
RCSL Nr.: 31-D691

Udgave: April 1983

Forfatter: Henrik Steen Møller

Titel:

RC8000 Betjeningsvejledning for operatører.

Nøgleord:

Danish Language. RC8000 hardware, NCP, autoloader, Monitoren, Operativsystemer, File-processeren, Utility programmer, Back-up af data, Discstat, fpastat, Coredump.

Resumé:

Beskrivelse af hardware og software på RC8000.

Manualen henvender sig fortrinsvis til personer der har eller skal betjene RC8000.

90 trykte sider.

Copyright © 1983, A/S Regnecentralen af 1979
RC Computer A/S
Udgivet af A/S Regnecentralen af 1979, København

Brugere af denne manual gøres opmærksom på, at specifikationerne heri uden forudgående varsel kan ændres af RC. RC er ikke ansvarlig for typografiske fejl eller regnefejl, som kan forekomme i denne manual, og er ikke ansvarlig for skader forårsaget af benyttelsen af dette dokument.

FORORD

Første udgave: RCSL No 31-D691.
RC8000 operatørvejledning.

Denne manual omhandler hardware og software på RC8000.

De første 10 kapitler omhandler den hardwaremæssige side af RC8000, mens de 24 resterende kapitler omhandler den softwaremæssige side. Bagest i manualen findes et stikordsregister, som tænkes anvendt ved opslag i manualen, så eventuelle ord eller forklaringer gerne skulle blive bedre belyst.

Kapitel 1 giver en kort redegørelse for hastigheder, størrelser og udtryk som anvendes på RC8000.

Endvidere refereres til følgende manualer:

RCSL No 31-D552: RC8000 Operating Systems, Introduction

RCSL No 42-i845: RC8000 Basis Programmel, Lærebog

RCSL No 31-D463: RC8000 Operating Guide, Rev. 3

Henrik Møller

A/S REGNECENTRALEN af 1979, april 1983

INDHOLDSFORTEGNELSE	SIDE
1. HASTIGHEDER PÅ RC8000	1
1.1 Størrelser på RC8000	1
1.2 Udtryk	1
2. SYSTEMOPBYGNING	2
3. RC8000 SYSTEMBUSSEN	3
4. INTERNT LAGER	4
5. CPU (Central processing unit)	5
5.1 CPU'ens microprogram	6
6. FPU (FLOATING POINT UNIT)	7
7. DISC-CONTROLLER OG PLADELAGERKANALEN	8
8. FRONT-ENDELLER DEVICE CONTROLLEREN	9
9. NCP	12
10. AUTOLOAD AF RC8000	13
11. MONITOREN	15
12. MONITOREN STYRER FØLGENDE	16
12.1 Monitorens funktioner	16
12.1.1 Input-output systemet	17
12.1.2 Monitorens kanalprogrammer	17
13. MONITORENS PROCESBESKRIVELSER	19
13.1 Monitorens interne processer	20
13.1.1 Monitorens eksterne procesbeskrivelser	20
13.2 Monitorens areal proces-beskrivelser	21
13.2.1 Monitorens privilegerede funktioner	21
13.2.1.1 Monitorens beskyttelse af BS	21

INDHOLDSFORTEGNELSE (fortsat)	SIDE
14. MONITORENS KÆDETABELLER OG DISCE	23
15. DISCES OPDELING I FYSISKE OG LOGISKE	24
16. BASISPROGRAMMEL PÅ RC8000	25
16.1 File processoren	25
17. UTILITY PROGRAMS	26
18. PROCES HIERARKI	28
19. OPERATIVSYSTEMER	29
19.1 Operativsystem s	29
19.2 Operativsystem BOSS	31
19.2.1 BOSS bruger katalog	34
19.2.2 BOSS kommandoer	34
19.2.3 Nedlukning af BOSS	36
19.2.4 Fejlsituationer under BOSS, udtagning af bosstest	36
19.3 Operativsystem MIPS/TS	37
19.4 Operativsystem SOS	37
19.4.1 SOS bruger katalog	38
19.4.2 Indlogging til operativsystem SOS	39
19.4.3 Operatørvejledning for SOS	39
19.4.4 Opstart af MIPS/TS	39
19.4.5 Opstart af MIPS/TS ved hjælp af s-bruger- katalog	40
19.4.6 Operatørvejledning for TEM	45
19.4.7 Operatørvejledning for PRIMO	48
20. BACK-UP AF DATA PÅ MAGNETBÅND, DISKETTE OG DISCE	52
20.1 SAVE/LOAD magnetbånd	52
20.2 SAVE af systemfiler på disc	56
21. UTILITY PROGRAMMER	66
21.1 Filedescriptors	66
21.2 Utility copy	66

INDHOLDSFORTEGNELSE (fortsat)SIDE

21.3	Utility head	66
21.4	Utility headpunch	66
21.5	Utility move	67
21.6	Utility nextfile	67
21.7	Utility rename	67
21.8	LOOKUP kommando	67
21.9	SET	69
21.10	I	70
21.11	O	70
21.12	CLEAR	71
21.13	Scope	73
22.	DISCSTATISTIK	76
23.	FPA-STATISTIK	77
24.	INCREMENTAL DUMP	78

BILAG:

A.	INDEX	79
----	-------------	----

1. HASTIGHEDER PÅ RC8000

1.

Milli = $1/1000$

Micro = $1/1000000$

Nano = $1/1000000000$

1.1 Størrelser på RC8000

1.1

1 byte = 8 bit

1 halvord = 12 bit

1 ord = 24 bit

1 dobbeltord = 48 bit

1.2 Udtryk

1.2

En fil er en samlet mængde oplysninger, også kaldet et dokument.

En bit er den tilstand en ferritkærne kan have i et givent tidsrum.

I et ferritkærnelager består en bit af en ferritkærne.

En paritet er en kontrol af et antal bit.

Synchron = mængden af I/O kommer i tegn følgende umiddelbart efter hinanden.

Asynchron = mængden af I/O kommer i tegn uafhængigt af hinanden.

Et program er en færdigbehandlet datamængde der er i stand til at udføre en række processer.

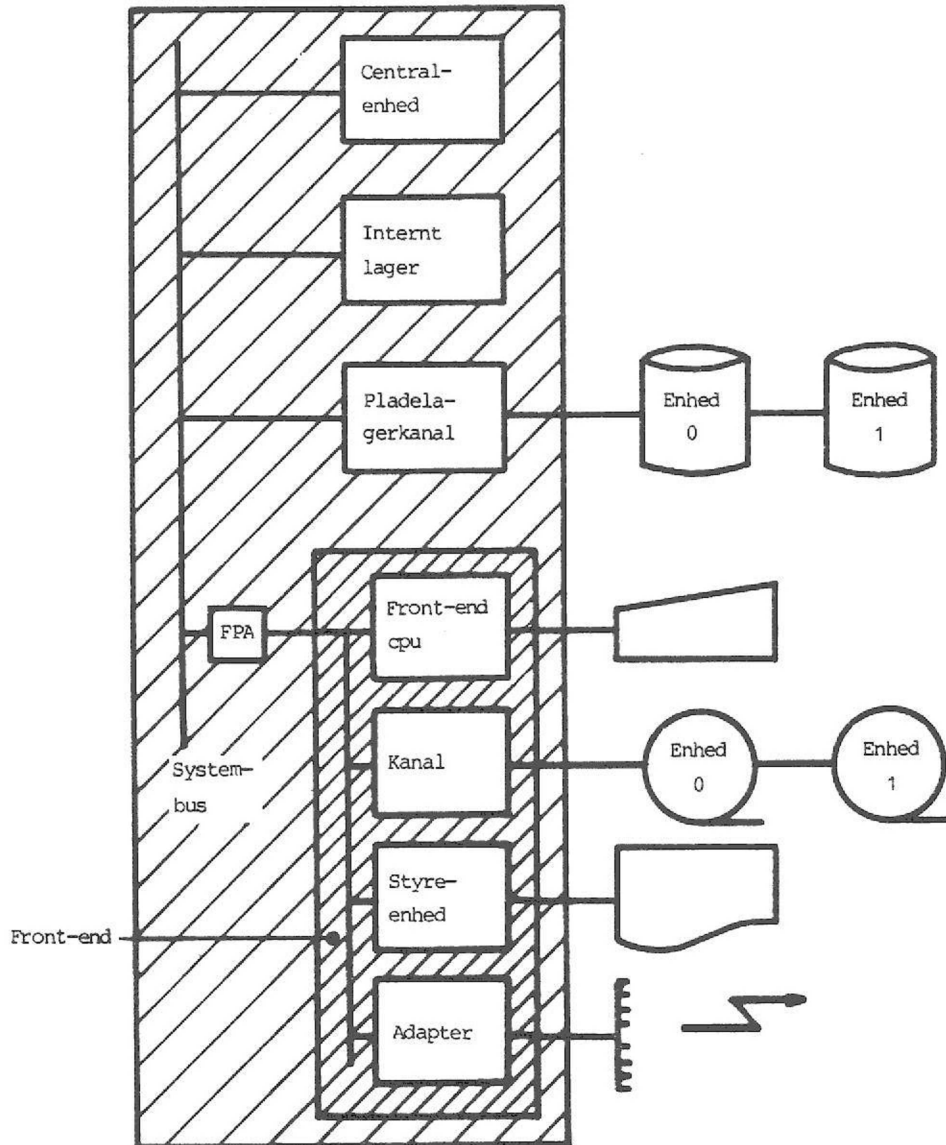
En proces er en intern samling af rettigheder.

Et operativsystem skal kunne letteregøre en programafvikling.

En entry er en indgang, som bruges for at kunne oprette en fil på en disc.

En internal bruges for at kunne oprette en proces.

En RC8000 består af 5 hovedkomponenter. Disse er CPU, lager, disc-controller, front-end og disce. CPUen er regne-enheden og lageret er hukommelsen. Disc-controlleren udfører transporter til disce og front-enden sørger for de andre ydre enheder. Det hele bliver bundet sammen af RC8000 systembussen.

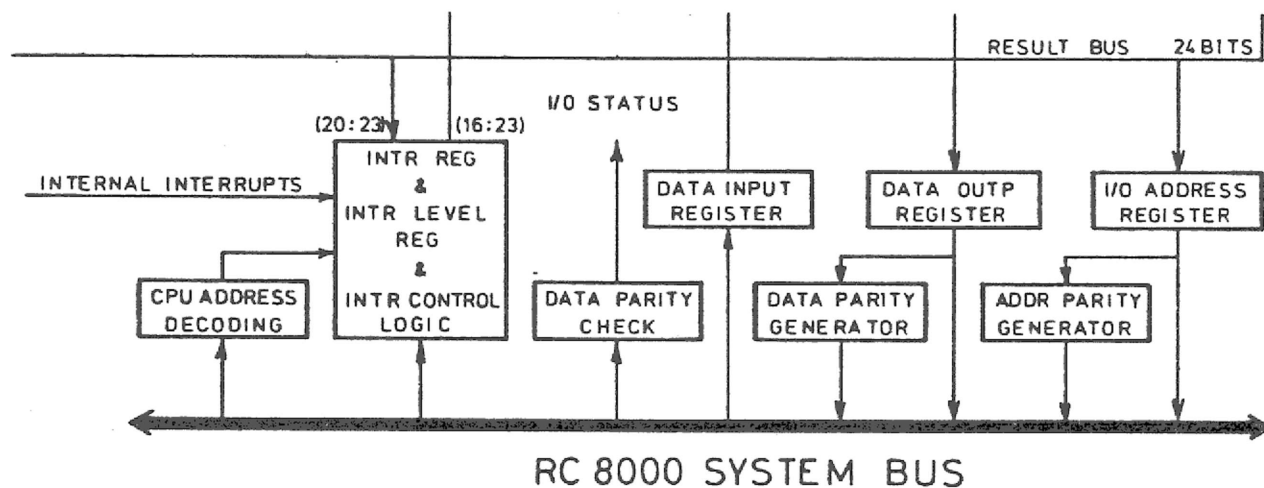


Figur 1: Systemkonfiguration på en RC8000.

RC8000 er en 24 bit ords maskine, opbygget omkring en systembus til interne data transporter. Systembussen overfører parallelt en 23 bitords adresse og et 24 bitords dataord, med en overførelses-tid på 300 nanosec.

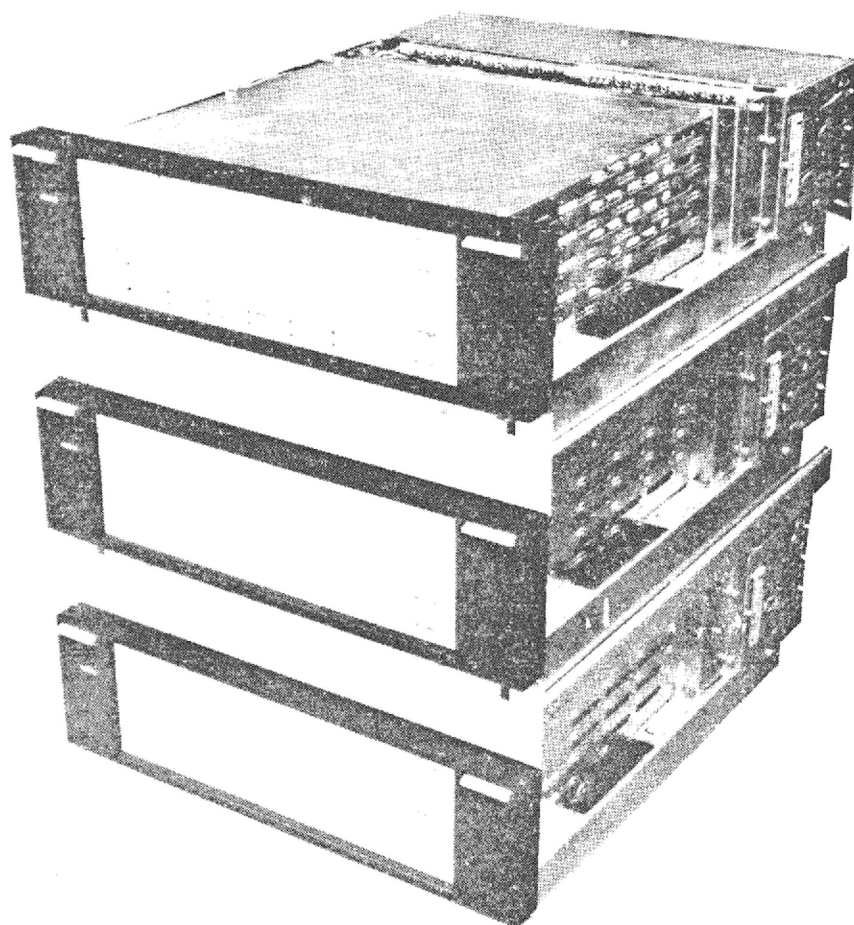
Systembussen er asynkron da der anvendes enheder med forskellige hastigheder.

Disc-controlleren får altid første prioritet. Generelt kan man sige at systembussen forenkler overfladekontakten mellem ydre enheder, den udjævner hastigheder mellem ydre enheder. Den giver ingen prioritet af jobs, alle får lov til at bruge systembussen. Den kan sammenlignes med os og et postbud. Vi skriver adressen og postbudet bringer brevet.



Figur 2: Strukturen af RC8000 systembussen.

Arbejdslageret kaldes også for primærlageret. Når datamaten er startet op, læses en instruktion ad gangen fra arbejdslageret, og instruktionen udføres. I arbejdslageret kan der være flere programmer ad gangen, man siger at datamaten er multi-programmeret. Arbejdslageret er inddelt i en mængde lige store dele, som kaldes ord. Et ord er netop den størrelse arbejdslager, som kan indeholde en maskininstruktion og en maskininstruktion er en ordre RC8000 kan tolke og udføre. Et program består af en række maskininstruktioner og når programmet afvikles, udføres de enkelte instruktioner i rækkefølge. Nye RC8000 anlæg anvender semiconductorlager eller halvleder i stedet for de ældre anlæg som anvender ferritkærlager. Semiconductorlager har 6 paritetsbit som retter eventuelle paritetsfejl, hvor ferritkærlager kun har 3 paritetsbit og disse fungerer som kontrolbits. Fordelen ved ferritkærlager er, at informationerne ikke er tabt ved strømsvigt. En kombination af ferritkærlager og semiconductorlager er også mulig. Ferritkærlager udbygges i moduler af 32 kw, medens semiconductorlager udbygges i moduler af 64 kw. Lageret kan udbygges i moduler fra 64 kw til 4 mill. ord. Accesbredden er 24 bit, dog er der ikke noget der forhindrer RC3000 i at behandle et 12 bit ord. Et ord kan læses på 0,5 microsekund.



Figur 3: CPU'en og det interne lager - modulopbygget.

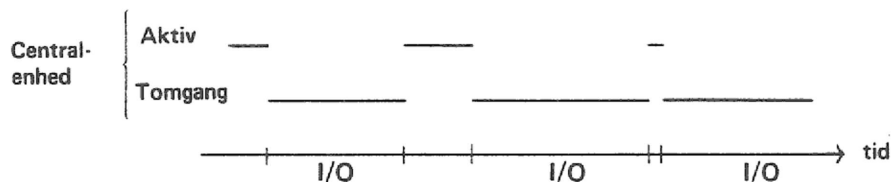
5. CPU (Central processing unit)

5.

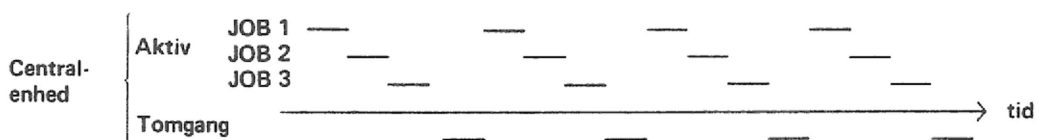
CPU'en er hovedsageligt systemets ydeevne, men den har ingen direkte indflydelse på betjeningsproceduren. Den har en instruktionshastighed på 1 million instruktioner pr. sekund med paritetskontrol af alle operationer. En typisk instruktion tager 1 til 3 microsekunder at udføre. Hvis der er cash memory på anlægget, bliver instruktionshastigheden kortet ned med cirka 1/3 i forhold til de normale CPU'er. CPU'en udfører programinstruktioner, som hentes fra det interne lager, og dens instruktioner bliver udført medens nye hentes fra lageret. Den har et 24 bitord format med mulighed for 64 basis instruktioner, og et 12 bitord er den mindste mængde data, der kan adresseres direkte. Den har 4 arbejdsregistre hvoraf 3 også fungerer som indexregistre, dvs. en slags indholdsfortegnelse. CPU'en indeholder registre og områder for logisk aritmetik. Dens vigtigste opgaver er:

- 1) generel kontrol af interrupts := prioritering af processer
- 2) beregning af I/O operationer
- 3) administration af RTC

RTC registret er et 16 bit read only register, og dette bliver talt op med 1 for hver 0,1 microsekund. Der er 5 modeller af CPU'er, model 10/S, 35/S, 45/S, 50/S og 55/S. Model 10, 35 og 45 bruger CPU model 811, medens model 50/S og 55/S bruger CPU model 820. Model 55/S bruger ligeledes CAM. Der er forskel i hastighed, pris og størrelse af lager. CPU'er fås med lagerstørrelse fra 64 k til 4 M ord.

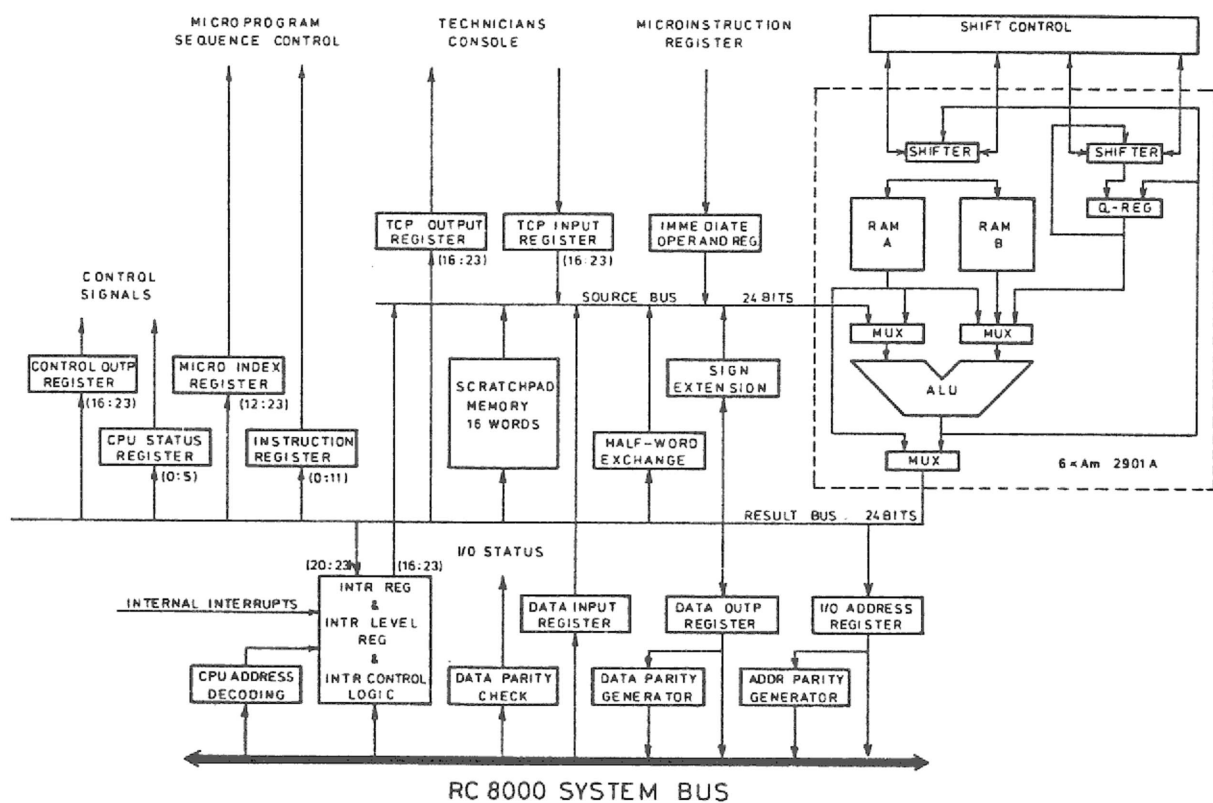


Figur 4: En dårlig udnyttelse af CPU'en.



Figur 5: En god udnyttelse af CPU'en.

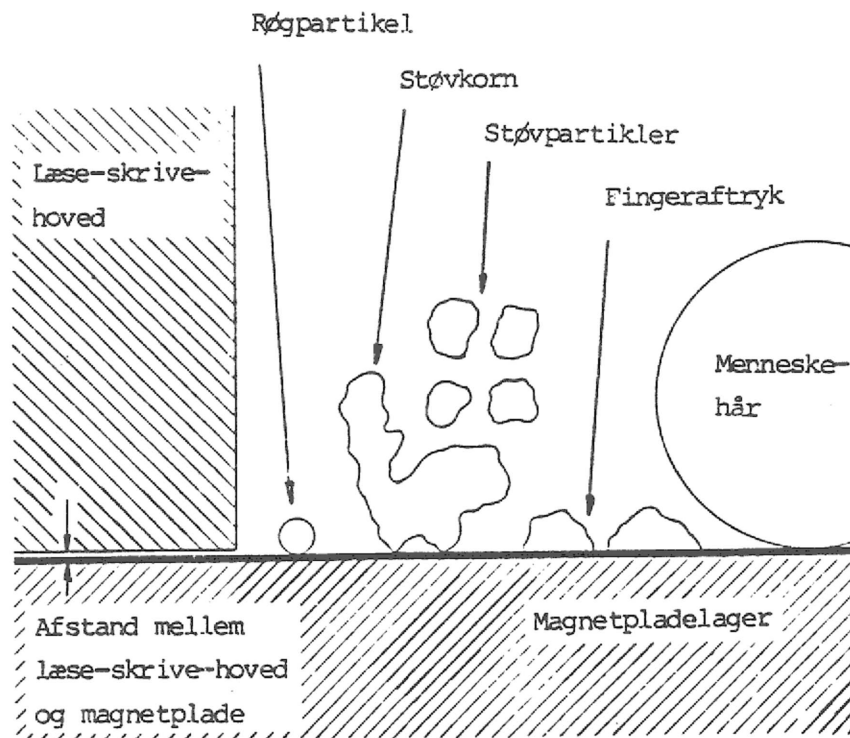
CPU'en har 32 generelle registre, blandt andre W0, W1, W2 og W3 registre. Nogle af disse registre er informationsregistre, base-registre, interruptsregistre, instruktionsregistre og I/O registre. Når instruktionerne er behandlet i de forskellige registre, bliver de sendt ned til ALU'en (Aritmetisk Logisk Unit). Herfra bliver de sendt til en resultat multiplexer, som behandler disse data inden den sender dem videre til en intern CPU resultbus, som til slut sender dem videre til systembussen.



Figur 6: CPU'ens dataveje.

FPU'en har et 48 bits dobbeltordsregister, hvormed den er designet til matematiske beregninger. Hvis et system har både CPU og FPU, vil FPU'en hver gang den modtager et dobbeltord, snuppe det fra CPU'en, og beregne det. Da FPU'en arbejder med et dobbeltordsregister, vil det gå meget hurtigere end hvis CPU'en skulle behandle den samme beregning. En almindelig CPU vil være cirka 25 microsekunder om at beregne et dobbeltord, medens FPU'en kun er 4,5 microsekunder om at udføre den samme beregning. FPU'en vil med fordel kunne bruges på anlæg der beskæftiger sig med matematiske beregninger.

Disc-controlleren udfører kontrol af programmer, uden indblanding fra CPU'en. Såfremt der er en DSA802 på anlægget, sidder der en switch og hvis denne står i "disable", er autoloade via disc ikke mulig. Hvis den står i "normal", er autoloade via disc mulig. Pladelagerkanalen bruges udelukkende til de forskellige typer disc. Der kan maksimalt kobles 4 disc pr. kanal og maksimalt 4 kanaler på systembussen. Pladelagerkanalen kobles direkte på systembussen. I tilfælde hvor et anlæg har mere end en systemdisc, er det den disc, som sidder på den første kanal på første controller, der er bestemmende for opstartproceduren. Totalkapaciteten er på 4000 MB eller ca. 16 stk. 248 MB disc. Transmissionshastigheden til og fra et discareal er 1,2 MB pr. sekund. Den gennemsnitlige søgetid er 10 millioner bit pr. sekund. Der er uafhængig positionering og overførelse af data. Der kan tilsluttes følgende typer disc: 12, 18, 33, 66, 124 og 248 MB.



Figur 7: Læse/skrivehovedets afstand fra discens overflade.

Selvom front-enden er en lille minicomputer, en RC3600, kan den ikke udføre brugerprogrammer, men er designet til at kommunikere med RC8000 og de ydre enheder. En device-controller er enten en front-end eller en koncentrator. Hvad enten det er en front-end eller en koncentrator, skal den altid have sin egen identitet (host). Den kan transmitere og modtage data via RC8000 systembussen med en hastighed på 600 kilo-bytes pr. sekund. Kommunikationen mellem front-enden og systembussen foregår via FPA'erne. Den ene sidder på RC8000 siden og den anden sidder på RC3600 siden. De kan betragtes som to subdevicer kaldet transmitter og receiver. Forbindelsen mellem FPA'erne indeholder blandt andet en kontrol linie og en autoloading request linie, som gør det muligt at initiere en autoloading-procedure i front-enden, eller der kan initieres en autoloading-procedure fra front-enden til RC8000. Denne konfiguration arbejder som et RC-NET system. FPA linien kan normalt kun være 12 meter lang. Såfremt der er længere afstand, kan der bruges lyslederkabel, minimum 25 meter, maksimum 100 meter.

Der kan maksimalt sendes 600.000 tegn/sekund igennem FPA linien og et tegn er 8 bit + 1 paritetsbit (odd parity).

Tilslutning af en ydre enhed på front-enden sker ved kald af macroer, hvor macroens navn specificerer typen af den ydre enhed og hvor mange af den pågældende type ydre enhed der skal tilsluttes. Der skal altid være mindst én hovedkonsol. Hovedkonsollen sidder direkte på front-enden, ligesom alle andre ydre enheder. Hver type enhed bruger hver sin type kanal og der kan maksimalt sidde 4 enheder pr. kanal. Ved hjælp af kanaludstyr og front-ends, kobles ydre enheder på konfigurationen. Tilslutning af terminaler foregår via kommunikations-controllere og der kan maksimalt sidde 4 AMX controllere pr. front-end. Hvis der benyttes terminalkoncentratorer (300/20) kan der sidde 3 terminaler og en seriel printer på hver kanal. Kommunikations-controllerne hedder

RC3680	=	scc/bsc	og har 1 synkron linie
RC3681	=	smx	- - 4 synkrone linier
RC3682	=	amx	- - 8 asynkrone linier

RC3683 = tmc - - 8/32 asynkrone linier
 RC3684 = hdlc - - 4 synkrone linier

Der kan maksimalt kobles 3 front-ends på systembussen og front-enden kan drive op til flere hundrede devicer, såfremt der er lager nok. Front-ends fås med lager fra 64 kb til 128 kb.

RC8000 Autoloadprogram af 1 eller flere front-ends

Brug af autoloadprogrammet

FP Utility programmet "autoload" bliver brugt til at autoloade en eller flere front-ends, som er forbundet via en FPA801 controller. Autoload informationen skal være repræsenteret som en fil på baggrundslageret.

Eksempel på brug af Utility programmet "autoload", til autoload af anden og tredje front-end:

Sæt flg. arealer på disken f.eks.	fe2 = set 400
Gør arealerne permanente:	scope user fe2
	fe3 = set 400
Monter discload-system på	scope user = fe3
device 10	
esc s	
Call 10 discsystem.	; navngiv device 10
esc s	; opret en all proces
all x run	
to x	
f = set mto discsystem 0 10	; sæt en fildescriptor til at
fe2 = move mess.yes f	; pege på fil 10 på discload-
Nextfile f	; systemet
fe3 = move mess.yes f	; flyt indholdet af filde-
	; scripteren over i det per-
	; manente område på disken
main36001 = autoload fe2	; autoload anden og tredje
main36002 = autoload fe3	; front-end med utility
	; "autoload" programmet

Funktion

Ved at initiere en autoloade funktion over FPA'erne, bliver front-enden tvunget til at starte en autoloade sekvens.

Når front-enden autoloades fra RC8000, indebærer det at flg. switche på front-enden skal pege opad 0, 10, 13 og 14 eller oktalt 46. Autoloade sekvensen starter nu en transport af load kommandoer til RC8000. Den første blok bliver herefter transporteret fra autoloadefilen til front-enden. Den næste kommando afleverer autoloadeprogrammet til RC8000, som forårsager at endnu en blok bliver transporteret til front-enden og på denne måde bliver blok efter blok transporteret til front-enden, som kontrollerer transporten med sit autoloadeprogram.

Når en "end" kommando bliver modtaget, afslutter bloktransporten og kommunikationen til front-enden bliver startet med en master clear operation.

Front-end software hedder Network Control Program. NCP er et programmelkompleks, som er sammensat af en række enkelte moduler, som initierer processer. I front-enden ses hvert af disse moduler, som en selvstændig MUS proces. Programmet omfatter coroutine-programmer og data-arealer, som f.eks. coroutine-beskrivelser og data-buffere. NCP indeholder alle data-arealer for hele programmelkomplekset. Processen NCP skal genereres specielt for hver front-end, da disse har forskellige identiteter (hoster) og forskellige kombinationer af ydre enheder. NCP styrer kommunikationen mellem RC8000 og front-enden. Ved hjælp af links kobles ydre enheder sammen. Et link er en logisk forbindelse mellem en proces ved en jobhost og en ydre enhed, som kaldes en devicehost. En forbindelse mellem RC8000 og en linieskriver koblet til front-enden sker ved hjælp af links. En jobhost er den host, hvor en given proces er oprettet. En devicehost er den, hvor en given ydre enhed er tilkoblet. Når en host er connected til NCP, kommer der en message fra NCP i form af connected to host xxxx, hvor host er værten. Der er 4 procesbeskrivelser for en front-end. Disse hedder MAIN36001, RECEIVER36001, TRANSMITTER36001 og HOST36001. Disse 4 processer har hvert sit devicenummer i monitoren.

Når autoloaddkappen trykkes i bund, stopper RC8000 og der bliver sendt en reset, hvorefter autoloaddfunktionen bliver initieret.

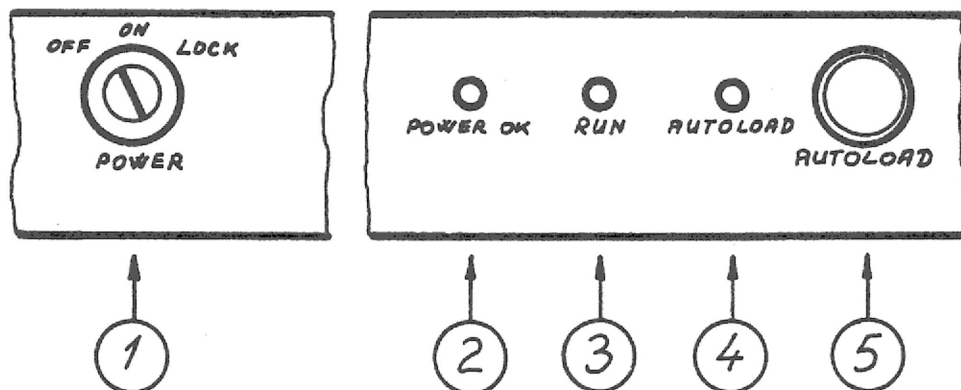
Der er 2 autoloadd signaler, hvor det første signal er forbundet til operatørens kontrol panel og udløser en autoloadd fra I/O device nummer 4 på systembussen, hvorpå systemdiscen sidder. Det andet autoloaddsignal kommer fra device nummer 2 som er første FPA's inputsignal til front-enden. I dette tilfælde er det anden autoloaddsignal-linie som bliver knyttet til front-enden og som bliver aktiveret, efter en lokal udvælgelse af loadmedie. Derefter sker en forberedelse til at sende den første blok afsted.

Denne forbindelse er gensidig, da RC8000 også kan udføre en autoloadd fra front-enden. Autoloadd aktionen starter med at sende et system "reset" signal til systembussen. Efter det, bliver lageret initialiseret og devicet som korresponderer med autoloaddsignalet, bliver aktiveret.

Devicebeskrivelsen i monitoren tilknytter et kanalprogram, som indeholder en initialiseringskommando og en læse kommando.

Autoloadd-delen på systemdiscen fylder 1049 segmenter startende i segment nr. 0 hvis discen er mindre end eller lig med 66 MB. Såfremt systemdiscen er større end 66 MB, fylder autoloadd-delen 1196 segmenter startende i segment nr. 0.

Der gælder særlige regler for fasthoved disce.

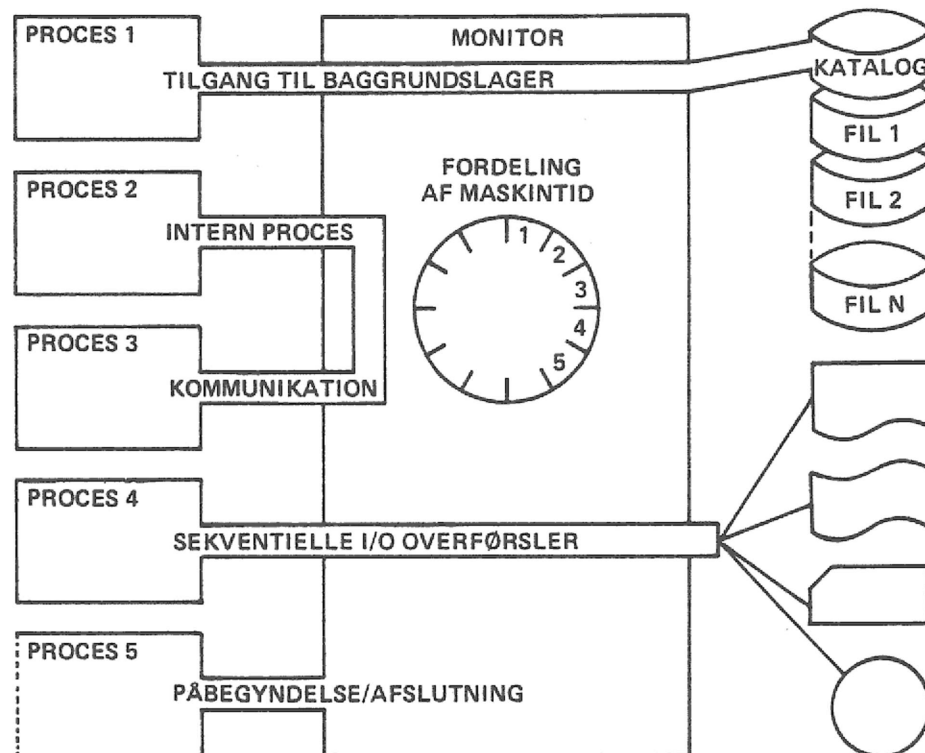


Figur 8: Operatørens kontrolpanel, med autoloaddkappen yderst til højre.

Efter autoloade indeholder primærlageret monitoren og et simpelt operativsystem kaldet "s", der sætter terminalbrugere i stand til at initiere og kontrollere interne processer. Yderligere kan "s" navngive ydre enheder.

Initielt ejer "s" alle ressourcer såsom primærlager, baggrundslager, message buffers etc. Bortset fra at være den initielle proces har "s" ingen speciel status og bliver derfor behandlet som andre interne processer. Det er muligt at erstatte "s" med et andet operativsystem.

Første gang systemet startes op, indlæses basisprogrammet fra en ydre enhed f.eks. fra en båndstation til discen. Herfra indlæses det ved daglig opstart, til det interne lager. Denne procedure kaldes autoloade. Området, hvor basisprogrammet ligger, kaldes for autoloade-delen. Basisprogrammet omfatter programmer der er systembestemt og dels en programpakke, der indeholder en række hjælpeprogrammer som kaldes system utility. Der kan suppleres med en række andre systempakker. Disse programmer styres via hovedkonsollen og startes evt. ikke automatisk, men kaldes efter behov. Disse programpakker kan f.eks. være database ALGOL, PASCAL, COBOL, COMAL, BASIC, SLANG og FORTRAN.



Figur 9: Monitoren's behandling af interne processer.

12. MONITOREN STYRER FØLGENDE

12.

- 1) børneprocesser
- 2) inddeling af CPU-tid mellem parallelle processer
- 3) fordeling af CPU-tid
- 4) tilrettelæggelse af jobs
- 5) overvågning af jobs
- 6) initiering, udførelse og afslutning af processer
- 7) kommunikation mellem processer (send mess - wait answ)
- 8) udførelse af privilegerede funktioner
- 9) varetagelse af baggrundslageret, overførelse imellem ydre enheder og det interne lager
- 10) holde styr på datamatens tilstand, et accountsystem

12.1 Monitorens funktioner

12.1

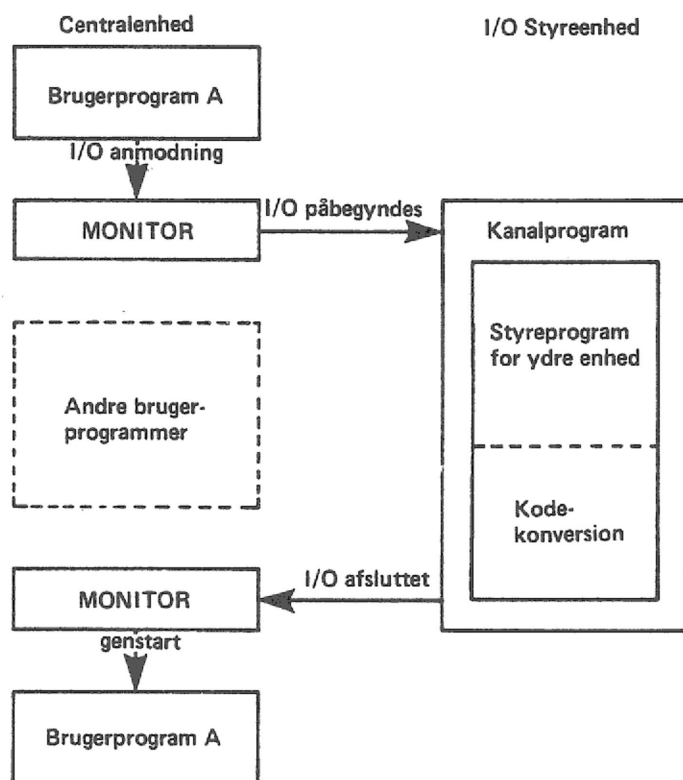
Monitoren kan deles op i 4 hovedfunktioner. Disse er:

- 1) Den centrale del som styrer kommunikationen mellem processerne og som styrer multiprogrammeringen.
- 2) Driverproc som styrer drivere til ydre enheder og I/O systemet.
- 3) Procfunc som behandler baggrundsarealer, slicetabeller og katalogbehandlingen.
- 4) Operativsystem s som blandt andet varetager oprettelse og nedlæggelse af interne processer og navngivning af ydre enheder.

Driverproc og procfunc ligger i hver sin proces.

I/O systemet er opbygget i 2 dele, hvor den ene del tager sig af I/O operationer. Driverne opbygger en kanalprogram-model med logiske adresser og kalder herefter I/O systemet. Denne ændrer adressen til fysiske adresser. I ord 8 i lageret står baseadressen, som peger på controller descriptor tabellen, således at device $:= \text{base} + 8 * \text{devicenummer} = \text{controller descriptor adressen}$.

UDFØRELSESMØNSTER



Figur 10: Udførelsesmønsteret for I/O systemet.

Monitorens kanalprogrammer styrer følgende:

- 1) Processer som varetager kommunikationen med front-enden (FPA'erne).
- 2) Processer som beskriver de fysiske disdrives.

3) processer som beskriver de fast opkoblede devicer på front-
enden.

4) processer som beskriver en fysisk disc opdeling i logiske.

Fælles for disse processer, er at de er entydigt defineret ved et
logisk devicenummer.

Tabellen viser nogle eksempler på devicenumre i monitoren.

devicenr.	0 = ptr
-	1 = ptp
-	2 = tty
-	3 = clock
-	5 = lpt
-	6 = disce
-	7 = disce
-	8 = disce
-	9 = disce
-	10 = mt0
-	11 = mt1
-	14 = fpa1
-	18 = fpa2
-	22 = fpa3
-	26 = disc beskrivelse
-	27 = autoload sys.disc
-	28 = disc1 beskrivelse
-	29 = autodump disc1
-	34 = evt. diablo 0
-	35 = evt. diablo 1
-	37 = evt. fd0
-	44 = fpa4
-	48 = fpa5
-	52 = fpa6
-	80 = sys.80

Op til Monitor rel. 8.1 kan der maksimalt være 24 processer i det interne lager ad gangen. Systemet tager 3 internals, så er der 21 internals til rådighed for andre processer. For hver proces der oprettes, bruges der en internal. Monitoren indeholder foruden de nødvendige instruktioner, create proces og remove proces, forskellige variable og tabeller til beskrivelse af datamatens tilstand. Beskrivelsen foreligger i form af en række procesbeskrivelser, idet datamatens aktivitet foregår i såkaldte processer. Til hver intern proces, findes en proces-beskrivelse der indeholder både faste og variable oplysninger. Procesnavn er en fast oplysning, mens tidsforbrug er en variabel oplysning, der løbende tælles sammen. Ajourføring foretages af monitoren. En proces kan kun råde over en del af dens egne ressourcer, inklusive discarealer til dens egne børn. En proces kan kun modificere, starte, stoppe og fjerne dens egne børn. Dataoverførelse mellem områder på BS. og det primære lager, varetages af monitoren. Da antallet af områder er varierende, findes der i monitoren en pulje arealprocesser, der kan tilknyttes de ønskede områder. Såfremt monitoren skal kommunikere med flere parallelle processer, er den i stand til at kommunikere med 6 procedurer kaldet send-message, wait-answer, wait-message, send-answer, wait-event og get-event. Ved brug af disse kommunikations-procedurer er man i stand til at sende data til andre processer. Ydre enheder er også at betragte som et barn af en proces.

En ydre enhed til CPU'en er repræsenteret ved en driver, et kanalprogram og disse to er identificeret ved et symbolsk navn. Kommunikationsprocedurerne kan så blive brugt til at initialisere blokke af data mellem forskellige processer, blandt andet til I/O operationer, eller til at etablere en kommunikation til terminaler.

En intern-proces repræsenterer en brugers rettigheder over systemressourcer, dvs. lagerområder, discplads og adgang til ydre enheder. Et procesnavn er en intern instruktion, en samling af rettigheder. Til hver interne proces, findes en procesbeskrivelse, der blandt andet indeholder oplysninger om procesnavn, placering i det interne lager, ressourcekrav, tidsforbrug, hierarkisk placering og processens tilstand. En proces kan være i 3 hovedtilstande, running, waiting og stop. Hvis en proces er i waiting, kan det være at der ventes på en datatransport, eller det kan være at processen er blevet swopped af operativsystemets prioriterings-strategi og endeligt af monitorens round robin. Hvert job får tildelt en timeslice på 25,6 millisekunder i lageret ad gangen. Samtlige processer der er i tilstanden running, får tildelt CPU-tid på skift af monitoren.

Der er således et nøje samarbejde mellem et operativsystem og den centrale del af operativsystemet.

Denne del udgør en selvstændig enhed, som kaldes Monitoren.

En række funktioner kaldes privilegerede funktioner og når disse skal udføres, må operativsystemet bede monitoren om at udføre operationen. Udskillelsen af operativsystemets kerne i en selvstændig enhed, bevirker at det er forholdsvis let at implementere nye operativsystemer til erstatning for eller som supplement til det eksisterende, da Monitoren kan understøtte ethvert operativsystem.

En disc er en ydre enhed, og set fra monitorens side, er den en extern proces, som beskriver en ydre enhed. Procesbeskrivelsen indeholder blandt andet, antallet af den pågældende ydre enhed, i dette eksempel antal fysiske disce. Derudover har monitoren oplysninger om devicenumre, størrelse på autoloaddel og størrelse på antal logiske disce.

Når en disc er monteret og meldt ready, bliver dens indgang overflyttet til hovedkataloget ved hjælp af monitor funktionen create entry og insert entry. Monitoren opererer ligeledes med funktioner som permanent entry, rename entry og set entry base i katalogerne.

Externe processer er en del af monitoren. De indeholder alle en proces-beskrivelse for hver ydre enhed der er repræsenteret på konfigurationen.

13.2 Monitorens areal proces-beskrivelser

13.2

En arealproces er en extern-proces, som f.eks. beskriver et areal på en disc. Input/output operationer mellem disce og lager bliver styret af monitorens arealprocesser. En arealproces er en ydre proces som beskriver en fil på et magnetbånd eller en fil på en disc. Monitoren indeholder en pulje arealprocesser, som er til rådighed for jobbet under udførelsestidspunktet.

13.2.1

13.2.1 Monitorens privilegerede funktioner

Privilegerede funktioner er input/output operationer, og instruktioner som ændrer monitorens beskyttelse af filer på baggrundslageret. Disse instruktioner kan kun udføres i monitor mode.

13.2.1.1 Monitorens beskyttelse af BS

13.2.1.1

Hver katalogindgang er nummereret med en integer fra 0 til 3 kaldet permkey. Monitoren undersøger om permkey er større end 2. Såfremt den er det, skal indgangen repræsenteres i hovedkataloget. Såfremt permkey er 0, 1 eller 2 er indgangen temporær eller login og forsvinder ved remove, finis, logout eller ved autoloading. Såfremt permkey er 3 er filen permanent og overlever remove, finis, logout eller autoloading.

Monitoren understøtter kataloghåndteringen ved hjælp af følgende kommandoer: change entry, create entry, lookup entry, lookup head and tail, permanent entry, remove entry, rename entry og set entry base.

En fil på discen består af et nummer på slicen, kædet sammen med dets indgang i kædetabellen. Kæden i kædetabellen skal starte i slice nummer 0. Hvis kæden er længere end nødvendigt, vil den blive skåret ned til den korrekte size. Maxsize af kataloget op til monitor 8.1 er 273 segmenter, som refererer til de 4096 katalogindgange.

Når en disc er monteret og denne indeholder et bikatalog, vil indgangen blive indsat i hovedkataloget med monitor funktion insert entry. Hvert dokument er inddelt i slices, som hver især kan deles med et antal segmenter. Hver slice korresponderer med et halvord i lageret, som tilsammen kaldes kædetabellen. Man kan selv bestemme på hvilken disc man vil have sit hovedkatalog, da dette blot skal angives i monitoren. Brugerne kan gemme programmer permanent på discene, der er ordnet som et arkiv af navngivne data-arealer. En fast del af hver disc er reserveret for katalogbeskrivelsen, som beskriver navne og steder på discarealet. I princippet er et hovedkatalog og et bikatalog det samme som en indholdsfortegnelse i en bog. Kataloget kan være delt i et uendeligt antal subkataloger inden i hinanden, og de har hver især sin specielle begrænsede adgang. Disse kataloger styres af monitoren.

Namekey bliver brugt af monitoren til at lede efter en entry i kataloget. Det er en ISO-værdi der udregnes, for at lettere gøre søgningen i kataloget.

En fysisk disc kan deles op i mange logiske disce. Det er en forudsætning at monitoren kender de logiske discs opdeling. Følgende eksempel viser en 66 MB disc der bruges som en fysisk disc. Anvendelig del af discen = 85344 segmenter. Ubrugt del af discen = 21 segmenter. Autoload-del af discen = 1050 segmenter. I alt samlet del af discen = 86415 segmenter. Når discen skal bruges, skrives en kitlabel på discen, ved hjælp af s kommandoen kitlabel. Dette gøres på følgende måde:

```
esc s
kitlabel 6 disc catdisc slow 131 42 2032
```

Der skrives en kitlabel på devicenr. 6 som er systemdiscen, katalognavnet er disc, devicet er slow (drum er fast), der laves et hovedkatalog på 131 segmenter, slicestørrelsen er 42 segmenter pr. slice og endelig det samlede antal slicer på discen.

Det næste eksempel viser en 66 MB disc opdelt i 2 logiske disce med en slice-størrelse på 3,42 segmenter pr. slice.

```
esc s
kitlabel 6 disc catdisc slow 131 3 1925
esc s
kitlabel 7 disc1 catdisc1 slow 131 42 1895
```

Autoload-del på system disc = 1050 segmenter. Data-del på system disc = 5775 segmenter. Data del af disc1 = 79590 segmenter.

Følgende syntaks er gældende for discs opdeling:

33 MB = 21 slicer = slicelængde * antal slicer = antal segmenter

66 MB = 42 - - - - - * - - - - -

248 MB = 105 eller 168 hvis hele discen benyttes som 1 fysisk device.

Basisprogrammet på RC8000 opdeles i følgende grupper:

- 1) file processoren
- 2) hjælpeprogrammer
- 3) operativsystemer

Denne opdeling af basisprogrammet i 3 grupper er værd at lægge mærke til af flere grunde. For det første afstikker opdelingen helt klare grænselinier i basisprogrammet, hvorved vedligeholdelse og udbygning forenkles. For det andet må man ved brugen af datamaten gøre sig klart, hvilke dele af basisprogrammet man ønsker at henvende sig til. For det tredje kan en række funktioner udføres af forskellige dele af basisprogrammet. For det fjerde udveksles der oplysninger mellem de forskellige programmer.

16.1 File processoren

16.1

Når et operativsystem har udvalgt et job, oprettes en intern proces hvori jobbet afvikles. Fra baggrundslageret indlæser operativsystemet herefter et program, der varetager afviklingen af de ønskede programmer. Dette afviklingsprogram kaldes for file processoren eller FP. FP anvendes under hele jobafviklingen til at indlæse og tolke styrekommandoer, aktiverer programmer og opdaterer tilstandsvariable. FP stiller ligeledes hjælpefunktioner til rådighed til blandt andet assembler-programmering. FP befinder sig permanent i det primære lager under hele jobafviklingen. Det vil sige at hvert job har en kopi af FP, men da det permanent kun optager nogle få halvord, betyder det rent ressourcemæssigt ingenting. FP læser altid én kommando ad gangen, uanset om det er en simpel eller en sammensat kommando. En kommando gemmes i kommandostakken, og først når samtlige simple kommandoer i kommandostakken er udført, indlæses næste kommando. FP kan ikke læse negative tal, derfor bruges to komplementet som omtalt i underafsnit 19.4.5.

Utility programmer kan inddeles i 3 hovedafsnit:

- 1) jobkontrol
- 2) kataloghåndtering
- 3) datahåndtering

Til denne gruppe henregnes programmer, der kan påvirke jobafviklingen på forskellige måder.

Endvidere omfatter gruppen programmer, der kommunikerer med operativsystemer.

Brugerne kan selv lave specielle utility programmer til specielle formål. Disse kan så anbringes på systemniveau, så andre også kan bruge dem.

Jobkontrol-programmer kan være afslutning af jobs, kommunikation mellem jobs og operatør, monteringer af bånd og printerkontrol.

1. Jobkontrol-programmer opdeles naturligt i følgende grupper:

- 1) Styring af input og output.
end, i, o.
- 2) Jobprogrammering.
if, mod, repeat.
- 3) Print tilmelding.
convert.
- 4) Jobtilmelding.
newjob og replace.
- 5) Operatørkommunikation.
opcom, opmess, kit, mount, release.

2. Kataloghåndtering

Med de kataloghåndterende programmer udføres de velkendte operationer såsom reservering, ændring og fjernelse af katalog-

indgange på pladelageret. Ligeledes bliver de brugt til søgende, sorterende og listende udskrifter af brugerfilindgange.

Gruppen omfatter ca. 22 programmer, hvoraf følgende omtales under utility programs.

Catsort, i, o, copy, head, headpunch, move, nextfile, clear, lookup, rename, scope og set kommando.

3. Datahåndtering

Ved datahåndtering forstås dataoverførsel, dataudskrivning, datamodifikation.

1) Dataoverførselsprogrammer er:

copy, load, move og save. Disse programmer omtales ligeledes under utility programs.

2) Dataudskrivningsprogrammer er:

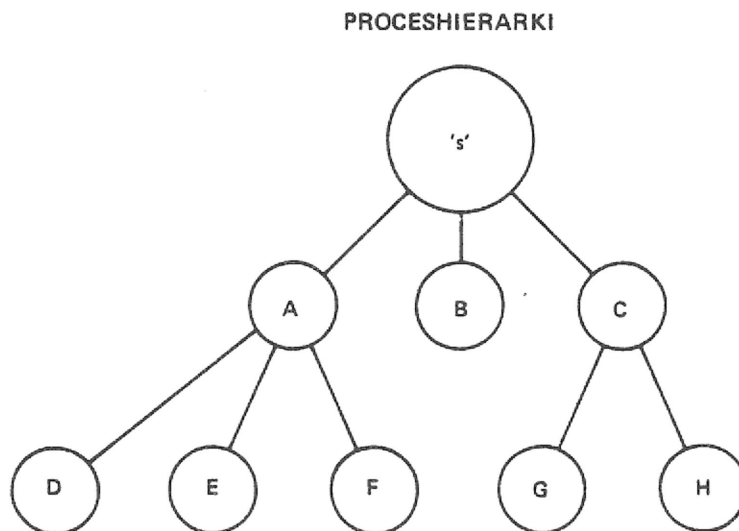
head og headpunch

Datamodifikationer er tekstredigeringsprogrammet EDIT der gør det muligt at modificere programtekster og data. EDIT rummer flere faciliteter end BOSS editoren.

Interne processer i RC8000 er opbygget efter hierarkisk struktur.

System s er grundstammen til alle andre processer. Operativsystem s er altså "far" for børneprocesserne A, B og C men er "bedstefar" til processerne D, E, F, G og H. Det vil sige, at hver proces kun kan råde over en del af forældreprocessens ressourcer.

Hvis vi betragter fig. 11 ser vi ind i det interne lager på RC8000. Her er det Monitoren der er forældreproces til System s, men operativsystemet må stadig henvende sig til Monitoren, når en privilegeret funktion skal udføres.



Figur 11: Proceshierarkiet i RC8000.

Ved et operativsystem forstås et styre- og overvågningsprogram, der har til opgave at lette og effektivisere brugen af datamaten. Disse opgaver kræver, at operativsystemet permanent må befinde sig i det primære lager, hvorved der naturligvis bliver tilsvarende mindre plads til rådighed for brugerprogrammer. Når man ønsker at få kørt et job på RC8000, må man have et operativsystem, og et jobudførelsessystem. Operativsystemer kan f.eks. være s, BOSS, SOS eller andre operativsystemer. Udførelsessystemet kaldes på RC8000 for file processoren.

Et operativsystem kontrollerer tildeling af CPU-tid, primær lager, BS. ressourcer og adgang til ydre enheder. Hver bruger må henvende sig til operativsystemet og bede om lov til at bruge dette. Brugerne logger ind til et brugerkatalog, som via dette administrerer ressourcer, køretid og lagerbeskyttelse. Operativsystemet må dog stadig henvende sig til monitoren, når en privilegeret funktion skal udføres.

Når et job skal afvikles af et operativsystem, oprettes en intern-proces, hvori programafviklingen foretages.

19.1 Operativsystem s

19.1

Operativsystem s' primære formål er dannelse, start, stop og fjernelse af interne proceser. s stiller ligeledes hjælperutiner til rådighed under programudførelsen. s er et primitivt operativsystem, som blandt andet ses ved behandlingen af ydre enheder. Tilmelding af en proces til operativsystem s foregår ved én af 3 slags kommandoer. All new eller job kommando. Oprettelse af en new proces giver et sæt standard værdier. All kommando giver alt hvad s maksimalt har tilbage, efter eventuelle processer er startet op. Job kommando giver et sæt individuelle parametre som er defineret i et brugerkatalog, kaldet s usercat. Når der køres med flere processer fra samme terminal eller konsol, bør man være varsom med anvendelse af remove og run kommando. Efter dannelse af en all new eller job proces, dannes en procesmodel i konsolbufferen. Konsolbufferen er blot et arbejdsområde til opstilling af en parameterliste til processen.

Der er 3 typer konsolbufferere:

- 1) privilegerede konsoller
- 2) buffere til fast konsol
- 3) buffere til reservation af en gruppe konsoller

Konsolbufferens indhold kan redefineres ved kald af nogle s kommandoer. Disse er blandt andre: prog, base area, buf, pr, pk, size, addr, temp og perm kommando.

Uafhængige af indholdet i konsolbufferen	Bruger indholdet i konsolbufferen	Ændrer indholdet i konsolbufferen
Call	Break	Addr
List	Bs	All
Lock	Create	Area
Max	Dump	Base
Modify	Exclude	Buf
Print	Include	Function
Replace	Init	Internal
Unlock	Load	Job
	Prio	Login
	Remove	Mode
	Run	New
	Start	Perm
	Stop	Proc
		Prog
		Project
		Size
		Temp
		User

Call og stop bruges når processen er startet. Include, exclude, load, start og create, kreerer en proces udfra konsolbufferens indhold. Break stopper den interne proces og dumper dens registre og genstarter processen. Print bruges til at printe dele af lageret ud. Clearcat kommando frigiver alle disodrives, kædetabeller og kataloger. Kit kommando inkluderer devicenummeret i BS systemet med det givne dokumentnavn. Kitlabel kommando bevirker at der

bliver lavet en kædehovedtabel og et tomt bikatalog. Maincat kommando definerer navn og size i segmenter på hovedkataloget. Oldcat kommando bevirker det samme som kit kommando, men tager alle disce med, som er kitlablet.

19.2 Operativsystem BOSS

19.2

Operativsystem BOSS har til opgave at gøre datamaten attraktiv og let at anvende, samt at sikre en god udnyttelse af maskinressourcer. BOSS varetager et operativsystems traditionelle opgaver som ressourceadministration, jobafvikling og jobovervågning. Sammenlignet med operativsystemet s er BOSS langt mere avanceret, da det styrer jobkø, i/o operationer, spooling, swapping og ressourceadministration. Operativsystem BOSS er bedst egnet til batch-jobs, da online-jobs bliver straffet i form af dårlig prioritering. På terminalerne kan der editeres og afvikles jobs.

Der kan startes job fra terminal, hulkort og strimmel. BOSS kan efter standardtrimning tage op til 20 jobs ind til jobkøen ad gangen og der kræves kun operatørindgreb i følgende situationer:

- tilmelding af offline jobs
- papirskift
- båndmonteringer
- betjening af ydre enheder

BOSS giver prioritering af jobs og foretager løbende account af forbrugt terminaltid, CPU-tid, antal båndmonteringer og forbrug af papir. Når BOSS er startet op som en all process, har s ingen

rådighed over ressourcer. BOSS bliver da en b rneproces af s, dvs. s bliver far. BOSS styrer nogle coroutiner som blandt andet styrer ydre enheder. For at effektivisere datamaten, kan BOSS simulere diverse ydre enheder, s ledes at overf relsen kun foreg r via et mellemlager, som bliver h ndteret af BOSS. Denne teknik kaldes for spooling. BOSS har indbygget en spoolfunktion, som g r at programmernes afvikling ikke h mmes af langsomme ydre enheder som f.eks. printere, datask rme og datatransporter til discarealer. Spooling virker i praksis p  den m de, at brugerprogrammets udskrift fanges af BOSS, der ved overl b gemmer det p  baggrundslageret, alt imens den styrer printningen af udskriften. Brugerprogrammet kan s ledes forts tte med andre opgaver, imens de tidligere dannede udskrifter printes. Normalt bliver udskriften f rst skrevet p  baggrundslageret, hvorefter BOSS, med kommandoen "convert", anmodes om at s rge for printningen af udskriften.

BOSS standard prioritering af jobs.

```
class 0 = nettok retid mindre end 30 sekunder
class 1 = nettok retid mindre end 3 minutter
class 2 = nettok retid mindre end 10 minutter
class 3 = nettok retid st rre end 10 minutter
```

BOSS' interne organisation: coroutiner og semaforer

BOSS arbejder internt med coroutiner og semaforer.

Coroutiner kan n rmest sammenlignes med processer, men er lettere at h ndtere, da de er interne i BOSS.

Der findes mange typer coroutiner:

- Pseudojobs til at betjene selve jobbene.
- Terminal-coroutiner.
- Device-coroutiner.
- Pageren, der styrer det virtuelle interne lager.
- Bankmanden, der uddeler ressourcer og devicer, prioriterer jobs og styrer swapning.

Coroutiner kommunikerer med hinanden via semaforer, som er et synkroniserings-element, der n rmest kan sammenlignes med tog-signaler (hvor ogs  navnet er l nt).

Prioriteringen af jobs og tildeling af ressourcer, sker så smidigt og kontrolleret, at der i praksis aldrig opstår "deadly embrace". Det vil sige, at f.eks. to jobs lammer hinanden, ved at begge kræver en ressource, som den anden allerede har.

19.2.1 BOSS brugerkatalog

19.2.1

Samtlige brugere, der kan benytte BOSS, er registreret i et brugerkatalog. Brugerne er opdelt i grupper, der kaldes projekter, og som identificeres ved et projektnummer. Hver bruger er udstyret med et brugernavn, der sammen med projektnummeret må anføres ved indlogging. Tilhørsforholdet til et projekt bestemmer brugers adgang til magnetbånd og baggrundslager. Medlemmer i det samme projekt kan således anvende hinandens magnetbånd, mens andre kan udelukkes fra dette. På baggrundslageret er et projekts ressourcer endnu stærkere beskyttet, idet kun medlemmer i projektet har adgang til ressourcerne. For hvert projekt indeholder brugerkataloget oplysning om hvilke ressourcer der er til rådighed for projektets medlemmer. Forskellige andre oplysninger vedrørende restriktioner i ressourceforbruget eller standardtildeling af ressourcer, kan yderligere være specificeret såvel for hele projektet, som for hver enkelt medlem. Foruden at forhindre misbrug, bidrager brugerkataloget således også til at lettere gøre brugen af BOSS.

BOSS indlogging

For at komme i forbindelse med operativsystem BOSS, må følgende indlogging foretages. Tast

esc boss

TYPE USER NAME AND PROJECT NUMBER	; BOSS skriver
elev01 3105	; brugeren skriver

IN 1983.01.01 12.00.00	; BOSS kvitterer for
BOSS REL. XXXX	; korrekt indtastning

Når brugeren ønsker at forlade terminalen tastes

logout

LOGGED IN XXXX OPERATIONS XXXX	; BOSS skriver
--------------------------------	----------------

19.2.2 BOSS kommandoer

19.2.2

disp

fortæller hvilke jobs der er kørende under BOSS.

disp login

fortæller hvem der er logget ind under BOSS.

disp convert

fortæller hvilke udskrifter der er under printning eller i kø.

hotnews

sender en aktuel meddelelse til alle som er logget ind.

mess <jobnavn>

sender en aktuel meddelelse til aktuel bruger eller alle.

kill <jobnavn>

fjerner et tilmeldt job.

kill <devno.>

stopper f.eks. printer 5 i en udskrift.

label kommando

skriver en label på et magnetbånd.

name kommando

navngiver et bånd uden label.

request

skriver en liste af messages fra BOSS som operatøren skal udføre.

reso max

skriver en liste over ledige ressourcer for klasse 3 jobs.

stop yes

lukker for tilmelding til jobkøen.

stop no

lukker op for jobkøen efter brug af stop yes.

close

lukker BOSS ned.

19.2.3 Nedlukning af BOSS

19.2.3

Ved nedlukning af BOSS, er det klogt at starte med at sende en hotnews til alle brugere, om at BOSS lukkes ned kl. xx.xx. Når alle brugere er informeret ved hjælp af hotnews og mess kommando, kan man starte nedlukningen af BOSS. Dette sker i første omgang med kommandoen stop yes. Herefter er der lukket for tilmelding af jobs, medens de jobs som allerede kører får lov til at køre færdig. Ved at bruge kommando disp og disp login, kan man hele tiden holde sig orienteret om afviklingen af jobkøen. Når denne er tom, og der heller ikke er printerudskrifter igang, kan man taste close. BOSS skriver "closed by operator". BOSS er nu i error og kan fjernes med remove kommandoen.

19.2.4 Fejlsituationer under BOSS, udtagning af bosstest

19.2.4

Såfremt BOSS går ned på en bossfault, skal operatøren foretage et dump af BOSS' testareal. Testarealet hedder bosstest, og skal ligge på discen hele tiden mens BOSS kører (ellers bliver der slet ikke lavet testoutput).

Arealet kan oprettes på følgende måde (inden BOSS binære program startes):

```

esc s
all x run
to x
bosstest=set 210      ; sæt areal på disc på 210 segmenter
scope user bosstest ; gør arealet permanent.
finis                ; process x fjerner sig selv.

                                ; derefter startes BOSS
esc s                    ; tast esc og s
all boss run            ; tastes
to boss                 ; BOSS svarer
bos                     ; kald BOSS' binære program.
                                ; Hvis BOSS nu går ned (bossfault), kan
                                ; testarealet gemmes til senere, således:
```



```

xxx bossfault 87      ; skrives af BOSS. Nummeret kan være 1 til
                        ; 199.

att s                  ; tast: esc og s

proc boss remove      ; tastes. Fjern BOSS processen en gang til
att s

all boss run           ; tastes. Start igen BOSS processen
to boss

gembf87 = set 210      ; tastes. Sæt et areal på discen, på 210
                        ; segmenter.

gembf87 = move bosstest ; tastes. Kopier bosstest over i
                        ; gembf87.

scope user gembf87     ; tastes. Gør arealet permanent
bos                    ; kald BOSS' binære program.

```

Hvis BOSS går ned gentagne gange, uden forklarlige årsager, bør man udfylde en fejlrapport og sende denne sammen med testoutput til REGNECENTRALEN.

19.3 Operativsystem MIPS/TS

19.3

Operativsystem MIPS/TS består af 3 moduler:

```

SOS   = Swapping Operating System
TEM   = Terminal Module (terminal kommunikation)
PRIMO = Printing Module (service modul)

```

19.4 Operativsystem SOS

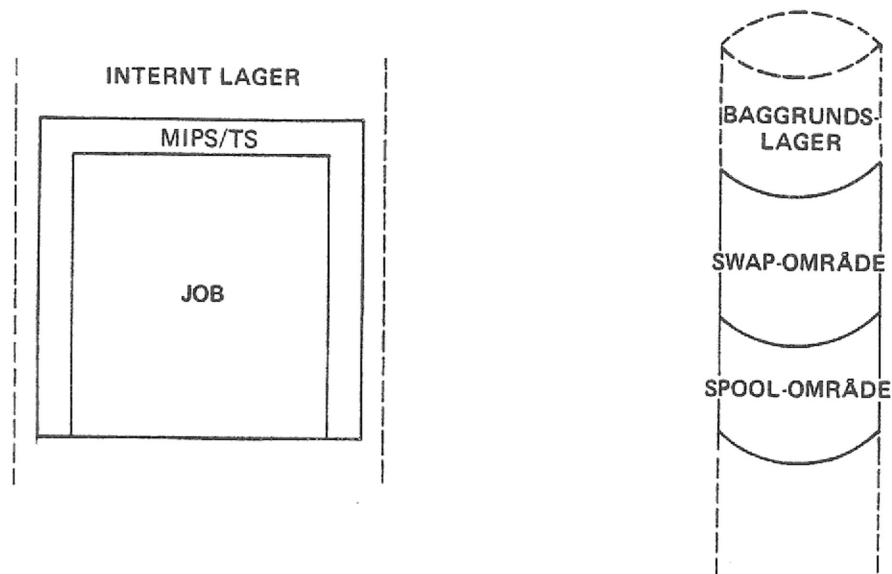
19.4

Operativsystem SOS er et multi-user Real Time operativsystem, som administrerer ressourcer, jobprioritering og ydre enheder. SOS prioriterer altid et onlinejob bedre end et batchjob. Hurtige input/output operationer belønnes af SOS ved en høj prioritet. SOS har i lighed med operativsystem BOSS, en indbygget spool-funktion, således at langsomme data-transporter til ydre enheder ikke belaster andre programmers afvikling.

Brugerprogrammer kan således fortsætte med andre opgaver, alt imens de tidligere dannede udskrifter printes.

Når et job har overskredet sin prioritering, smider SOS jobbet ud

på et areal på baggrundslageret, mens et andet job får køretid. Denne teknik kaldes for swapping.



Figur 12: MIPS/TS i det interne lager, med swap og spool område på discen.

19.4.1 SOS brugerkatalog

19.4.1

Samtlige brugere der benytter sig af operativsystem SOS, er registreret i et brugerkatalog.

For hver bruger indeholder brugerkataloget oplysninger om hvilke ressourcer der er til rådighed for den enkelte bruger.

Forskellige andre oplysninger vedrørende restriktioner i ressourceforbrug eller standard tildeling af ressourcer kan yderligere være specificeret for hver enkelt bruger.

Brugeren kan ligeledes være udstyret med et password, som anføres ved login rutinen.

Dette password kan til enhver tid ændres af brugeren.

Foruden at forhindre misbrug, bidrager brugerkataloget således også til at lettere gøre brugen af datamaten.

19.4.2 Indlogging til operativsystem SOS

19.4.2

For at komme i forbindelse med operativsystem SOS, skal man taste følgende:

esc sos

>RUN ELLER GO ELEV01 PASS < > ; tastes af bruger

16.00 SOS : ELEV01 ENROLLED ; SOS kvitterer for korrekt ID

Efter brug af terminal tastes:

finis

16.00 SOS : ELEV01 REMOVED AFTER FINIS

19.4.3 Operatørvejledning for SOS

19.4.3

Operatørens opgaver i forbindelse med systemets drift omfatter hovedsageligt:

- opstart
- indgreb vedrørende systemet og job
- indgreb vedrørende ydre enheder
- nedlukning
- behandling af systemfejl

19.4.4 Opstart af MIPS/TS

19.4.4

SOS operativsystem kan enten startes op som en all proces, efter opstart af TEM og PRIMO, eller SOS kan startes ved hjælp af job-kommando.

Ved opstart af SOS som en all proces tastes følgende:

esc s

ALL SOS PROG BSOS RUN

bsos er det binære sosprogram der aktiveres.

Opstart (SOS)Eksempel 1:

FROM SOS

02.58 SOS: MESSAGE SOS RELEASE: 3.1 820801 0

02.58 SOS: MESSAGE SOS CHILD RESOURCES

02.58 SOS: MESSAGE SOS INTERNAL 3

02.58 SOS: MESSAGE SOS BUF 147

02.58 SOS: MESSAGE SOS AREA 134

02.58 SOS: MESSAGE SOS SIZE 50688

02.58 SOS: MESSAGE SOS STARTED

Eksempel 1 viser opstart af SOS som en proces under operativsystemet s.

19.4.5 Opstart af MIPS/TS ved hjælp af s-brugerkatalog

19.4.5

Følgende standard jobfil vil kunne anvendes til at placere disse processer i SUSERCAT, hvorved oprettelse af processerne indskrænkes til en enkelt simpel kommando

```

susercat = set 10 disc
scope user susercat
scatop newcat 60 42 10 disc 0 0 disc1 0 0
scatop insert SOS 1.0    0 0 30.30 5.1984 0.0 8388609,
                        8388605 8388609 8388605 50000 bsos 8388609,
                        8388605 0 0 40 1000 0 0 0 0
scatop insert TEM 1.0    0 0 20.6 0.1056 0.0 8388609,
                        8388605 8388609 8388605 10000 btem,
                        8388609 8388605 0 0 20 800 0 0 0 0

scatop insert PRIMO 1.0  0 0 8.3 0.1984 0.0 8388609,
                        8388605 8388609 8388605 7500 bprimo,
                        8388609 8388605 0 0 20 800 0 0 0 0

```

Der gøres opmærksom på, at der antages en installation med 2 disenheder som specificeret i NEWCAT-kommandoen, og at såvel

size som behov for discplads i SOS-processen naturligvis afhænger af de programmer, der skal eksekveres under SOS.

Efter indsættelse i SUSERCAT startes processerne op ved brug af følgende kommandoer

JOB SOS

JOB TEM

JOB PRIMO

NB: FP kan ikke læse negative tal. Brug derfor to-komplementet til det negative tal.

F.eks. -833607 skrives som $16777216 - 833607 = \underline{8388609}$.

Indgreb under kørslen

Ved systemtrimningen defineres en operatørnøgle (et såkaldt password) som kan anvendes ved operatørindgreb. Sådanne indgreb accepteres kun når de foretages fra opstart-terminalen.

Indgreb vedrørende SOS-systemet

Ved at bruge følgende to kommandoer kan operatøren ændre systemets tilstand:

$$\left\{ \begin{array}{l} \text{LOCK} \\ \text{OPEN} \end{array} \right\} \text{ <operatørnøgle>}$$

LOCK-kommandoen bevirker at SOS afviser forsøg på at oprette job eller tilslutte terminaler til multiterminaljob. LOCK kan anvendes til at dræne systemet; SOS vil da udskrive en meddelelse på opstartterminalen, når det sidste job har forladt systemet, for at fortælle at systemet nu er tomt.

esc SOS

LOCK OPR

16.56 SOS: READY

OPEN-kommandoen bevirker at SOS accepterer oprettelse af job og tilslutning af terminaler til multi-terminaljob.



Hvis ALL-parameteren vælges, gælder kommandoen samtlige tilmeldte job.

STOP-kommandoen bevirker at det (de) pågældende job afbrydes på en sådan måde at jobafviklingen kan genoptages.

START-kommandoen aktiverer afbrudte job. Afbrydelsen kan for eksempel være en PAUSE-meddelelse der anmoder om et operatørindgreb.

Kommandoerne BREAK og KILL fjerner det (de) pågældende job. Ved BREAK får jobbet lov til at udskrive en fejlmeddelelse inden det fjernes.

esc SOS

KILL OPR ALL

16.59 SOS: READY

Eksemplet viser et operatørindgreb, der fjerner alle jobs.

Indgreb under kørslen (SOS)

Indgreb vedrørende ydre enheder

Som regel benytter SOS-job ikke ydre enheder der kræver operatørindgreb; alligevel vil der være et behov hos nogle brugere for at få lavet sikkerhedskopier på magnetbånd eller lignende. Job der benytter magnetbåndstationer og lignende ydre enheder må anmode om montering af dokumenter. Sådanne anmodninger udskrives på opstart-terminalen.

Når dokumentet er monteret, skal operatøren navngive den pågældende ydre enhed og det monterede dokument og derefter genstarte jobbet. Enheden og dokumentet navngives ved brug af CALL-komman-

doen:

CALL <enhedsnummer> <dokumentnavn>

START-kommandoen kan gives umiddelbart efter.

16.58 SOS: PAUSE RC MOUNT MT

esc SOS

CALL 10 MT START OPR RC

16.59 SOS: READY

Eksemplet viser montering af et magnetbånd og genstart af jobbet.

Nedlukning (SOS)

SOS drænes ved hjælp af LOCK-kommandoen. Såfremt systemet genererer testoutput, vil HALT også lukke testoutputfilen. Efter nedlukningen kan SOS-processen fjernes.

esc SOS

LOCK OPR

16.54 SOS: MESSAGE SOS SYSTEM EMPTY

16.54 SOS: READY

esc SOS

HALT OPR

16.55 SOS: PAUSE SOS SYSTEM CLOSED

Eksemplet viser nedlukning af systemet med forudgående dræning.

Behandling af systemfejl

Systemet vil kunne gå ned under kørslen på en af følgende måder:

- 1) På grund af en intern programfejl eller en transportfejl fra programarealet BSOS. Følgende fejlmeddelelse udskrives på opstart-terminalen:

PAUSE SOS ***FAULT

Bemærk at såfremt opstart-terminalen reserveres af andre pro-

cesser, for eksempel ved indlogging til BOSS, vil det ikke være muligt for SOS at udskrive sådanne fejlmeddelelser.

- 2) Uden at udskrive nogen fejlmeddelelse. Her bør man "breake" SOS-processen, hvorved den sidst genererede portion testoutput udskrives i testarealet, således:

```
esc S
PROC SOS BREAK
READY
```

- 3) En hård fejl på swaparealet umuliggør videre kørsel. Systemet standser efter at have udskrevet meddelelsen

```
MESSAGE SOS STATUS <statusord> SWPSOS
```

Ved ovenstående fejl bør man sætte sig i forbindelse med Teknisk Service, hvortil det genererede testoutput overgives.

I alle fejlsituationer bør man, såfremt systemet er trimmet med testoutput, "move" dette fra testarealet TSTSOS til et arbejdsareal, hvorfra det med testprogrammet TRACE kan udskrives for videre analyse, idet TSTSOS bliver overskrevet ved en ny systemstart.

TRACE-programmet genereres automatisk ved systeminstallation, og kaldes således:

```
TRACE <testareal>.<segmenter>
```

```
esc S
PROC SOS REMOVE NEW SOS RUN
READY
TO SOS
O LP
TRACE TSTSOS.10000 (alt udskrives)
O C
```


ATT S

PROC SOS REMOVE

(ny opstart)

Eksemplet viser udskrivning af testoutput inden genstart. Test-output er i øvrigt kun forståeligt for Teknisk Service.

Det skal endnu en gang nævnes at de anvendte navne SOS, SWPSOS og TSTSOS kun er eksempler; andre navne kan lige så vel bruges, blot de ikke overstiger 8 tegn, således at flere ens systemer med hvert sit navn samtidigt kan køre.

SOS meddelelser

SOS kan udskrive meddelelser af følgende typer:

- 1) Systemmeddelelser
- 2) Brugermødelelser
- 3) Meddelelser fra job

I typerne 1 og 2 kan der forekomme både normale meddelelser og fejlmeddelelser.

SOS-meddelelserne har følgende opbygning:

<time>.<minut> SOS: <meddelelse> (normal)

<time>.<minut> ***SOS: <meddelelse> (fejl)

Systemmeddelelserne kan vedrøre opstart eller systemfejl.

19.4.6 Operatørvejledning for TEM

19.4.6

Operatørens opgaver i forbindelse med systemets drift indskrænker sig til:

- opstart
- nedlukning
- behandling af systemfejl

Følgende beskrivelse forudsætter kørsel under operativsystemet s (ref. [1]). Køres TEM under et andet operativsystem, kan meddelelserne være anderledes.

Opstart

Ud fra de trimmede værdier beregner systemet under opstarten den nødvendige mængde ressourcer. Startes TEM-processen med flere eller færre ressourcer end det nødvendige antal, udskrives de optimale værdier på opstart-terminalen; mangler der ressourcer, indledes disse meddelelser med *** og kørslen afbrydes med meddelelsen ***INIT TROUBLE.

Selve programmet kaldes ved at skrive BTEM.

esc S

NEW TEM SIZE 10000 AREA 6 BUF 20

READY

esc S

BASE -8388607 8388605 PROG BTEM RUN

READY

MESSAGE TEM RELEASE: 2.3 820801 0

MESSAGE TEM SIZE 10462

MESSAGE TEM STARTED

Opstart af TEM ved hjælp af job kommando

TEM startes normalt op med jobkommando, hvor TEM's ressourcer så bliver hentet fra s usercat.

Følgende ressourcer er standard, men kan variere en del fra anlæg til anlæg. Tast følgende:

esc s

job tem ; operatør skriver

READY

MESSAGE TEM RELEASE: 2.3 820801 0 ; TEM skriver

MESSAGE TEM SIZE 10462

MESSAGE TEM AREA 7

MESSAGE TEM BUF 27

MESSAGE TEM STARTED

Nedlukning

Ved nedlukning af TEM, må operatøren sikre sig at der ikke er brugere under SOS. Når systemet er tomt, kan TEM fjernes med remove kommandoen.

```
esc s
PROC TEM REMOVE
READY
```

Behandling af systemfejl

Systemet vil kunne gå ned under kørslen på en af følgende måder:

- 1) På grund af en intern programfejl. Følgende fejlmeddelelse udskrives på opstart-terminalen:

```
PAUSE   TEM ***FAULT
```

TEM har et testareal, som hedder TEMTEST. Såfremt det ikke ligger på discen, skal det oprettes i en all proces.

```
esc s
all tem run                ; opret en all proces
to tem
temtest=set 42             ; afsæt areal på 42 segmenter
scope user temtest
```

```
esc S
```

(husk bredt papir i printer.)

```
PROC TEM REMOVE NEW TEM RUN
```

```
READY
```

```
TO TEM
```

```
O LP
```

```
TRACE TEMTEST.10000      (alt udskrives)
```

```
O C
```

```
esc S
```

```
PROC TEM REMOVE
```

```
---
```

(ny opstart)

Eksemplet viser udskrivning af testoutput inden genstart. Test-output er i øvrigt kun forståeligt for Teknisk Service.

Bemærk at det ikke vil være muligt for TEM at udskrive fejlmeddelelser, såfremt opstart-terminalen reserveres af andre processer, for eksempel ved indlogging til BOSS.

19.4.7 Operatørvejledning for PRIMO

19.4.7

Operatørens opgaver i forbindelse med systemets drift er meget begrænsede og omfatter:

- opstart
- indgreb under kørslen
- nedlukning
- behandling af systemfejl

Følgende beskrivelse forudsætter kørsel under operativsystemet s (ref. [1]). Køres PRIMO under et andet operativsystem, kan kommandoerne og meddelelserne være anderledes.

Opstart

PRIMO startes lettest med job kommandoen, hvor ressourcekravene er specificeret i s usercat.

Tast:

esc s

JOB PRIMO

READY

```
MESSAGE PRIMO SIZE 7342      ; PRIMO skriver
MESSAGE PRIMO AREA   3      ; PRIMO skriver
MESSAGE PRIMO BUF    8
MESSAGE PRIMO STARTED      ; PRIMO skriver
```

Indgreb under kørslen

Operatøren har kun få muligheder for indgreb under kørslen og kun i forbindelse med lokale ydre enheder.

Når en datatransport til en lokal lineskriver har et andet kø-kriterie (for eksempel papirtype; indledningsvis er dette kriterie tomt) end den foregående, standses transporten og følgende meddelelse udskrives på hovedkonsollen:

```
PRIMO: CHANGE <procesnavn> <gruppenavn> = <kønavn>
```

Lad os antage at en lineskriver med procesnavnet PRINTER har monteret papir i formatet A4. Følgende meddelelse udskrives nu:

```
PRIMO: CHANGE PRINTER PAPER = MONITOR
```

Det vil sige at de ønskes skrevet ud på papir i formatet MONITOR. Operatøren bør nu skifte papir og skrive:

```
esc PRIMO
= PRINTER
```

Udskrivningen vil nu blive genoptaget antagende at det ønskede format er monteret.

Nedlukning

Når PRIMO skal lukkes ned, skal operatøren sikre sig at der ikke er brugere under SOS eller andet operativsystem som benytter sig af PRIMO.

Når dette er gjort, kan PRIMO fjernes med REMOVE kommandoen.

```
esc s
PROC PRIMO REMOVE
READY
```

Hermed standses straks enhver funktion i PRIMO og samtlige køer fjernes.

Behandling af systemfejl

Systemet vil kunne gå ned under kørslen på en af følgende måder:

1) På grund af en intern programfejl. Følgende fejlmeddelelse ud-

skrives på opstart-terminalen:

```
PAUSE    PRIMO ***FAULT
```

- 2) Uden at udskrive nogen fejlmeddelelse. Her bør man "breake" PRIMO-processen, hvorved den sidst genererede portion testoutput udskrives i testarealet, således:

```
esc s
PROC PRIMO BREAK
READY
```

Såfremt PRIMO ikke har noget testareal, kan dette oprettes i en all proces, ved at taste:

```
esc s
ALL PRIMO RUN           ; lav en all process
TO PRIMO
PRIMOTEST=SET 100       ; afsæt areal på discen
SCOPE USER PRIMOTEST   ; gør areal permanent
```

Eksempel:

```
esc s
PROC PRIMO REMOVE NEW PRIMO RUN
READY
TO PRIMO
O LP
TRACE PRIMOTEST.10000    (alt udskrives)
O C
```

```
esc s
PROC PRIMO REMOVE
---                      (ny opstart)
```

Eksemplet viser udskrivning af testoutput inden genstart. Testoutput er i øvrigt kun forståeligt for Teknisk Service.

Bemærk at det ikke vil være muligt for PRIMO at udskrive fejlmeddelelser, såfremt opstart-terminalen reserveres af andre processer, for eksempel ved indlogging til BOSS.

20. BACK-UP AF DATA PÅ MAGNETBÅND, DISKETTE OG DISCE

20.

Back-up af data på magnetbånd

Formålet med at tage back-up af data, er hurtigt at kunne retablere tab af data efter menneskelige eller maskinelle fejl.

Back-up af data bør foretages mindst en gang om ugen, da en back-up ellers ikke vil være relevant. Sammen med back-up'en, bør udskrifter fra kørslen af back-up opbevares i brandskabe eller eksternt ude i byen.

20.1 SAVE/LOAD magnetbånd

20.1

SAVE

Med SAVE-programmet kan såvel katalogindgange som områder gemmes på magnetbånd.

PROGRAMKALD

save <bånd spec.> <dump spec.>

$$\langle \text{bånd spec.} \rangle ::= \langle \text{båndnavn} \rangle . \langle \text{filnr.} \rangle \left\{ .\text{label} . \langle \text{labelspec.} \rangle \right\}_0^1$$

$$\langle \text{filnr.} \rangle ::= \left\{ \begin{array}{l} \text{last} \\ \langle \text{heltal} \rangle \end{array} \right\}$$

$$\langle \text{labelspec.} \rangle ::= \langle \text{navn} \rangle$$

$$\langle \text{dump spec.} \rangle ::= \left\{ \text{scope} . \langle \text{scopeangivelse} \rangle \right\}_0^4 \left\{ \langle \text{navn} \rangle \right\}_0^*$$

$$\langle \text{scopeangivelse} \rangle ::= \left\{ \begin{array}{l} \text{temp} \\ \text{login} \\ \text{user} \\ \text{project} \\ \text{perm} \\ \text{all} \end{array} \right\}$$
FUNKTION

De i dumpspecifikationen anførte katalogindgange og områder gemmes på det specificerede magnetbånd. Båndspecifikationen angiver båndnavn og filnummer og rummer muligvis en LABEL-parameter. SAVE undersøger altid, om magnetbåndsfilen tidligere er blevet skrevet

med SAVE-programmet, og hvis det er tilfældet, overskrives filen kun, såfremt LABEL-parameteren optræder i programkaldet. Den specificerede dumplabel eller en standard label skrives på filen efterfulgt af det aktuelle tidspunkt.

$$\langle \text{mount param} \rangle ::= \left\{ \begin{array}{l} \text{Mountspec.} \langle \text{device no} \rangle \\ \text{release.} \left\{ \begin{array}{l} \text{yes} \\ \text{no} \end{array} \right\} \end{array} \right\} \begin{array}{l} 1 \\ 0 \end{array} \left\{ \langle \text{modekind} \rangle. \right\} \begin{array}{l} 1 \\ 0 \end{array}$$

Hvis "mountparameter" ikke specificeres, vil programmet anvende standardset af parametrene for mag.båndstation, "modekind" (med mindre "mtnavn" er filedescriptor), for at slutte af med at frigive båndstationen.

Eksempel:

Mountspec.no.nrz.release.no
release.no

$$\langle \text{tape param} \rangle ::= \langle \text{tape name} \rangle. \langle \text{file no} \rangle. \left\{ \langle \text{tape name for next volume} \rangle \right\} \begin{array}{l} 9 \\ 0 \end{array} \left\{ \text{label.} \langle \text{dump label name} \rangle \right\} \begin{array}{l} 1 \\ 0 \end{array}$$

$\langle \text{tape name} \rangle ::= \langle \text{filedescriptor af filer på magnetbånd} \rangle$
 $\langle \text{magnetbåndets navn} \rangle$
 last
 $\langle \text{file no} \rangle ::= \langle \text{integer} \rangle$

Hvis $\langle \text{file no} \rangle = \text{last}$, vil den første fil, som ikke indeholder "version label", blive brugt. Af hensyn til køretid er brugen af integer parameter tilrådelig.

Ønskes mere end en sikkerhedskopi, skal man blot angive samtlige "tape parametre". Båndene bliver i så fald identiske på nær $\langle \text{tape name} \rangle$.

$$\langle \text{special param} \rangle ::= \left\{ \begin{array}{l} \text{list.} \left\{ \begin{array}{l} \text{yes} \\ \text{no} \\ \text{name} \end{array} \right\} \\ \text{segm.} \langle \text{integer} \rangle \end{array} \right\}$$

List.yes standard; fuld listning
 List.no ingen listning
 List.name kun navne på de kopierede filer listes

 Segm.<integer> integer skal være mellem 1 og 9. (Segment/
 blok)
 Standard er 1 segment.

$$\begin{aligned}
 \langle \text{save spec} \rangle ::= & \left\{ \begin{array}{l} \langle \text{modifiers} \rangle \\ \langle \text{kitspec} \rangle \\ \langle \text{entryspec} \rangle \end{array} \right\} \\
 \langle \text{modifiers} \rangle ::= & \left\{ \begin{array}{l} \text{change kit.} \langle \text{bs device navn} \rangle . \langle \text{bs device spec} \rangle \\ \langle \text{newscope} \rangle . \langle \text{newscope spec} \rangle \end{array} \right\}
 \end{aligned}$$

<bs device navn> det aktuelle kit
 <bs device spec> det <savede> kit

$$\langle \text{kit spec} \rangle ::= \text{kit.} \left\{ \begin{array}{l} \text{main} \\ \langle \text{bs device navn} \rangle \end{array} \right\} \text{ det "savede" kit}$$

$$\langle \text{entry spec} \rangle ::= \left\{ \begin{array}{l} \langle \text{navn} \rangle \\ \langle \text{navn} \rangle . \text{scope} . \langle \text{scopespec} \rangle \\ \text{docname} . \langle \text{dokumentnavn} \rangle \\ \text{docname} . \langle \text{dok.navn} \rangle . \text{scope} . \langle \text{scopespec} \rangle \\ \text{scope} . \langle \text{scopespec} \rangle \\ \text{in} . \langle \text{bs-filer} \rangle \end{array} \right\}$$

In.<bs-filer> ::= Filer, nævnt i <bs-filer>, "saves". I denne fil
 (<bs-filer>) er <nl> tilladt som separator imellem parametrene.

Når LAST angiver filnummeret, benyttes den første fil på båndet,
 der ikke tidligere er blevet anvendt af SAVE.

Dumpspecifikationen angiver, hvilke katalogindgange og områder,
 der skal gemmes. Såvel SCOPE-puljer som enkelt navne kan angives.
 Udelades dumpspecifikationen, gemmes LOGIN-puljen.

Eksempel:

```
save mt123456.1 scope.user
```

USER-puljen gemmes på fil 1 på MT123456.

Eksempel:

```
save mt123456.last
```

```
save mt123456.2.label.rcdump
```

```
save mt123456.last data1 data2 data3
```

FEJL

Efter fremkomsten af én af de tre første udskrifter i nedenstående liste, afbrydes programmet, og båndet er uberørt.

FEJLUDSKRIFTER

```
*** SAVE ERROR IN TAPEPARAM <parametre>
```

De fejlagtige og de resterende båndparametre udskrives, og programmet afbrydes.

```
*** SAVE ERROR PARAM <parametre>
```

De fejlagtige og de resterende parametre udskrives, og programmet afbrydes.

```
*** SAVE DUMPLABEL <specifikation>
```

Den specificerede dumplabel er ulovlig.

```
<navn> ENTRY IN USE
```

Den specificerede katalogindgang blev benyttet af en anden og kunne derfor ikke gemmes.

```
<navn> NOT ALLOWED
```

Den specificerede katalogindgang må ikke gemmes.

```
*** NOT FOUND: <navn>
```

Den angivne katalogindgang findes ikke.

```
*** SAVE NOT OK <antal fejl>
```

Denne udskrift fremkommer, når der er registreret fejl.

20.2 SAVE af systemfiler på disc

20.2

```

esc s
Job errorsnoop      : opsamlers fejl på interne/eksterne processer
Job discud          : base -8388607 8388605
to discud
O lp                : output til std printer
Head 1
Mode list.yes
Save mtxxxx.1.label.system segm.4 scope.user
Pause discud mount mtxxxx
esc s
Call xx mtxxxx start
Message discud release mtxxxx
O C
Finis
Pause discud finis 0

```

Save af brugerfilen på disc

```

esc s
Job savedisc1       : base 0 8388605
To savedisc1
O LP
Head 1
Mode list.yes
Save mtxxxx.2.label.disc segm.4 kit.disc scope.perm
Message savedisc1 release mtxxxx
O C
Finis

```

Load

Med LOAD-programmet retableres katalogindgange og områder, der er gemt med SAVE-programmet.

Eksempel:

```
load mt123456.1
load mt123456.last
load mt123456.1 scope.temp
load mt123456.1 scope.user data1 data2
load mt123456.1 load.no
load mt123456.last survey.yes
```

PROGRAMKALD

```
load <båndspec.> <dumpspec.> <speciel param>
```

```
<båndspec.>::= <båndnavn>.<filnr.>
```

```
<filnr.>::= { last
              <heltal> }
```

```
<dumpspec.>::= { scope.<scopeangivelse> }04 { <navn> }0*
```

```
<scopeangivelse>::= { temp
                      login
                      user
                      project
                      perm
                      all }
```

```
<speciel param>::= { load.no }01 { survey.yes }01
```

FUNKTION

De i dumpspecifikationen anførte katalogindgange og områder re-tableres fra det specificerede magnetbånd. I båndspecifikationen anføres såvel båndnavn som filnummer. Udelades dumpspecifikationen, re-tableres hele filen, ellers re-tableres de angivne SCOPE-puljer, katalogindgange og områder. En oversigt over de retable-rede katalogindgange udskrives på CURRENT OUTPUT FILE.

Når LOAD.NO parameteren anføres, foretages ingen re-tablering, men kørselsrapporten udskrives.

Anvendes parameteren SURVEY.YES, udskrives oversigter over indhold et af samtlige filer fra 1 til og med den angivne. Ingen filer loades.

FEJL

Efter fremkomsten af én af de tre første udskrifter i nedenstående liste afbrydes programmet.

FEJLUDSKRIFTER

*** LOAD: ERROR IN TAPEPARAM <parametre>

En fejlramt båndparameter samt de resterende parametre udskrives.

*** LOAD: ERROR IN PARAM <parametre>

En fejlramt parameter samt de resterende parametre udskrives.

*** LOAD: NO DUMPLABEL ON FILE <filnr.>

Den specificerede fil er ikke genereret med SAVE-programmet.

*** <navn> MONITOR <heltal> RESULT <heltal> <årstal>

Den specificerede katalogindgang kunne ikke retableres af den anførte grund.

*** NOT FOUND <navn>

Den anførte katalogindgang findes ikke på filen.

*** LOAD NOT OK <antal fejl>

Denne udskrift fremkommer, når der er registreret fejl.

SAVE/LOAD Diskette

SAVE:

Med SAVE-programmet kan såvel katalogindgange som områder gemmes på diskette.

PROGRAMKALD:

fdsave <diskette spec.> <dump spec.>

<diskette spec.>::= <diskettenavn>. <filnr.>

<diskettenavn>::= $\left\{ \begin{array}{l} \text{max 6 tegn} \\ \text{max 6 cifre} \end{array} \right\}$

$$\begin{aligned} \langle \text{filnr.} \rangle ::= & \text{heltal min.1} \left\{ \begin{array}{ll} \text{max 45} & \text{dobbeltsidet} \\ \text{max 19} & \text{enkeltsidet} \end{array} \right\} \\ \langle \text{dump spec.} \rangle ::= & \left\{ \langle \text{navn} \rangle \right\}_1^* \end{aligned}$$
FUNKTION:

De i dumpspecifikationen anførte katalogindgange og områder gemmes på den specificerede diskette og specificerede fil.

Eksempel: fdsave voll.3 pip fil0 text om

Indgangene med navnene pip, fil0, text og om bliver savet på diskette voll fil3.

Vedr. fejludskrifter refereres til pågældende manual.

LOAD:

Med LOAD-programmet retableres katalogindgange og områder, der er gemt med save.

PROGRAMKALD:

$$\begin{aligned} \text{fdload} & \langle \text{diskette spec.} \rangle \langle \text{dump spec.} \rangle \\ \langle \text{diskette spec.} \rangle ::= & \langle \text{diskettenavn} \rangle . \langle \text{filnr.} \rangle \\ \langle \text{diskettenavn} \rangle ::= & \left\{ \begin{array}{l} \text{max 6 tegn} \\ \text{max 6 cifre} \end{array} \right\} \\ \langle \text{filnr.} \rangle ::= & \text{heltal min.1} \left\{ \begin{array}{ll} \text{max 45} & \text{dobbeltsidet} \\ \text{max 19} & \text{enkeltsidet} \end{array} \right\} \\ \langle \text{dump spec.} \rangle ::= & \left\{ \langle \text{navn} \rangle \right\}_0^* \end{aligned}$$
FUNKTION:

De i dumpspecifikationen anførte katalogindgange og områder retaberes fra den specificerede diskette. Udelades dumpspecifikationen retableres hele filen, ellers retableres de angivne katalogindgange og områder.

Eksempel: fdload vol1.3

Alle indgange og områder i fil3 vil blive retableret.

Eksempel: fdload vol2.1 entry5

Indgangen med navnet entry5 i fil1 vil blive retableret.

Navngivning af diskette

FD-INIT

For at kunne bruge en diskette til FD-SAVE skal der på disketten kreeres en volume-label med visse oplysninger. Dette bevirker også, at det tidligere indhold af disketten slettes.

PROGRAMKALD

$$\text{fdinit } \langle \text{diskettenavn} \rangle \quad \langle \text{special-param} \rangle \quad \left. \vphantom{\begin{matrix} \text{diskettenavn} \\ \text{special-param} \end{matrix}} \right\} \begin{matrix} 4 \\ 0 \end{matrix}$$

$$\langle \text{diskettenavn} \rangle ::= \begin{cases} \text{max 6 tegn} \\ \text{max 6 cifre} \end{cases}$$

$$\langle \text{special-param} \rangle ::= \left\{ \begin{array}{l} \text{trackformat.} \left\{ \begin{matrix} 0 \\ 1 \\ 2 \end{matrix} \right\} \end{array} \right\} \begin{matrix} 1 \\ 0 \end{matrix}$$

$$\left\{ \begin{array}{l} \text{doublesided.} \left\{ \begin{matrix} \text{yes} \\ \text{no} \end{matrix} \right\} \end{array} \right\} \begin{matrix} 1 \\ 0 \end{matrix}$$

FUNKTION

Diskettens volume-label bliver initialiseret med de i kaldet givne parametre.

$\langle \text{diskettenavn} \rangle$ er det navn, brugeren vil have tilknyttet den pågældende diskette.

trackformat	0, betyder 128 tegn pr. sektor
	1, - 256 - - -
	2, - 512 - - -

doublesided	angiver, om disketten er enkeltsidet eller dobbeltsidet.
-------------	--

Standard er	trackformat.0
	doublesided.no

Eksempel på navngivning af disketter

```
fdinit <diskettenr.> trackf.2 doublesided.yes return
```

```
from s
message iris mount fd0<diskettenr.>
pause iris enable fd0<diskettenr.>
```

```
esc s
call 30 fd0<diskettenr.> start ; tryk return
ready
```

```
from iris
```

```
flexible disc initialized:
```

```
volume id      : <diskettenr.>
owner id       :
trackformat    : 2
doublesided    : yes
conversion
from s
message iris suspend fd0<diskettenr.>
```

```
from iris
: no
```

```
end        31
```

Simpel brug af FD survey

Fdsurvey udskriver indholdet af en diskette, hvor filerne er gemt med FD-save.

eks. på brug af Fd-survey.

esc s

call <devicenr.> <diskettenavn>

esc s

new me run ; opret en proces

to me

fdsurvey <diskettenavn> ; giv følgende FP parametre
; output bliver en listning af
; alle filer incl. label.

fdsurvey <diskettenavn> <filnr.> max $\left\{ \begin{array}{l} 19 \text{ for enkelt-sided} \\ 45 \text{ for dobbelt-sided} \end{array} \right\}$

SAVE/LOAD discpackSAVE

Med SAVE-programmet kan såvel katalogindgange som områder gemmes på discpacks.

PROGRAMKALD

discopy save <kit spec.> $\left\{ \text{list.} \left\{ \begin{array}{l} \text{yes} \\ \text{no} \end{array} \right\} \right\}^1_0$ <dump spec.>

<kit spec.>::= to.<devno.>

$$\langle \text{dump spec.} \rangle ::= \left\{ \text{scope.} \langle \text{scopeangivelse} \rangle \right\}_0^1 \left\{ \langle \text{navn} \rangle \right\}_0^*$$

$$\langle \text{scopeangivelse} \rangle ::= \left\{ \begin{array}{l} \text{system} \\ \text{project} \\ \text{user} \end{array} \right\}$$

FUNKTION

De i dumpspecifikationen anførte katalogindgange og områder gemmes på den logiske enhed med nummeret $\langle \text{devno} \rangle$. TO-device må ikke være inkluderet i BS-systemet, og de indgange, der skal kopieres, må ikke eksistere i bikataloget på device $\langle \text{devno} \rangle$.

Anvendes parameteren `list.yes`, udskrives oversigt på current output file over de kopierede katalogindgange.

Dumpspecifikationen angiver, hvilke katalogindgange og områder, der skal kopieres. Såvel SCOPE-puljer som enkelt navne kan angives. Udelades dumpspecifikationen, kopieres SYSTEM-puljen.

Eksempel:

`save to. 7 scope.user list.yes`

USER-puljen kopieres til device 7. En udskrift over de kopierede indgange leveres.

Fejludskrifter er selvforklarende.

LOAD

Med LOAD-programmet kan såvel katalogindgange som områder retableres fra discpacks.

PROGRAMKALD

$$\text{disccopy load } \langle \text{kit spec.} \rangle \left\{ \text{list.} \left\{ \begin{array}{l} \text{yes} \\ \text{no} \end{array} \right\} \right\}_0^1 \langle \text{dump spec.} \rangle$$

$$\langle \text{kit spec.} \rangle ::= \left\{ \begin{array}{ll} \text{from} & \langle \text{devno} \rangle \\ \text{to} & \langle \text{docname} \rangle \end{array} \right\}_2^2$$

$$\langle \text{dump spec.} \rangle ::= \left\{ \text{scope.} \langle \text{scopeangivelse} \rangle \right\}_0^1 \left\{ \langle \text{navn} \rangle \right\}_0^*$$

$$\langle \text{scopeangivelse} \rangle ::= \left\{ \begin{array}{l} \text{system} \\ \text{project} \\ \text{user} \end{array} \right\}$$

FUNKTION

De i dumpspecifikationen anførte katalogindgange og områder retableres fra den logiske enhed med nummeret <devno> til device med navnet <docname>. FROM-device må ikke være inkluderet i BS-systemet. Hvis indgangene eksisterer i forvejen, bliver de overskrevet.

Anvendes parameteren list.yes, udskrives oversigt på current output file over de kopierede katalogindgange.

Eksempel:

```
load from.6 to.disc1 scope.user
```

USER-puljen på den discpack, der er monteret på device6, bliver loaded til disc1.

Fejludskrifter er selvforklarende.

Eksempel på brug af Disc-copy typein

Monter masterdisc i discdrive 0 og systemdiscen i discdrive 1 med plug 1 ude af drivet.

```
-- Autoload
```

```
Lav en all-proces
```

```
-- all x run
```

```
to x
```

```
-- disccopy typein
```

```
tryk 'write project' ind på discdrive 0.
```

```
-- bin from.26.all to.28.all
```

```
udskrift fra disccopy (43155 segm copied).
```

```
-- end
```

```
-- finis
```

```
pause x finis FP
```

Afmonter masterdisc og sæt den på plads. Monter systemdisc'en i discdrive 0 og datadisc i discdrive 1 og systemet er klar til brug.

21. UTILITY PROGRAMMER

21.

Catsort (utility part 2) = katalogsortering.

eks. 1 catsort docname.disc system.yes

Udskrifter bliver alle entries i hovedkataloget med dokumentnavn disc, sorteret efter base og entrynavn.

eks. 2 catsort subcat.disc1 system.yes

Udskriften bliver alle entries i bikataloget med dokumentnavn disc1, sorteret efter base og entrynavn.

21.1 Filedescriptors

21.1

Pip = set mto mt123456 0 3
i pip

21.2 Utility copy

21.2

Copy kan kopiere en tekst over i en anden fil på en ydre enhed.

eks. tpe = copy data.
filen data bliver udhullet på punchen i lige paritet.

21.3 Utility head

21.3

Head laver et sideskift på en linieskriver.

eks. o lp
head 1

21.4 Utility headpunch

21.4

Headpunch laver en læsbar skrift på en strimmel.

21.5 Utility move

21.5

Bevirker blokvis kopiering af filer fra disc eller magnetbånd. Hvis mess.yes vil det bevirke at output udskrives med antal bytes og checksum.

eks. pip = move mess.yes pap; indholdet af pap bliver flyttet over i pip, og der laves checksum over antal overførte bytes.

21.6 Utility nextfile

21.6

Nextfile adderer 1 til filnummer.

Eks. t = set mto mt123456 0 3
nextfile t

21.7 Utility rename

21.7

Rename pip.pap

Rename ændrer filnavn fra gammel/ny.

21.8 LOOKUP kommando

21.8

Med lookup kommandoen udskrives specificerede katalogindgange.

Ved brug af LOOKUP kommandoen fås følgende oplysninger:

- 1) hoved (head) placering, klasse og navn
- 2) krop (tail)

I kataloget findes der for hvert område, ét individ (record) der beskriver området.

Et individ i kataloget kaldes en katalogindgang (catalog entry) og den består af 2 dele, nemlig head og tail.

Eks. 1

```
<navn> = set <størrelse> <dokumentnavn> <dato> <diverse> ; scope
<første slice> <navnenøgle> <katalognøgle> <base nedre> <base øvre>
```

```
Data=set 18 disc d.830101 16.45.50 0 0 0 0 ; user
1207 72 3 5 100 ;
```

I hovedet står følgende oplysninger:

Første slice i katalogindgangen udpeger nummeret på den første slice, der indgår i et område. Sammenkædningen af de forskellige slices håndteres af monitoren.

Navnenøgle

Oplysningen anvendes af monitoren ved opslag i kataloget.

Katalognøgle

Scopeangivelsen er bestemt af katalognøglen og basen.

Basen

Basen består af et talpar, der angiver den nedre og den øvre basegrænse. Scope angivelsen er bestemt af basen og katalognøglen.

I kroppen består oplysningerne af:

navn

Det skal overholde følgende regler:

Det må højst bestå af 11 tegn (bogstaver eller tal).

Det skal indledes med et bogstav.

Der må kun anvendes små bogstaver.

Størrelsen.

Angiver antallet af segmenter som filen fylder på pladelageret.

Eksempel på brug af lookup kommandoen

```
lookup data
```

```
lookup data1 data2
```

FEJLUDSKRIFTER

```
*** LOOKUP CONNECT <udfil>
```

Den specificerede udfil kunne ikke tilknyttes, og i stedet anvendes CURRENT OUTPUT FILE.

*** LOOKUP PARAM <parameter>

Parameterfejl. Den resterende del af parameterlisten ignoreres

*** LOOKUP <navn> UNKNOWN

Der findes ingen tilgængelige katalogindgange med det anførte navn. De øvrige parametre behandles normalt.

*** LOOKUP <navn> NO RESOURCES

Programmet afbrydes, da der er for få arealprocesser til rådighed.

21.9 SET

21.9

Med SET-programmet kan en temporær katalogindgang oprettes.

PROGRAMKALD

<navn>= set $\left\{ \begin{array}{l} \langle \text{størrelse} \rangle \\ \langle \text{modekind} \rangle \end{array} \right\} \left\{ \langle \text{dokumentnavn} \rangle \right\}$
 $\left\{ \langle \text{dato} \rangle \langle \text{filnr.} \rangle \right\}_2^1$

FUNKTION

Parameterlisten læses og fortolkes, og en temporær katalogindgang med det specificerede navn og indhold søges oprettet. Hvis det ikke er muligt, fordi der i forvejen findes en temporær katalogindgang med det specificerede navn, ændres denne i overensstemmelse med parameterlisten.

Første parameter kan være områdestørrelse eller MODEKIND-angivelse, og de øvrige parametre fortolkes i overensstemmelse hermed.

Udeblivelsesværdien for parametre er 0, for dato dog d.0, dvs. det aktuelle tidspunkt.

FEJL

I tilfælde af fejl oprettes eller ændres ingen katalogindgang.

Eksempel

data = set 18

data = set 18 dk123456

data = set mto mt123456 0 1

21.10 I

21.10

Med I-programmet vælges CURRENT INPUT FILE.

PROGRAMKALD

i <filnavn>

FUNKTION

CURRENT INPUT FILE stakkes, og der gemmes herved oplysninger om navnet og positionen. Når det er udført vælges filen, der er specificeret i programkaldet, til CURRENT INPUT FILE.

Eksempel

i komfile

Filen, der er beskrevet ved katalogindgangen KOMFILE, vælges til CURRENT INPUT FILE.

FEJLUDSKRIFTER

*** I CALL

Der optræder en venstreside i programkaldet.

*** I PARAM

Der er parameterfejl i programkaldet.

*** I <filnavn> <årsag>

Filen kan ikke tilknyttes af den anførte grund.

21.11 O

21.11

Med O-programmet vælges CURRENT OUTPUT FILE.

PROGRAMKALD

o <filnavn>

FUNKTION

En END OF MEDIUM karakter skrives på CURRENT OUTPUT FILE og brugen afsluttes. Når det er gjort vælges filen, der er specificeret i programkaldet til CURRENT OUTPUT FILE. Hvis der ikke findes en fil med det anførte navn, oprettes et område med dette navn, og området vælges til CURRENT OUTPUT FILE.

FEJL

Ved fejl vælges PRIMÆR OUTPUT FILE TIL CURRENT OUTPUT FILE.

Eksempel

o uddata

Filen UDDATA vælges til CURRENT OUTPUT FILE.

FEJLUDSKRIFTER

*** O CALL

Der optræder en venstreside i programkaldet.

*** O PARAM <parameter>

Den anførte parameter må ikke forekomme.

*** O <filnavn> <årsag>

Den specificerede fil kunne ikke tilknyttes af den angivne grund.

21.12 CLEAR

21.12

Med CLEAR-programmet fjernes specificerede katalogindgange.

PROGRAMKALD

$$\text{clear } \langle \text{scope specifikaion} \rangle \left\{ \langle \text{sp} \rangle \langle \text{navn} \rangle \right\}_1^N$$

$$\langle \text{scope specifikaion} \rangle ::= \langle \text{scopeangivelse} \rangle \left\{ . \langle \text{kitnavn} \rangle \right\}_0^1$$

$$\langle \text{scopeangivelse} \rangle ::= \left\{ \begin{array}{l} \text{temp} \\ \text{login} \\ \text{user} \\ \text{project} \end{array} \right\}$$

FUNKTION

De specificerede katalogindgange i den anførte SCOPE-pulje fjernes.

Programmet ignorerer fejlramte parametre, men afbrydes dog efter fejl, der forårsager en af de 4 første fejludskrifter fra nedenstående liste.

Eksempel

```
clear login data1 data2
clear user data1 data2
```

FEJLUDESKRIFTER

*** CLEAR CALL

Venstreside optræder i programkaldet.

*** CLEAR <scope spec.> ILLEGAL SCOPE

Ulovlig scopeangivelse.

*** CLEAR <scope spec.> BS DEVICE UNKNOWN

Det specificerede kit er ikke monteret.

*** CLEAR PARAM <parameter>

Ulovlig parameter i programkaldet. Den resterende del af parameterlisten ignoreres.

*** CLEAR <scope spec.> <navn> UNKNOWN

Den specificerede katalogindgang findes ikke.

*** CLEAR <scope spec.> <navn> ENTRY IN USE

Katalogindgangen kunne ikke fjernes, da den blev benyttet af en anden.

SCOPE-begrebet er nært tilknyttet projektopdelingen af brugerne i BRUGERKATALOGET, idet en katalogindgang er tilknyttet et projekt eller en bruger i et projekt.

SCOPE PROJECT

Til hvert projekt findes en pulje katalogindgange, der er tilknyttet projektet. Katalogindgange i disse puljer karakteriseres med SCOPE-betegnelsen PROJECT.

SCOPE USER

Der er ligeledes tilknyttet en pulje katalogindgange til hver bruger i et projekt. Disse katalogindgange er karakteriseret med SCOPE-betegnelsen USER.

SCOPE LOGIN

Til hver terminalbruger er der også tilknyttet en pulje katalogindgange med SCOPE-betegnelsen LOGIN. Denne pulje slettes, når terminalen forlades (LOGOUT).

SCOPE TEMP

Under afviklingen af et job er der tilknyttet en pulje katalogindgange til jobbet. SCOPE-betegnelsen er TEMP, og puljen slettes ved jobafslutningen.

BESKYTTELSE

I tilknytningsforholdet er der indlagt en beskyttelsesmekanisme for at forhindre uønsket adgang til et område. Katalogindgange, der er tilknyttet et fremmed projekt eller en anden bruger, er usynlige, hvilket indebærer, at såvel læsning som skrivning er umulig.

Med SCOPE-programmet overføres katalogindgange til en specificeret SCOPE-pulje.

PROGRAMKALD

scope <scope specifikation> $\left\{ \begin{matrix} \text{<sp> <navn>} \\ \end{matrix} \right\}_1^N$

$$\langle \text{scope specififikation} \rangle ::= \langle \text{scope angivelse} \rangle \left\{ . \langle \text{kitnavn} \rangle \right\}_0^1$$

$$\langle \text{scope angivelse} \rangle ::= \left\{ \begin{array}{l} \text{temp} \\ \text{login} \\ \text{user} \\ \text{project} \end{array} \right\}$$

FUNKTION

De angivne katalogindgange overføres til den specificerede SCOPE-pulje, og forinden fjernes de eksisterende katalogindgange med de specificerede navne i puljen, så en navnekonflikt forhindres.

En katalogindgang, der ikke beskriver et område, kan gøres permanent på et privat kit ved at anføre kitnavnet efter scopeangivelsen.

FEJL

Fejlramte parametre ignoreres, og programmet afbrydes kun efter de fire første fejludskrifter i nedenstående liste.

Eksempel

```
scope user rdata
scope login rdata1 rdata2 rdata3
scope user rdata1 rdata2 rdata3
```

FEJLUDESKRIFTER

*** SCOPE CALL

Venstreside optræder i programkaldet.

*** SCOPE <scope spec.> ILLEGAL SCOPE

Illegal scopeangivelse:

*** SCOPE <scope spec.> BS DEVICE UNKNOWN

Det specificerede kit er ikke monteret.

*** SCOPE PARAM <parameter>

Parameterfejl i programkaldet. Den resterende del af parameterlisten ignoreres.

*** SCOPE <scope spec.> <navn> UNKNOWN

Der findes ingen katalogindgang med det anførte navn.

*** SCOPE <scope spec.> <navn> PROTECTED

Den anførte katalogindgang er beskyttet og kan derfor ikke ændres.

*** SCOPE <scope sepc.> <navn> ENTRY IN USE

Den anførte katalogindgang benyttes af en anden og kan derfor ikke ændres.

*** SCOPE <scope spec.> <navn> NO RESOURCES

Der er ikke tilstrækkelige ressourcer til rådighed for at kunne gennemføre SCOPE-ændringen.

*** SCOPE <scope spec.> <navn> CHANGE BS DEVICE IMPOSSIBLE

Den anførte katalogindgang kan ikke gøres permanent på det specificerede kit, da den beskriver et område på et andet kit.

En discstatistik bør udskrives ved daglig systemnedlukning, dog mindst en gang om ugen.

Bemærk: alle oplysninger er nulstillet efter autoloade.

En discstatistik kan udskrives ved at oprette en all-proces under operativsystem "s" og derefter følge følgende syntax:

syntax: discstat disc.<navn>.<devno>

hvor <navn> er navnet på discen, og <devno> er det logiske nummer på den fysiske disc, hvorfra statistikken ønskes udskrevet.

Eksempel

En discstatistik ønskes udskrevet med oplysninger om deviceno 26. Statistikken er identificeret med navnet statdisc. Den understregede tekst tastes af operatøren.

esc s

call 26 statdisc ; navngiv device 26 (systemdisc)
ready

esc s

all x run ; lav en all proces
to x
o lp ; output til linieprinter
head 1 ; lav et sideskift på lineprinteren

discstat disc.statdisc.26 ; udskriv discstatistik

o c ; vend current input/output

finis ; fjern processen

En fpa-statistik bør udskrives ved daglig systemnedlukning, dog mindst én gang om ugen.

Bemærk: alle oplysninger er nulstillet efter autoloade.

En fpa-statistik kan udskrives ved at oprette en all proces under operativsystem 's' og derefter følge følgende syntax:

Syntax: fpastat <proc.spec><function>
 <proc.sepc> ::= core8000.<name(fpa-mainproc)>
 <function> ::= $\left\{ \begin{array}{l} \text{test.yes} \\ \text{test.no} \end{array} \right\}$

Parameteren test.yes bevirker at der udover statistiske oplysninger bliver udskrevet testoutput fra FPA-processen.

FPA-mainproces bliver navngivet som følger:

main36001	$\left\{ \begin{array}{l} \text{Første FPA} \\ \text{Anden FPA} \\ \text{etc.} \end{array} \right\}$	RC3600
main36002		
main35001	$\left\{ \begin{array}{l} \text{Første FPA} \\ \text{Anden FPA} \\ \text{etc.} \end{array} \right\}$	RC3500
main35002		

Eksempel

Den understregede tekst skrives af operatøren.

esc s
ALL x run
 ready
 to x

fpastat core8000.main3600 $\left\{ \begin{array}{l} 1 \\ 2 \end{array} \right\}$ test. $\left\{ \begin{array}{l} \text{yes} \\ \text{no} \end{array} \right\}$

Når RC8000 har været nede, foretages automatisk en dump af lagerets indhold til autoloaddelen på systemdiscen. Denne del fylder 210 segmenter svarende til 15360 halvord. Ved hver autoloadd gemmes denne dump automatisk og bliver først overskrevet ved næste autoloadd.

Dette dump kan flyttes til en fil, således at det er muligt at kontrollere indholdet. Flytningen sker ved at oprette en all process under operativsystem "s" og gøre følgende, når processen er oprettet.

```

esc s
ALL x run                ; lav en all process
    to x
Dump8000 = set 210        ; afsæt areal på 210 segmenter
scope user dump8000      ; gør arealet permanent

```

Eksempel

```

eks. 1. { dump8000 = movedump device.disc }
eks. 2. { dump8000 = movedump device.27  }

```

1. Incrementaldumpet fra discen med navn disc bliver kopieret til filen med navn dump8000.
2. Incrementaldumpet fra discens autoloaddel (device 27) bliver kopieret til filen med navn dump8000.

Det er vigtigt at få savet dumpet op på et magnetbånd og sende magnetbåndet, samt en forklarende udfyldt fejlmelding til REGNECENTRALEN.

arbejdslager 4
arealproces 19, 20, 21
asynkron 1, 3
autoload 8, 10, 11, 13, 14, 15, 21, 76, 78
autoload-del 13, 15, 20, 24, 78

back-up 52
baggrundslager 10, 14, 16, 21, 25, 32, 34, 38
base 66, 68
batchjob 37
bikatalog 23, 31, 66
bit 1, 4, 7
bitord 5
BOSS 27, 31, 32, 34, 37, 51
bruger katalog 29, 34, 38
bussen 7
byte 1

CLEAR 71
coroutiner 32
CPU 2, 5, 6

devicehost 12
devicenummer 17, 18
device-controller 9
disc 1, 2, 8, 20, 21, 24, 66
disccopy 63
disc-controller 2, 3, 7
diskette 58
dobbeltord 1, 7
dokument 1, 66
driverproc 16

entry 1, 23, 66
entrynavn 66

ferritkærne 1
ferritkærnelager 1, 4
fil 1, 10, 23, 54, 66
FP 25, 41
FPA 9
FPU 6
front-end 2, 9, 10, 11, 12, 13, 17, 18

halvleder 4
halvord 1, 23, 25, 78
host 12
hovedkatalog 21, 23, 24, 31

I 70
internal 1, 19
intern-proces 29

job 25, 29, 41
jobhost 12
jobkommando 46

kanalprogram 13, 17, 19
katalog 67, 68
katalogbehandling 16
katalogbeskrivelse 23
katalogindgang 21, 23, 26, 52, 54, 57, 67, 69, 71
katalognøgle 68
kitlabel 24
koncentrator 9
konsolbuffer 29
kædetabel 23, 30

lager 2, 15, 16, 25, 29
lagerbeskyttelse 29
link 12
load 56
login 21
lookup 67

maskininstruktion 4

micro 1

microsekund 4, 5, 6

milli 1

millisekund 20

monitoren 12, 13, 14, 16, 18, 19, 20, 21, 22, 23, 24, 28, 29, 68

namekey 23

nano 1

nanosec 3

NCP 12

O 70

onlinejob 37

operativsystem 1, 14, 16, 20, 25, 28, 29, 31, 37, 38

ord 1, 4

paritet 1

permanent 21, 23, 25, 29, 36, 78

permkey 21

pladelager 27

pladelagerkanalen 8

PRIMO 48

primærlager 4, 14

proces 1, 12, 14, 16, 19, 25, 30, 32, 48, 51

procesnavn 20

procfunc 16

program 1, 4, 25

SAVE 52, 56

scope 73

segment 23, 24, 31, 36, 68, 78

semafor 32

semiconductorlager 4

set 69

slice 23, 24, 68

slice størrelse 24

slicetabel 16

SOS 37, 47

spooling 31, 32

subkatalog 23

swapping 31, 38

synchron 1

systembus 2, 3, 6, 8, 9, 10, 13

TEM 45

temporær 21, 69

LÆSERBEMÆRKNINGER

Titel: RC8000 betjeningsvejledning for
operatører.

RCSL Nr.: 31-D691

A/S Regnecentralen af 1979 bestræber sig på at forbedre kvalitet og brugbarhed af sine publikationer. For at opnå dette ønskes læserens kritiske vurdering af denne publikation.

Kommenter venligst manualens fuldstændighed, nøjagtighed, disposition, anvendelighed og læsbarhed:

Angiv fundne fejl (reference til sidenummer):

Hvordan kan manualen forbedres:

Andre kommentarer:

Navn: _____ Stilling: _____

Firma: _____

Adresse: _____

Dato: _____

På forhånd tak!

..... Fold her

..... Riv ikke - Fold her og hæft

Frankeres
som
brev

 **REGNECENTRALEN**
af 1979

Informationsafdelingen
Lautrupbjerg 1
2750 Ballerup