvor resultatet af cos-
  kaldet ligger
DI:= adresse returneret ved cos-kaldet
SI:= adresse på sin-værdien (* SI = DI-8 *)
kald add   (rutine nr 3)
```

exp(sin(par1)) kan udregnes som følger (her er det ikke nødvendigt at flytte resultater, da der kun er en resultat-værdi af gangen):

```
SI:= adresse på par1
kald sin   (rutine nr 28)
kald exp   (rutine nr 24)
```