Interne notater
STATISTISK SENTRALBYRÅ

90/23

12. desember 1991

HÅNDBOK I BRUK AV TAB

Av
Kristian Lønø

REVIDERT UTGAVE

FORORD

Oslo 12/12-91

TAB har blitt brukt til tabellproduksjon i SSB siden 1985. Det brukes i dag av de fleste seksjoner i fagavdelingene i Oslo, og blir stadig mer brukt også på Kongsvinger. Tab er et strategisk program på stormaskinsiden. Programmet er maskineffektivt og det er relativt enkelt å bruke. Det er derfor behov for en håndbok i TAB-programmering.

Dette er en revidert utgave av INO 90/23. De feil som er funnet, er rettet opp. Videre er oppgavedelen som var basis for TAB-kursene fjernet. Flytdiagrammene er forbedret og antall feilmeldinger utvidet. Dessuten har jeg laget et stikkordsregister. Dette håper jeg gjør boken lettere å bruke.

Følgende nye sider er lagt til:

Nytt pkt. 11.2 Instruksjonene i forhold til inn- og utfiler
" 11.3 TAB-programmets utførelse
Vedlegg VII: Tegnenes sorteringsrekkefølge
Vedlegg VIII: IBM's overpunch for lagring av negative tall
Vedlegg IX: Program med automatisk beregnede restgrupper

Boken er ment å være et hjelpemiddel for medarbeidere som bruker eller skal bruke TAB i arbeidet sitt.

Jeg har lagt vekt på å bruke mange eksempler, disse er tatt fra program som er laget i Byrået.

Jeg er fortsatt åpen for kommentarer til boken. Finner du feil eller mangler håper jeg du kontakter meg, slik at vi kan få endret disse til en eventuell neste utgave.

Kristian Lønø
Gruppe for EDB,
Personsstatistikk (GEP),
Oslo

INNHOLDSFORTEGNELSE

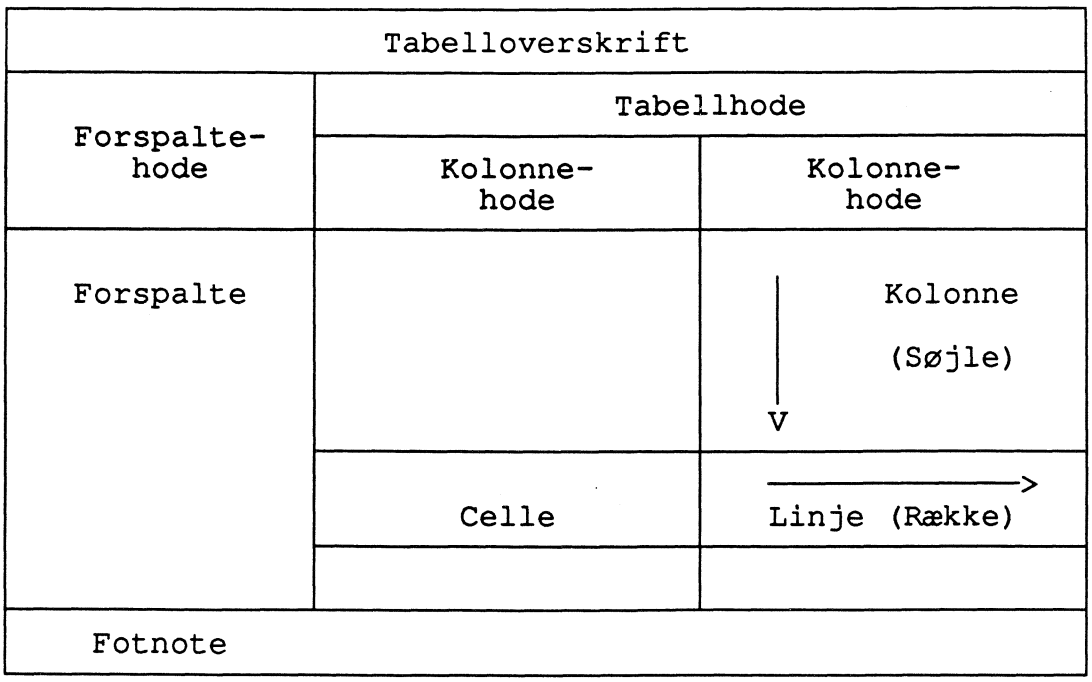
1. INNLEDNING

TAB er et standardprogram for å lage tabeller på stormaskin. Programmet er utviklet av Danmarks Statistik (DS). DS sørger også for vedlikehold og oppgraderinger. Vi vil få nye versjoner fra DS når de foreligger.

Programmet egner seg godt til å lage trykkeklare tabeller. TAB er spesielt slagkraftig når det lager relativt små tabeller (inntil ca. 10 000 records) fra store datafiler (mer enn ca. 100 000 records). Programmet kan også brukes til å aggregere store datafiler til et høyere nivå som gir færre records. Her vil eksekveringstiden øke med antall records programmet skriver ut.

Noen av eksemplene i denne håndboka ligger lagret på Comparex-maskinen på Kongsvinger, de ligger lagret som membere på det partisjonerte datasettet (PDS'et) TAB.DIV. Her ligger også en del andre nyttige eksempler og hint. Fra TAB.DIV kan du kopiere over membere med eksempler til egen bruker og kjøre dem selv. Du får da testet hvordan instruksjonene virker, og eksemplene er fine som utgangspunkt for å lage egne program.

2. HVA EN TABELL BESTÅR AV



Eksempel:

Tabell 1. Folkemengde etter aldersgrupper. Fylke. 1985

Fylke	Innb. i alt	Aldersfordeling 0-15 år	16-66 år	67-79 år	80 år og over
Hele landet	4145845	905040	2678215	424012	138578
01 Østfold	235039	49820	151568	26054	7597
02 Akershus	386278	87023	263067	27621	8027
03 Oslo	447351	67554	299804	58759	21234
:					
:					
:					
20 Finnmark	76650	18664	50683	5860	1443

Tallene gjelder 1. januar 1985

Tabelloverskrift:	1. linje (Tabell 1 Folkemengde...)
Tabellhode:	3. linje (Aldersfordeling)
Kolonnehode:	Overskrift til kolonnene (F.eks. Innb. i alt)
Forspaltehode:	Overskrift til forspalten (Fylke)
Forspalte:	Ledetekst til linjene (Hele landet, 03 Oslo...)
Kolonne:	Tall (F.eks. 424012, 26054, 27621, 58759, 5860)
Linje:	Tall (F.eks. 76650, 18664, 50683, 5860, 1443)
Celle:	Et tall der kolonne og linje møtes (F.eks. 5860)
Fotnote:	Siste linje (Tallene gjelder...)

3. TAB

Tab er altså et dansk standardprogram for tabellproduksjon. Byrået har fått tillatelse av DS til å bruke det.

Tab bygger på filebehandlingsprogrammet PLUK (også utviklet av DS) som sammen med Tab er blant de mest brukte standardprogrammene der. Syntaksen i Pluk og Tab er helt lik, og det er mange funksjoner som er identiske for Pluk og Tab. Danmarks Statistik har derfor valgt å lage en komplett manual for Pluk, mens Tab-manualen bare inneholder de instruksjoner som er spesielle for Tab. De andre instruksjonene er bare beskrevet i Pluk-manualen. Derfor har jeg prøvd å lage en håndbok i bruk av Tab. Det er ikke ment å være noen komplett manual, men de fleste instruksjoner og parametre er med. Dessuten har jeg lagt vekt på å bruke eksempler fra programmer som er lagd i Byrået.

Fordi Tab er et dansk program har instruksjonene og de fleste parametrene danske navn (noen har engelske). Disse har som oftest norske ord som er omtrent like. To unntak som vi bør ha klart for oss er at det vi kaller linjer, kaller danskene rækker, og det vi kaller kolonner, heter søjler på dansk. Derfor dukker disse to danske ordene opp i teksten av og til.

Et Tab-program består av:

1. Definisjoner

(2. Seleksjon)

(3. Forbehandling)

4. Opptelling

5. Beskrivelse av forspalten

(6. Beregninger)

7. Beskrivelse av utskrift

De delene som står i parentes er frivillige, de andre må være med. Denne rekkefølgen må følges når vi programmerer, og derfor har jeg valgt å gjennomgå instruksjonene i denne rekkefølgen. Det er noen instruksjoner som kan brukes flere steder i programmet (IF - ELSE-tester kan f. eks. brukes både i forbehandling, beregninger og utskrift). Disse blir bare forklart første sted i programmet du kan bruke dem.

Vi kan bare ha 1 input-fil til et Tab-program. Denne filen må være sekvensiell, og de felt som brukes skal være innenfor de første 4096 bytes av recorden. Recordformatet på input-filen kan være fast (F,FB), variabelt (V,VB) eller undefined (U).

Input-filen til Tab er 1 logisk fil. Denne logiske filen kan bestå av flere fysiske filer, poenget er at vi kan hekte sammen (konkatenere) flere filer. På den måten vil vi kunne lese flere fysiske filer som om de var én fil. Disse filene må selvfølgelig ha lik filbeskrivelse. Sammenhektingen gjør vi i JCL (eksempel; se JCL til eksemplet til FIND).

TAB består av 2 hovedprogrammer; Tab1 og Tab2. Tab1 er delt opp i 3 underprogrammer; TAB1, TAB1A og TAB1B, mens Tab2 bare består av TAB2. For Tab1 blir det forvirrende med at både hovedprogrammet og et av underprogrammene heter det samme. For å gjøre forvirringen litt mindre bruker jeg store bokstaver når jeg refererer til underprogrammet TAB1, og små bokstaver når jeg refererer til hovedprogrammet Tab1 (dvs. når jeg refererer til TAB1, TAB1A og TAB1B samlet).

Forskjellen mellom Tab1 og Tab2 er først og fremst hvordan forspalten bygges opp.

1. Tab1 - Bygger opp forspalten etter 1 eller flere forspaltekriterier. Disse kriteriene er som oftest felt fra input-filen. Hver verdi disse feltene har, vil resultere i en linje på utskriften. Hvert forspaltekriterium vi definerer i programmet vårt vil gi ett bruddnivå.

   Eksempel: Forspaltekriterier; land og varenummer

```
 Norden              1000   <=== Totalsum

 Sverige              300   <=== 1. bruddnivå
  0101010              50   <=== Laveste nivå
  0101090             250
 Danmark              700
  0101010             550
  0101090             150
```

   Som vi ser av eksemplet over vil alle verdiene for underliggende nivåer gå igjen for hver verdi av det overliggende nivå.

2. Tab2 - Vi må definere hver eneste linje som skal skrives ut til tabellen vår. Det betyr at vi ikke har noen bruddnivåer slik som i Tab1, men vi kan lage forspalten omtrent som vi vil. Det gir flere muligheter, men det betyr også at det blir mer programmering.

   Eksempel:

```
  BEGGE KJØNN      57
  Menn             24
  Kvinner          33

  HELE LANDET      57
  Akershus         40
  Oslo             17
```

Denne tabellen består egentlig av 2 tabeller; 1 for kjønn og 1 for fylker. Her har vi 1 forspalte som ikke kan defineres ved hjelp av forspaltekriterier, derfor må den lages i Tab2.

JCL for kjøring av TAB1A-program (uten forspaltetekstfil):

```
//0303KRL  JOB 8019,'TAB1A uten TXT-fil',TIME=1,
//           MSGLEVEL=(0,0),MSGCLASS=X,CLASS=D,NOTIFY=0303KRL
/*ROUTE    PRINT RMT6
//UTEN     EXEC TAB1A,TIME=1
//INPUT    DD DSN=TAB.DIV(DATA),DISP=SHR
//TABELL   DD SYSOUT=*,COPIES=1
//SYSIN    DD *
:
Her skriver du inn programmet ditt
:
```

JCL for kjøring av TAB1B-program (med forspaltetekstfil):

```
//0303KRL  JOB 8019,'TAB1B med TXT-fil',
//           MSGLEVEL=(0,0),MSGCLASS=X,CLASS=A,NOTIFY=0303KRL
/*ROUTE    PRINT RMT6
//MED      EXEC TAB1B
//INPUT    DD DSN=TAB.DIV(DATAFIL1),DISP=SHR
//TABELL   DD DSN=TK414.S8019.KRL.TABELL8,DISP=(NEW,CATLG),
//            DCB=(RECFM=FBA,LRECL=133,BLKSIZE=23142),
//            SPACE=(23142,(20,20),RLSE),UNIT=SSB
//FYLKTXT  DD DSN=TAB.DIV(FYLKTXT),DISP=SHR
//KOMMTXT  DD DSN=TAB.DIV(KOMMTXT),DISP=SHR
//SYSIN    DD *
```

JCL for kjøring av TAB2-program (med forspaltetekstfil), her ligger programmet på source-biblioteket til Tab. Dette skal brukes til Tab-program som brukes i produksjonskjøringer:

```
//0303KRLZ JOB 8019,'TAB2 med TXT-fil',
//           MSGCLASS=X,CLASS=A,NOTIFY=0303KRL
/*ROUTE    PRINT RMT6
//MED2     EXEC TAB2,PARM='TXT=1986'
//INPUT    DD DSN=TAB.DIV(DATAFIL1),DISP=SHR
//TABELL   DD DSN=TK414.S8019.KRL.TABELL7,DISP=OLD
//FORSPTXT DD DSN=TAB.DIV(FORSPTXT),DISP=SHR
//SYSIN    DD DSN=SSB2.SOURCE.TAB(TAB61032),DISP=SHR
```

JCL for kjøring av TAB1-program (med SELKRIT. Bare de linjer som har kodene FYL og BOK eller har blanke i pos. 73-75 blir tatt med i programmet når det eksekveres):

```
//0303KRL  JOB 8019,'TAB1 med SELKRIT',
//           MSGLEVEL=(0,0),MSGCLASS=X,CLASS=A,NOTIFY=0303KRL
/*ROUTE    PRINT RMT6
//SEL      EXEC TAB1,TIME=1
//INPUT    DD DSN=PP414.S8019.TABKURS.DATAFIL2,DISP=SHR
//TABELL   DD SYSOUT=*,COPIES=1
//SELKRIT  DD *
FYL,BOK
//SYSIN    DD *
```

5. KORTFORMAT (PLASSERING AV INSTRUKSJONER OSV.)

Plassering av instruksjoner osv. er posisjonsbestemt.
Posisjonene er disse:

Kolonne 01-08: Label/Blanke

Kolonne 10-14: Instruksjoner

Kolonne 16-71: Parametre

Kolonne 73-75: SELKRIT (seleksjonskriterium for kompileringen)
I forbindelse med kompileringen av programmet kan vi velge hvilke programlinjer vi har skrevet som skal være med i kompileringen og hvilke som ikke skal være med. Det angir vi med å skrive en 3-bokstavskode i pos. 73-75. For å fortelle hvilke koder som skal være med må vi ha med et JCL-kort (Se JCL). Bare de linjene i programmet vårt som har koden vi angir i JCL-kortet og de linjene som ikke har noen kode i pos. 73-75, blir kompilert.

Kommentar: * i kolonne 1 (eller etter parametre).

Tips: Kommentarer kan du gjerne bruke for å "fjerne" instruksjoner som du vil bruke senere. Istedenfor å slette linja med instruksjonen, plasserer du en * i første posisjon, og dermed er instruksjonen omgjort til en kommentar. Når du trenger instruksjonen igjen, skriver du bare en blank istedenfor *. Dette kalles å "kommentere vekk" instruksjoner.

Eksempel:

```
1         10    16                                                  73
|         |     |                                                   |
|         |     |                                                   |
|         |     |                                                   |
V         V     V                                                   V

AARGANG   FIELD (49,2,X) * ÅRSTALL
          :
          :
          RÆKKE R3:R14,(N3 = 611:612:613:614:621:622:623:624:     FFF
                             625:626:627:629)                     FFF
          :
          :
          FORSP TXTFIL=FORSPTXT,TXT=(5,59),ASA=65,RAKNR=2,
                PRINT=1
```

6. LOGISKE UTTRYKK

Logiske uttrykk kan brukes i disse instruksjonene:

SELECT
HOP
IF
CASE
LOOP
RÆKKE
SØJLE
SGRP

Et logisk uttrykk er en setning som inneholder en logisk relasjon. Logiske relasjoner er f. eks. =, >, < osv.

Her følger en liste over de relasjonene du kan bruke:

=, EQ	(lik, EQual)
^=,NE	(ulik, Not Equal)
<, LT	(mindre enn, Less Than)
NL	(ikke mindre enn, Not Less than)
>, GT	(større enn, Greater Than)
>=	(større enn eller lik)
<=	(mindre enn eller lik)
==>	(implikasjon)
<==>	(biimplikasjon)

De logiske uttrykkene brukes som betingelser til instruksjonene over, og de skal alltid skrives i parentes. I parentesen skal det være 1 blank på hver side av relasjonen (se eksemplene under).

Enkle logiske uttrykk:

1. FELTNAVN1 RELASJON FELTNAVN2

2. FELTNAVN RELASJON KONSTANT

3. FELTNAVN RELASJON SPACES/ZEROES/LOW-VALUES/HIGH-VALUES

Eksempel:

```
SELECT   INPUT (GMLLAND = NYTTLAND)

         SØJLE S1,(KOMMNR < 0412),PERSONER

         IF     (VERDI = SPACES)
```

Merk:

For relasjonene lik (=) og ulik (NE) kan flere verdier angis og de kan skrives som intervall:

```
SELECT INPUT (KOMMNR = 1727,1831-1839,1870)
```

2 eller flere uttrykk kan settes sammen med AND eller OR. Det skal være 1 blank på hver side av AND og OR (se eksemplene under). For å skille uttrykkene lønner det seg å bruke parenteser (max 5 nivåer).

```
SELECT   INPUT (RECTYPE = 1 OR CCCN = 3601001 AND MENGDE > 400)
SELECT   INPUT ((RECTYPE = 1 OR CCCN = 3601001) AND MENGDE > 400)
SELECT   INPUT (RECTYPE = 1 OR (CCCN = 3601001 AND MENGDE > 400))
```

De 2 første betingelsene ovenfor er like, mens den tredje gir et annet resulat. Som vi ser blir uttrykket mer oversiktlig når vi bruker parenteser.

Tips: Bruk parenteser, det gjør uttrykkene lettere å lese og minsker sjansen for at du gjør feil (spesielt ved bruk av OR).

7. HVORDAN LAGER JEG TAB-PROGRAMMER

Slik bør du gå fram når du lager et TAB-program:

0. Hvis dataene dine ligger på tape eller kassett, sørg for å få lagd en testfil på ca. 100-1000 records.
1. Kopier et tidligere TAB-program som ligner det du skal lage.
2. Gjør alle endringer i filbeskrivelsen (FIELD).
3. Endre SØJLER og Forspalten (FORSP).
4. Lag MOVE-instruksjoner med plassering etter raskt øyemål.
5. Legg inn TEST-instruksjon som stopper etter ca. 100 records.
6. Kjør programmet (med TEST-instruksjon!).
7. Se hvordan resultatet blir.
8. Lag overskriften (HDR) etter raskt øyemål.
9. Gjør nødvendige tilpasninger i MOVE-instruksjonen.
10. Kjør programmet på nytt (med TEST-instruksjonen).
11. Se hvordan resultatet blir. Ta en rask titt på tallene også (hvis de kan fortelle deg noe på 100 records).
12. Ordne overskriften (HDR) og MOVE-instruksjonene
13. Kjør programmet på nytt (med TEST-instruksjonen).
14. Se om tabellen ser ut som den skal.
15. Fjern TEST-instruksjonen.
16. Kjør programmet på hele filen.
17. Se hvordan resultatet blir. Sjekk tallene nøye.
18. Gå igjennom programmet og se om du har programmert riktig.
19. Skriv tabellen ut.

Når du etter hvert begynner å programmere mer kompliserte tabeller bør du begynne med den enkle delen av tabellen, og så etter hvert legge inn en etter en av de mer kompliserte delene. Alt dette gjør du selvfølgelig bare på en testfil (bruk TEST-instruksjonen).

8. TESTKJØRING AV TAB-PROGRAMMER

Testing av programmer består av 2 deler: Syntakssjekk og logisk test.

Syntakssjekk

Sjekking av syntaksen gjøres av TAB-programmet før det starter eksekveringen. Hvis det er feil i syntaksen, vil programmet bli stoppet, du får returkode 16, og du vil få ut feilmeldinger. Disse vil normalt stå under den linja i programmet feilen var i, og det vil være anvist med *** hvor på linja feilen er. Pr. idag har vi ikke noen liste over feilmeldinger med nærmere forklaringer, men de fleste feilmeldingene vi får ut i kjørerapporten skulle være greie å tolke. Når det ikke er flere syntaksfeil igjen, vil selve programmet bli utført.

Logisk test.

Den logiske testen går ut på å teste om programmet gjør det vi vil at det skal gjøre, f.eks at beregninger blir riktige og at riktige felt blir aggregert. Det er meget viktig å sjekke at logikken i programmet er riktig. Hvis den ikke er det, kan vi få ut gale tall i tabellen vår. Derfor er det alltid en fordel å kunne kontrollsjekke tallene i tabellene vi lager. Hvis det ikke er mulig å kontrollsjekke tallene, bør du se nøye igjennom programmet ditt for å se om du har gjort alt riktig. Dessuten bør du vurdere riktigheten av tallene så godt det lar seg gjøre.

9. INSTREAM TESTFILER

Er ikke datafilen du skal bruke ferdig når du lager tabellprogrammet kan/må du lage en testfil selv. Denne kan du legge inn som en Instream fil i JCL-oppsettet til programmet. Denne bør selvfølgelig være mest mulig lik datafilen din. Under viser jeg et TAB-program som har innfilen (INPUT) og fylkeskatalogen (FYLKTXT) som Instream filer i JCL:

```
//0414KRL  JOB 8019,'Test med Instream fil',MSGCLASS=X,
//          CLASS=A,NOTIFY=0414KRL
//INSTREAM EXEC TAB1B
//INPUT    DD *
01011111000334501100000010222000
02112222000334600100000005111001
01021333000334700300000001111002
//TABELL   DD SYSOUT=*
//KOMMTXT  DD DSN=TAB.DIV(KOMMTXTS),DISP=SHR
//FYLKTXT  DD *
01 Østfold
02 Akershus
//SYSIN    DD *
         START TYPE=P
FYLKE    FIELD (1,2,X)              * FYLKE
KOMM     FIELD (1,4,X)              * KOMMUNENR
KJØNN    FIELD (5,1,X)              * KJØNN
ANTALL   FIELD (9,6,X)              * ANTALL PERSONER
         SØJLE S1,,ANTALL
         FORSP FYLKE,TXTFIL=FYLKTXT,TXTKEY=(1,2,X),TXT=(1,22),
               PRINT=1,NEWLINE
         FORSP KOMM,TXTFIL=KOMMTXTS,TXTKEY=(1,4,X),TXT=(1,25),PRINT=1
TABELL   FILE  PRINT,OVERFLOW,MAXLIN=46
         HDR   1,1,'FYLKE       ANTALL'
         MOVE  (13,10,Z-),S1
```

Dette eksemplet viser JCL til test-kjøring av et TAB1B-program der input-filen og fylkeskatalog er lagt inn som instream filer, mens kommunekatalogen ligger på et annet datasett.

Ved å bruke Instream testfiler vil du kunne lage programmene før datafilen din er ferdig. Når så datafilen er klar, bytter du ut Instream testfilen med datafilen. JCL-en vil da se slik ut:

```
//0414KRL  JOB 8019,'Kjøring på datafilen',MSGCLASS=X,
//          CLASS=A,NOTIFY=0414KRL
//INSTREAM EXEC TAB1B
//INPUT    DD DSN=PP414.S8019.TABKURS.DATAFIL2,DISP=SHR
//TABELL   DD SYSOUT=*
//KOMMTXT  DD DSN=TAB.DIV(KOMMTXTS),DISP=SHR
//FYLKTXT  DD DSN=TAB.DIV(FYLKTXTS),DISP=SHR
//SYSIN    DD *
```

10. FEILSØKING

Når du kjører et TAB-program vil det alltid være en sjanse for at programmet vil bli avbrutt før det er ferdig. Når dette skjer kaller vi det en ABEND (engelsk: ABnormal ENDing). Når dette oppstår vil du få en returkode (RC) på den første side i kjørerapporten din. Det første du gjør er da å sjekke hvilken returkode programmet har gitt. Så slår du opp i oversikten over feilmeldinger i Vedlegg I i denne boken og ser om du utfra det som står der kan finne ut av hva som har gått galt, og hva du må gjøre for å få programmet til å gå riktig neste gang du prøver. Når TAB-programmet blir avbrutt, vil du normalt få ut en kjørelogg som vanligvis står nederst i kjørerapporten din. Denne kjøreloggen vil også hjelpe deg til å finne feilen. Men hvordan skal du tolke den informasjonen kjøreloggen gir deg? La oss vise et eksempel på et program som vil bli avbrutt med en av de vanligste returkodene, nemlig U0107.

Programmet under vil bli avbrutt på record nr. 2, feltet MENGDE, hvis verdi skal bli telt opp, men ikke er numerisk.

```
//0414KRL  JOB 8019,'Abend eksempel',MSGCLASS=X,CLASS=A,NOTIFY=0414KRL
//INSTREAM EXEC TAB1B
//INPUT    DD *
100012345003232
2       0003244
300364772003347
//TABELL   DD SYSOUT=*
//KOMMTXT  DD DSN=TAB.DIV(KOMMTXTS),DISP=SHR
//FYLKTXT  DD *
01 Østfold
02 Akershus
//SYSIN    DD *
         START TYPE=P
LANDOMR  FIELD (1,1,X)              * VERDENSDEL
MENGDE   FIELD (2,8,X)              * MENGDE I KG
VERDI    FIELD (10,8,X)             * VERDI I KRONER
         SØJLE S1,,MENGDE
         SØJLE S2,,VERDI
         FORSP LANDOMR,PRINT=6
TABELL   FILE  PRINT,OVERFLOW
         HDR   1,1,'VERDENSDEL      TONN   1000 KR'
         MOVE  (11,10,Z-)*2,S1/1000
```

Når dette programmet kjøres, får du returkode (RC) U0107 (forsøk på regning med verdi som ikke er numerisk) og dessuten en dump (utlisting av diverse interne registres verdier) og en kjørelogg med nyttig informasjon. Her vil det stå hvilken instruksjon (dvs. hvilken linje i programmet) kjøringen skar seg på, og det vil stå hvor mange records som er lest fra input-filen. Det som står i dumpen er ikke mulig å lese for den vanlige programmerer, og det er heller ikke nødvendig, så den hopper vi lett over.

Utdrag av kjøreloggen ved abend med returkode (RC) U0107:

```
************************ TAB1B KØRSELSLOG ** 0414KRLD ** TAB1B  *
*                                                               *
*           ************** A B E N D ********************       *
*           *                                          *        *
*           * INTERRUPT 0F62C8 (DATA)                  *        *
*           * STATEMENT 00004                          *        *
*           * INDGANG : MAIN (OPTÆLLING I SØJLER)      *        *
*           *                                          *        *
*           ********************************************        *
*                                                               *
* INPUT                                          DSN=SYSIN    *
*       2 RECORDS LÆST HERAF         2 SELEKTERET               *
*                                                               *
* TABELL                                         DSN=SYSOUT   *
*       INGEN LINIER DANNET                                     *
*                                                               *
*                                                               *
*****************************************************************
```

Av denne loggen leser vi 3 viktige ting:

1) Abend oppstått ved eksekvering av instruksjon 00004 (STATEMENT)
2) Abend oppstått i opptellingsfasen (OPPTÆLLING I SØJLER)
3) Abend oppstått for record nummer 2 (2 RECORDS LÆST)

Ut fra denne informasjonen vil vi finne ut at feilen oppsto under eksekvering av instruksjon nr. 4. Vi finner da ut hvilken instruksjon som er nr. 4. Det viser seg å være denne:

```
4       SØJLE S1,,MENGDE
```

Av dette skjønner vi at feilen har oppstått ved opptelling av feltet MENGDE. Vi vet også hva slags feil som har oppstått (opptelling av ikke numerisk felt). Nå trenger vi å vite i hvilken record feilen oppsto. Dette finner vi også i kjøreloggen. Der står det at feilen oppsto i record nummer 2. Vi kan da gå inn og se på innfilen (hvis den ikke ligger på tape eller kassett), finne record nr. 2 og sjekke hva som står i feltet MENGDE. I dette tilfellet innholdt MENGDE blanke karakterer, altså er feltet ikke numerisk. For å få programmet til å virke må vi enten rette feltets verdi eller legge inn en ekstra test i programmet.

Den ekstra testen blir slik:

```
//0414KRL  JOB 8019,'Test mot Abend',MSGCLASS=X,CLASS=A,NOTIFY=0414KRL
//INSTREAM EXEC TAB1B
//INPUT    DD *
100012345003232
2       0003244
300364772003347
//TABELL   DD SYSOUT=*
//KOMMTXT  DD DSN=TAB.DIV(KOMMTXTS),DISP=SHR
//FYLKTXT  DD *
01 Østfold
02 Akershus
//SYSIN    DD *
         START TYPE=P
LANDOMR  FIELD (1,1,X)             * VERDENSDEL
MENGDE   FIELD (2,8,X)             * MENGDE I KG
VERDI    FIELD (10,8,X)            * VERDI I KRONER
         IF    (MENGDE = SPACES)   * HVIS MENGDE ER BLANK, SETT 0
         SET   MENGDE=0
         ENDIF
         SØJLE S1,,MENGDE
         SØJLE S2,,VERDI
         FORSP LANDOMR,PRINT=6
TABELL   FILE  PRINT,OVERFLOW
         HDR   1,1,'VERDENSDEL       TONN   1000 KR'
         MOVE  (11,10,Z-)*2,S1/1000
```

Det vi gjør i programmet er å teste om feltet MENGDE bare inneholder blanke tegn. Hvis det er tilfelle, setter vi feltet lik 0. Deretter teller vi opp verdien til MENGDE (dvs. 0). Hvis du skal telle opp verdien til flere felt som kan inneholde blanke, kan du lage en test for hvert av feltene.

11. TAB1

Tab1 er delt inn i 3 programmer:

TAB1

TAB1A

TAB1B

TAB1

For å kunne lage tabeller i TAB1, må inputfilen være sortert stigende på forspaltekriteriene. Forspalten bygges opp ut fra 1-14 felt fra inputfilen. Det betyr at vi kan ha max 14 forspaltekriterier i TAB1.

TAB1 er det mest effektive av TAB-programmene, så hvis du vet at inputfilen din er sortert på forspaltekriteriene og du ikke må ha summer øverst, lønner det seg å bruke TAB1.

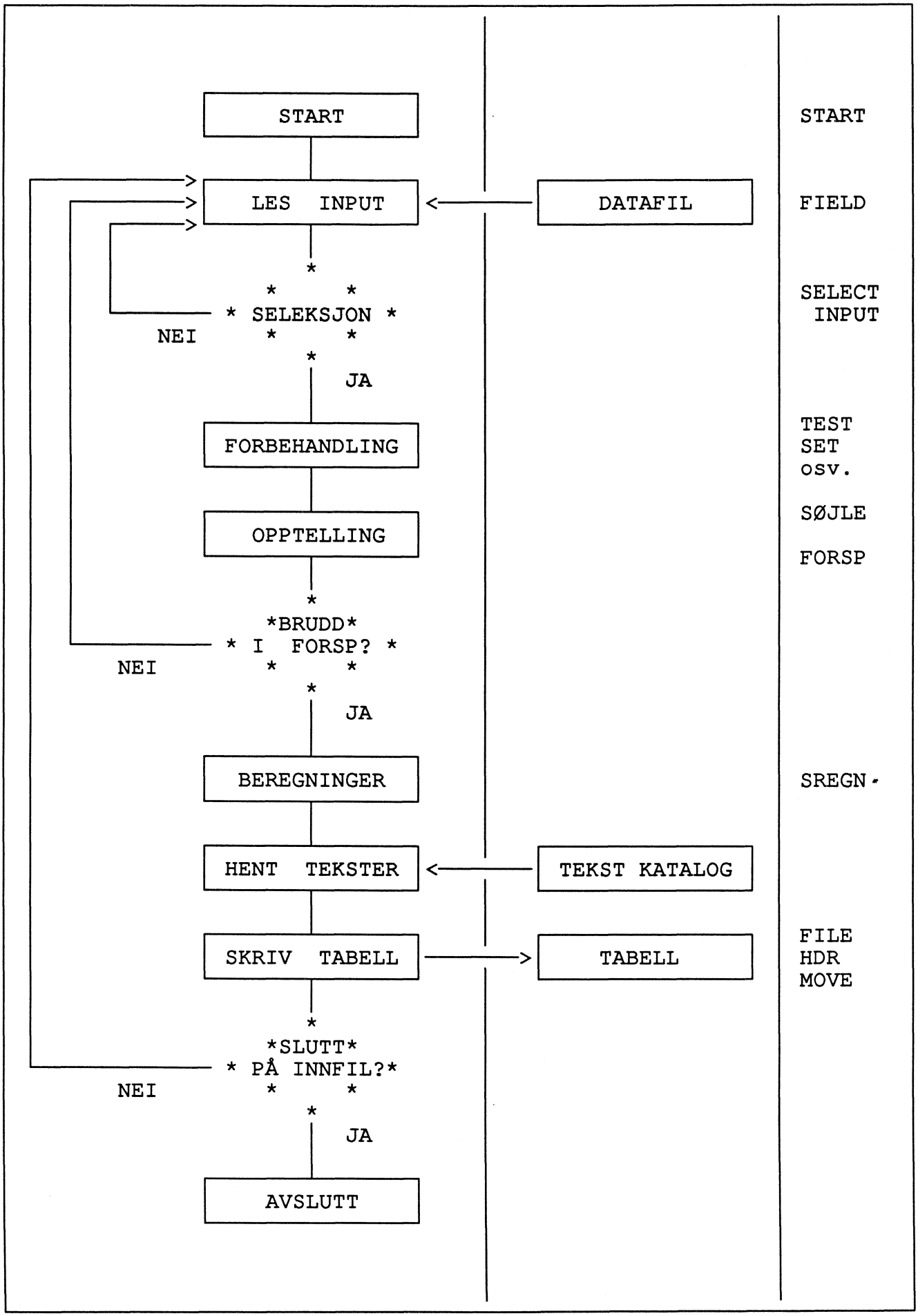
TAB1 egner seg godt til å lage lage aggregerte filer ut fra en basisfil.

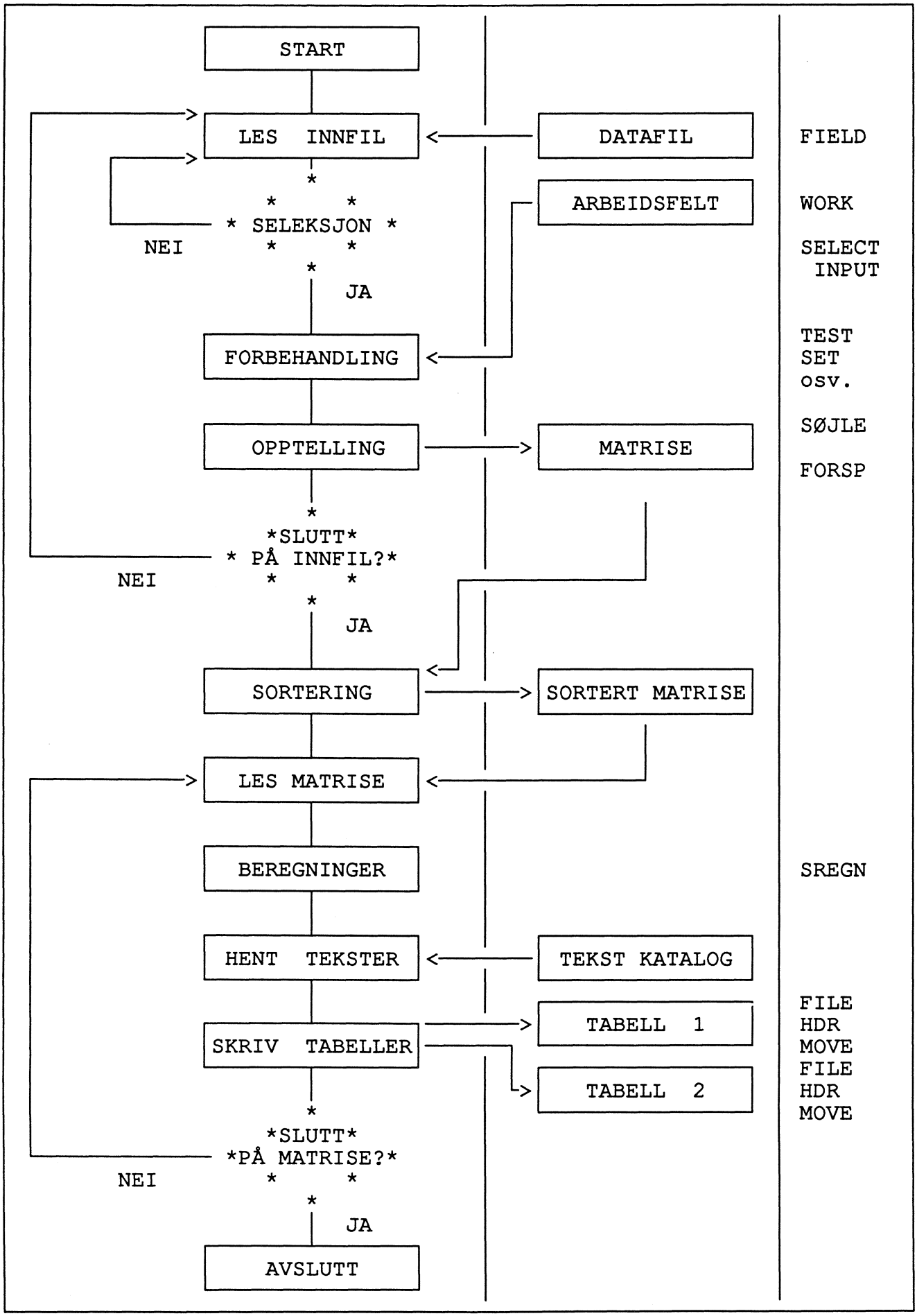
TAB1A

I motsetning til TAB1, krever ikke TAB1A at inputfilen er sortert på forspaltekriteriene. Det betyr at det ikke er nødvendig at forspaltekriteriene er felt i inputrecorden, de kan også beregnes i forbehandlingen, dvs. før opptelling. I TAB1a bygges det opp en matrise (tabell) i storage (og på disk hvis nødvendig). Det betyr at det blir separate opptellings- og utskrivingsfaser i TAB1A. Dette medfører at felt fra inputfilen ikke kan refereres i utskriftsfasen. I TAB1A kan vi dele opp tabellen i 1-9 deltabeller (vha TABEL, men summen av TABEL- og FORSP-kriteriene kan ikke være mer enn 13), hver deltabell vil inneholde elementærlinjer (linjer på laveste nivå i tabellen), sumlinjer (linjer med aggregerte tall) og totallinje. Dessuten kan det lages sumtabeller (etter samme system som sumlinjer) og totaltabell.

TAB1B

Den eneste forskjellen mellom TAB1A og TAB1B er at i TAB1A skrives sumlinjer ut nederst, mens i TAB1B skrives de ut øverst. Dette betyr at TAB1B krever noe mer ressurser, derfor lønner det seg å bruke TAB1A hvis det ikke skal lages sumlinjer, eller hvis det ikke spiller noen rolle om sumlinjene kommer øverst eller nederst i tabellen.


START
LES INPUT
DATAFIL
* SELEKSJON *
NEI
JA
FORBEHANDLING
OPPTELLING
BRUDD
* I FORSP? *
NEI
JA
BEREGNINGER
HENT TEKSTER
TEKST KATALOG
SKRIV TABELL
TABELL
SLUTT
* PÅ INNFIL?*
NEI
JA
AVSLUTT
START
FIELD
SELECT
INPUT
TEST
SET
osv.
SØJLE
FORSP
SREGN-
FILE
HDR
MOVE


START
LES INNFIL
DATAFIL
FIELD
ARBEIDSFELT
WORK
* SELEKSJON *
NEI
SELECT
INPUT
JA
FORBEHANDLING
TEST
SET
osv.
OPPTELLING
MATRISE
SØJLE
FORSP
SLUTT
* PÅ INNFIL?*
NEI
JA
SORTERING
SORTERT MATRISE
LES MATRISE
BEREGNINGER
SREGN
HENT TEKSTER
TEKST KATALOG
FILE
HDR
MOVE
TABELL 1
SKRIV TABELLER
FILE
HDR
MOVE
TABELL 2
SLUTT
PÅ MATRISE?
NEI
JA
AVSLUTT

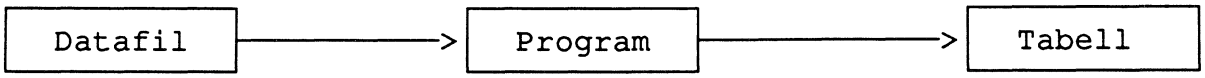
```
//0414KRLD JOB (8019),'TAB1B-eks.',CLASS=A,MSGCLASS=X,NOTIFY=0414KRL
//TAB      EXEC TAB1B
//INPUT    DD   *
101 0102100 0010 1 0003
106 0101090 0015 1 0001
103 0101090 0010 1 0002
105 0105300 0100 2 0032
106 0101010 0020 1 0002
103 0101090 0025 1 0040
//LANDKAT  DD   *
005SVERIGE
101DANMARK
103FINLAND
105ISLAND
110NORGE
//TABELL   DD   SYSOUT=*,COPIES=1
//SYSIN    DD   *
         START TYPE=P
LAND     FIELD (1,3,X)                          * LAND
CCCN     FIELD (5,7,X)                          * 7-SIFRET CCCN
VERDI    FIELD (13,4,X)                         * VERDI
RECTYPE  FIELD (18,1,X)                         * INNF = 1, UTF = 2
MENGDE   FIELD (20,4,X)                         * MENGDE
SELECT   INPUT (RECTYPE = 1 AND LAND = 101-106)
         IF    (LAND = 106)
         SET   LAND=005
         ENDIF
         SØJLE S1,,MENGDE
         SØJLE S2,,VERDI
         TOTAL TXT='NORDEN'
         FORSP LAND,TXTFIL=LANDKAT,TXTKEY=(1,3,X),TXT=(4,12),
               PRINT=1,NEWLINE
         FORSP CCCN,PRINT=3
         SREGN S3=S2*100000/S1
TABELL   FILE  PRINT,OVERFLOW,MAXLIN=46
         HDR   (1-2),(1-72)
INNFØRSEL AV DYR.
                   ANTALL    1 000 KR            PRIS
         MOVE  (15,12,E-)*2,S1, ZZZ ZZZ ZZZ
         MOVE  (42,12,E-),S3,Z ZZZ ZZ9,99
```

Dette programmet gir denne tabellen:

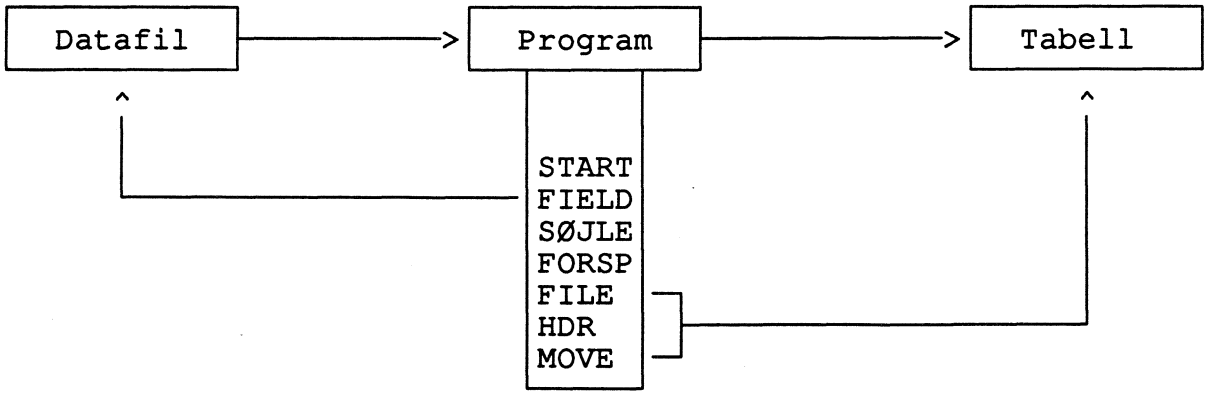
INNFØRSEL AV DYR.

	ANTALL	1 000 KR	PRIS
NORDEN	48	80	1 666,67
SVERIGE	3	35	11 666,67
0101010	2	20	10 000,00
0101090	1	15	15 000,00
DANMARK	3	10	3 333,33
0102100	3	10	3 333,33
FINLAND	42	35	833,33
0101090	42	35	833,33

11.2. INSTRUKSJONENE I FORHOLD TIL INN- OG UTFILER



Datafilen behandles av programmet som lager tabellen.



START må være med for å si fra at nå begynner programmet.

FIELD brukes til å beskrive innfilen. En FIELD-instruksjon for hvert felt vi trenger fra innfilen.

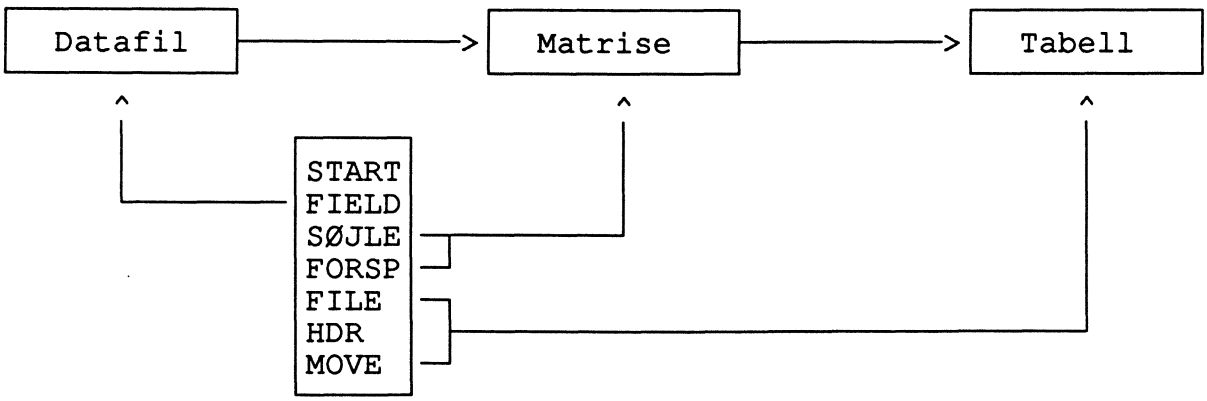
SØJLE bruker vi for å si hva som skal telles opp i kolonner. En SØJLE-instruksjon pr. kolonne

FORSP bruker vi til å si hvilket felt som skal danne forspalte. Det blir en linje for hver verdi av feltet som finnes i innfilen. Bruker vi flere FORSP-instruksjoner får vi flere nivåer i forspalten. Da vil det lages en linje for hver kombinasjon av forspaltekriteriene.

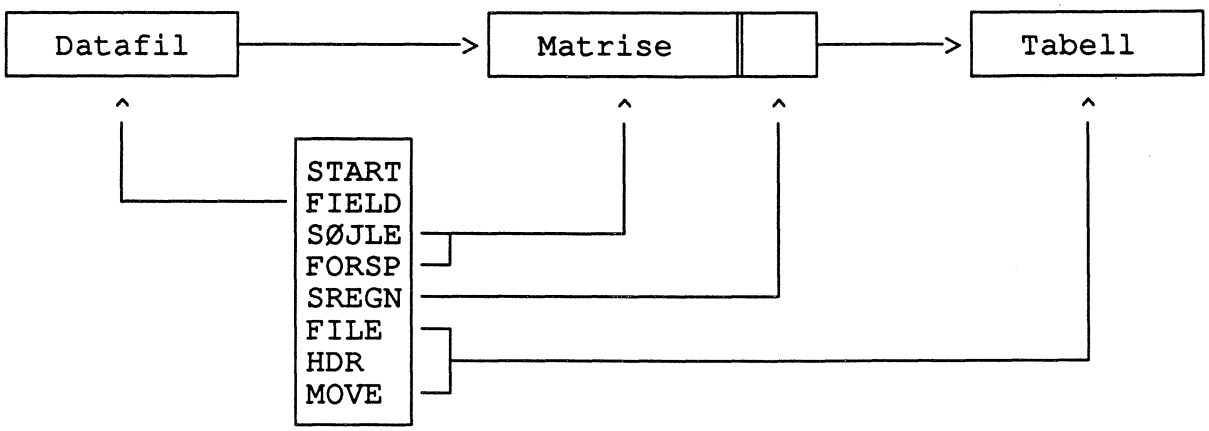
FILE bruker vi for å si til hvilken fil (vha. DDnavn) den ferdige tabellen skal sendes. Videre hva slags tabell vi skal lage.

HDR brukes til å lage overskrift til tabellen vår.

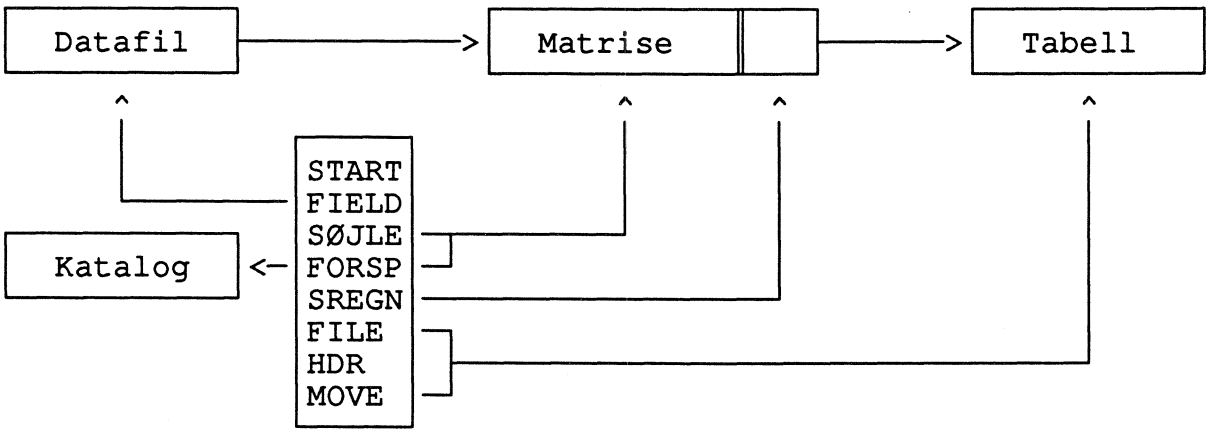
MOVE bruker vi for å flytte en eller flere kolonner ut til tabellen



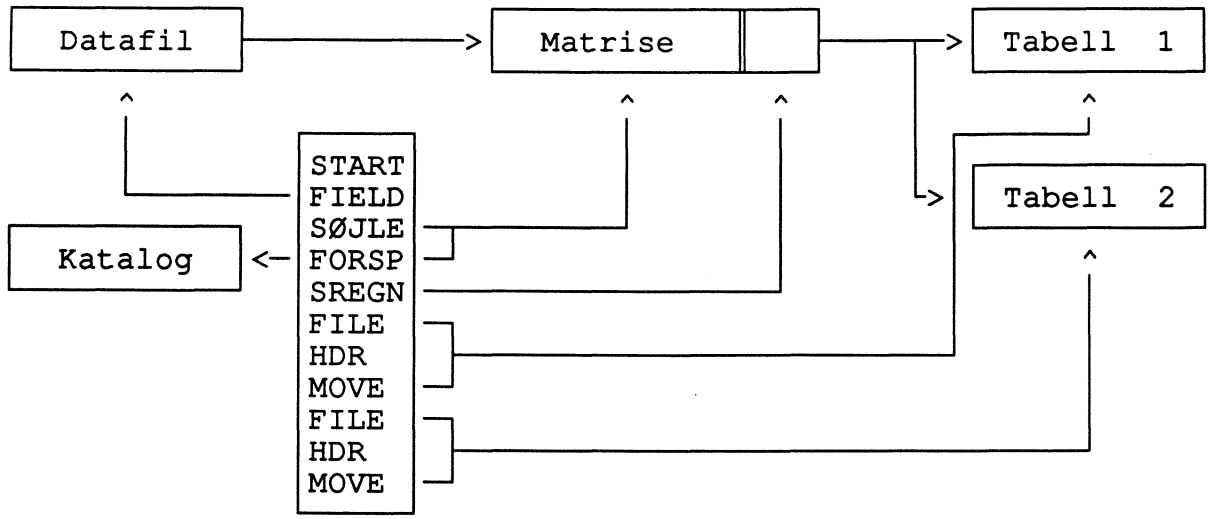
SØJLE og FORSP-instruksjonene i programmet vårt definerer en midlertidig matrise. Denne matrisen blir fylt opp ved gjennomkjøringen av innfilen. Når innfilen er gjennomlest, vil
vi stå igjen med denne matrisen. All videre bearbeiding programmet gjør, skjer på denne matrisen eller en utvidelse av den.



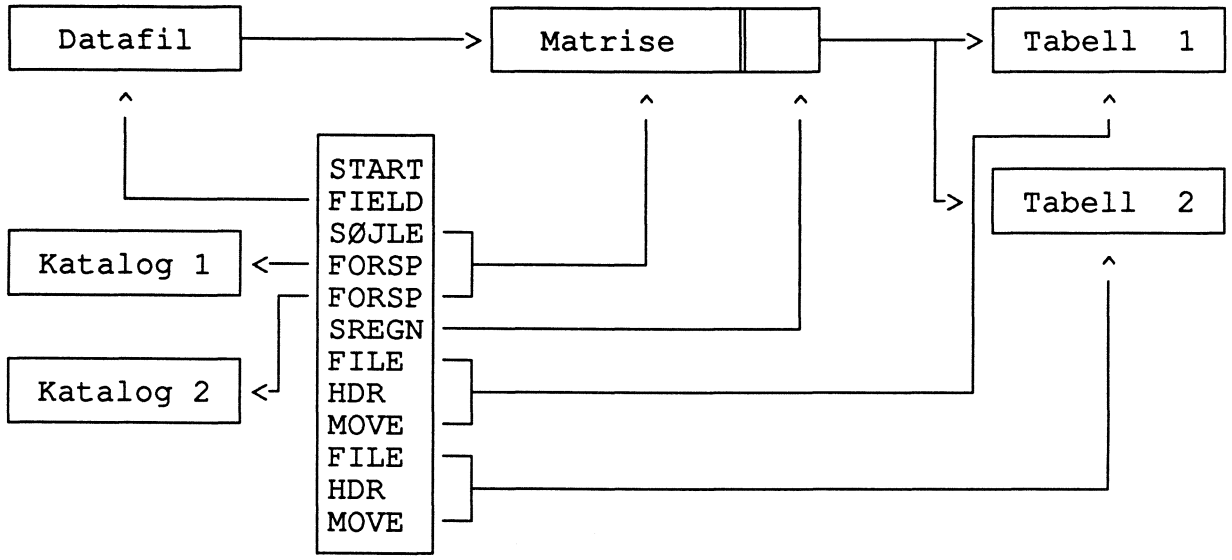
SREGN lager nye kolonner (utvider matrisen) eller endrer innholdet i eksisterende kolonner.



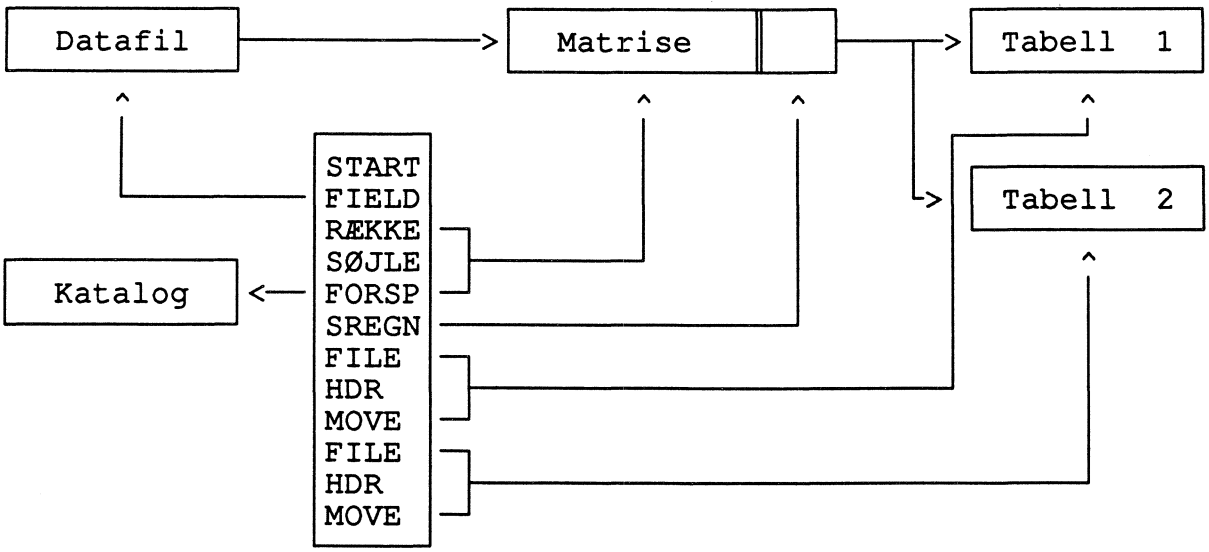
Når vi skal ha ut tekster istedenfor koder i forspalten, bruker vi kataloger som inneholder alle mulige koder med tilhørende tekster. Katalogen må være sortert stigende på kodene. Vi beskriver katalogen i FORSP-instruksjonen.



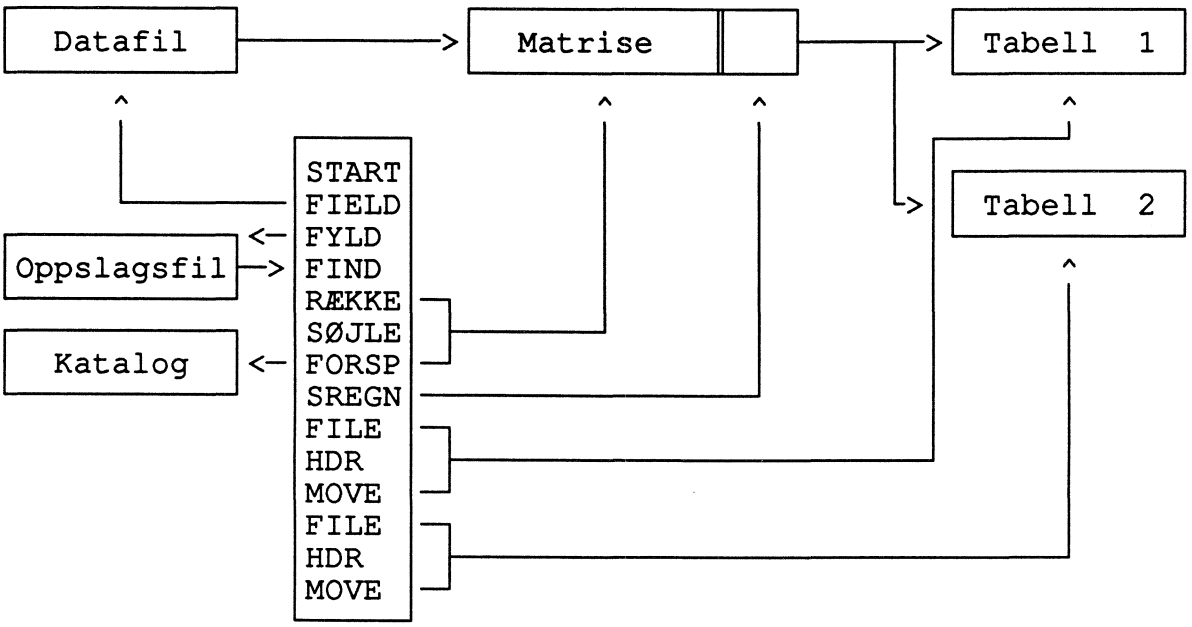
Når vi skal lage mer enn en tabell i et program, definerer vi først alle kolonner som skal være med i tabellene. Forspalten må være lik for alle tabellene, så den endrer vi ikke nå. For hver tabell må vi ha med FILE, HDR -og MOVE-instruksjonene i rekkefølgen som vist på figuren over.



Har vi flere forspaltekriterier som skal ha tekster, må vi ha en katalog for hvert.



Når vi bruker TAB2, må vi ha med instruksjonen **RÆKKE**. Denne bruker vi til å definere hver enkelt linje i tabellen, en RÆKKE-instruksjon pr. linje. FORSP-instruksjonen vil nå bli brukt til å beskrive hvilken tekst hver enkelt linje skal få. Det betyr at rekkenr blir koblingsbegrep mellom tall og tekst. Tekstkatalogen i TAB2 har en annen oppbygning en den har i TAB1.



FYLD brukes til å definere en oppslagsfil. En oppslagsfil er en fil som vi på grunnlag av en oppslagskode (nøkkel) kan slå opp i og hente informasjon fra. Dette kan være data vi trenger for å lage tabellen vår, men som ikke finnes på datafilen vår.

Med **FIND** gjør vi et oppslag fra oppslagsfilen. På grunnlag av den verdi oppslagskoden vi søker på har, får vi hentet den informasjonen som finnes på oppslagsfilen til denne koden. Denne informasjonen legges i et arbeidsfelt (WORK-felt).
FYLD og FIND kan også brukes i TAB1A og TAB1B. FIND kan også brukes i forbindelse med SREGN.

11.3. TAB-PROGRAMMETS UTFØRELSE

TAB-program kan deles opp i seks faser:

Initiering (SETUP).
Forbehandling (MAIN) og opptelling (TAELOP).
Summeringer.
Beregninger. (SREGN, POSTCP)
Sortering.
Utskrift.

Initiering:

Før TAB-programmet leser den første record fra innfilen, foretas det en initiering. Alle systemvariabler og arbeidsfelt gis her startverdi. I denne fasen kan vi også gjøre forskjellige ting (f.eks nullstille indekserte arbeidsfelt).

Når vi skal lage vår egen initieringsfase, må vi si fra til TAB om dette. Det gjør vi ved å starte initieringen med SETUP i posisjon 1-5 på en egen linje før den første forbehandlingsinstruksjonen. Dessuten må vi si ifra når initieringen er ferdig. Dette gjør vi med instruksjonen STOP. Ikke nok med det, men vi må også si ifra til TAB at nå er alt klart for å starte forbehandlingen av første record. Det gjør vi ved å skrive MAIN i posisjon 1-4 på linja etter STOP-instruksjonen.

Eksempel på intiering av indeksert arbeidsfelt:

```
          START TYPE=P
PNR       FIELD (1,11,X)
FØDSÅR    FIELD (6,2,X)
ALDER     WORK  (3,X)*8
SETUP
          LOOP  TALLY8,1,8,1
          SET   ALDER(TALLY8)=0
          ENDLOOP
          STOP
MAIN
          TEST  100
          :
          resten av programmet
          :
```

Forbehandling og opptelling:

Både forbehandling og opptelling utføres én gang pr. lest record fra innfilen (Innfilen beskrives ved hjelp av FIELD-instruksjonen, én FIELD-instruksjon for hvert felt fra innfilen vi skal bruke i programmet vårt). Forbehandling er frivillig, den bruker vi til å utføre eventuelle omkodinger og andre operasjoner på dataene som er være nødvendige å gjøre før opptellingen foretas. Dataene som blir talt opp lagres i en midlertidig matrise. Denne matrisen lages automatisk på grunnlag av SØJLE- og FORSP-instruksjonene våre.

For hver søyle vi definerer lages det en kolonne i matrisen. Linjene i matrisene bestemmes av forspalten (FORSP). Det lages en linje i matrisen for hver kombinasjon av verdier forspaltekriteriene våre har. Antall kolonner i matrisen er gitt ved SØJLE-instruksjonene våre. Antall linjer avgjøres av hvor mange kombinasjoner av verdier det finnes av forspaltekriteriene våre på innfilen. Det betyr at matrisen utvides med en ny linje hver gang en record har en kombinasjon av verdier for forspaltekrieteriene som ikke finnes i matrisen fra før. Når programmet har behandlet alle records i innfilen, vil matrisen være klar til videre behandling. Innfilen vil ikke lenger være nødvendig og vil derfor ikke lenger være tilgjengelig. De resterende fasene i TAB-programmet utføres på matrisen.

Vi illusterer dette med et enkelt eksempel. Vår innfil inneholder 3 felt; navn, kjønn og inntekt. Vår tabell skal ha kjønn i forspalten og vi skal ha én kolonne med antall personer og én med inntekt.

Programmet vårt kan se slik ut:

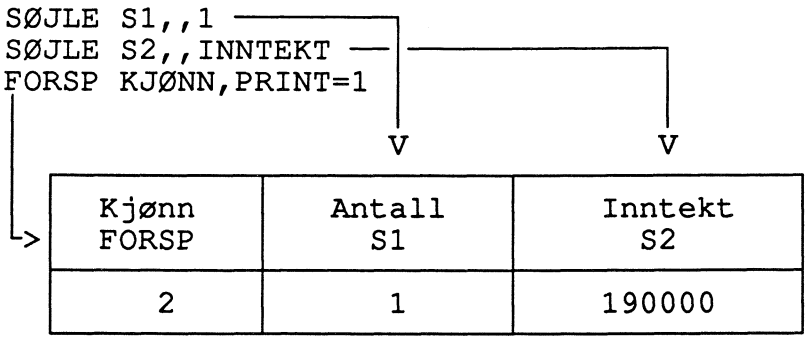
```
         START TYPE=P
NAVN     FIELD (1,8,X)
KJØNN    FIELD (9,1,X)
INNTEKT  FIELD (11,8,X)
         SØJLE S1,,1
         SØJLE S2,,INNTEKT
         FORSP KJØNN,PRINT=1
TABELL   FILE  PRINT,OVERFLOW,MAXLIN=46
         HDR   (1-3),(1-72)
Tabell 1. Samlet inntekt fordelt på kjønn.

Kjønn       Antall     Inntekt
         MOVE  (11,6,Z),S1
         MOVE  (23,10,Z),S2
```

I dette lille programmet vil vår matrise bestå av to kolonner, én for antall personer og én for sum inntekt. Linjene i matrisen bestemmes av forspalten, som er kjønn. Det betyr at det blir laget en linje for hver verdi kjønn har i innfilen.

Vi kjører programmet vårt på en innfil som denne:

```
Kari     2 00190000
Liv      2 00230000
Per      1 00195000
Tor      1 00130000
Anne     2 00260000
Jens     1 00240000
```

Etter første record (Kari), vil matrisen se slik ut:



Kjønn FORSP	Antall S1	Inntekt S2
2	1	190000

Etter andre record (Liv), vil matrisen se slik ut:

Kjønn FORSP	Antall S1	Inntekt S2
2	2	420000

Etter tredje record (Per), vil matrisen se slik ut:

Kjønn FORSP	Antall S1	Inntekt S2
2	2	420000
1	1	195000

Etter siste record (Jens), vil matrisen se slik ut:

Kjønn FORSP	Antall S1	Inntekt S2
2	3	680000
1	3	565000

Når hele innfilen er gjennomgått og talt opp i vår matrise, har vi ikke bruk for innfilen mer. Resten av programmet bruker de ferdige opptelte dataene i matrisen.

Opptellingsfasen er nå ferdig, og vår matrise er klar til videre behandling.

Summeringer:

Denne fasen utføres automatisk av TAB. Her summeres tallene i kolonnene til hvert forspaltenivå. Vår matrise får en sumlinje og vil se slik ut:

Kjønn FORSP	Antall S1	Inntekt S2
2	3	680000
1	3	565000
Sumlinje =>	6	1245000

Sortering:

Dette er også en fase TAB gjør automatisk. Matrisen vår sorteres stigende etter forspaltekriteriene. Etter sorteringen ser den slik ut:

Kjønn FORSP	Antall S1	Inntekt S2
	6	1245000
1	3	565000
2	3	680000

Beregninger:

Her kan vi lage nye kolonner i matrisen vår, eller vi kan endre innholdet i eksisterende kolonner. Vi tar utgangspunkt i vår lille matrise og vil gjerne regne gjennomsnittlig inntekt. Det gjør vi ved å legge til en slik SREGN-instruksjon etter FORSP-instruksjonen:

```
SREGN S3=S2/S1
```

Matrisen vil nå utvides med en kolonne (S3). Det betyr at regnestykket vil bli utført for hver linje i matrisen. Matrisen vil nå se slik ut:

Kjønn FORSP	Antall S1	Inntekt S2	Snitt S3
	6	1245000	207500
1	3	565000	188333
2	3	680000	226667

Utskrift:

I utskriftsfasen skriver vi ut matrisen eller de delene av den (kolonnene) vi har bruk for. Vi bruker MOVE-instruksjonen for å fortelle programmet hvor kolonnene skal plasseres i tabellen. MOVE-instruksjonen forteller hvilken kolonne som skal skrives ut og hvor i tabellen den skal stå. Vi trenger normalt én MOVE-instruksjon for hver kolonne som skal skrives ut. Hver linje i matrisen vil bli skrevet ut slik MOVE-instruksjonene tilsier. Overskriften vi lager med HDR-instruksjonen vil bli skrevet ut øverst på hver side i tabellen.

Vi endrer litt på HDR- og MOVE-instruksjonene i vårt lille program:

```
        HDR   (1-3),(1-72)
Tabell 1. Samlet inntekt fordelt på kjønn.
                                          Gjennomsnittlig
Kjønn       Antall     Inntekt                inntekt
        MOVE  (11,6,Z),S1
        MOVE  (23,10,Z),S2
        MOVE  (40,10,Z),S3
```

Med disse HDR- og MOVE-instruksjonene vil den ferdige tabellen vår få dette utseende:

Tabell 1. Samlet inntekt fordelt på kjønn.

Kjønn	Antall	Inntekt	Gjennomsnittlig inntekt
Total	6	1245000	207500
1	3	565000	188333
2	3	680000	226667

Kort oppsummering:

Initieringen skjer før innfilen leses.
Forbehandling og opptelling gjøres på data fra innfilen og lagres i en midlertidig matrise.
Summeringene utføres på den midlertidige matrisen.
Beregninger utvider den midlertidige matrisen med en eller flere nye kolonner.
Sortering gjøres stigende på forspaltekriteriene i den midlertidige matrisen.
Utskrift skjer ved at hele eller deler av den midlertidige matrisen skrives ut (etter våre spesifikasjoner).

11.4. BETYDNING AV DE ULIKE INSTRUKSJONER

Alle instruksjoner og parametre skal skrives med STORE bokstaver. Tekststrenger (overskrifter osv.) kan skrives med små bokstaver. De instruksjoner som er understreket skal være med i programmet. Alle andre instruksjoner er frivillige. Det samme gjelder for parametrene, de som er understreket må være med hvis instruksjonen brukes, resten er frivillige.

Komma (,) brukes som skille mellom parametrene i programmet. Hvis 2 eller flere parametere er adskilt med skråstrek (/), kan man velge en av dem.

Eksempler:

```
TABEL
TOTAL
FORSP
```

Her ser vi at av disse tre er det bare FORSP vi må ha med. Når vi viser syntaksen til instruksjonene er både instruksjonsnavnet og de nødvendige parametre understreket. Dette fordi når vi velger å bruke en instruksjon, må instruksjonsnavnet være med for at syntaksen skal bli riktig, selv om instruksjonen er frivillig (se under).

```
TABEL FELTNAVN,PRINT=,PAGE/PAGE1/CPAGE,SKIP=,TOTKEY=,
      SELECT=/SUPRESS=
```

Når vi bruker denne instruksjonen må vi ha med TABEL, FELTNAVN og PRINT=. Dessuten kan vi (hvis vi trenger) ha med SKIP=, TOTKEY= og enten PAGE eller PAGE1 eller CPAGE og enten SELECT= eller SUPRESS=, f.eks:

```
TABEL FELTNAVN=FYLKE,PRINT=(3,2),PAGE,TOTKEY=00
```

Når vi refererer til karakterstrenger, skal disse stå i fnutter (') (se eks. under). Tall behøver ikke å stå i fnutter.

Et kolon (:) alene på en linje symboliserer at det skal stå en eller flere instruksjoner istedenfor :.

Eksempel:

```
IF    (BETINGELSE)
      :
      ELSE
      :
      ENDIF
```

Dette betyr at vi må ha med en eller flere instruksjoner etter IF, og at vi kan ha med en eller flere instruksjoner etter ELSE, for eksempel slik:

```
IF    (S1 NE 1,2)
MOVE  (10,10,E-)*3,S1,ZZ ZZZ ZZZ
ELSE
MOVE  (10,10,X),'          :'       <=== Karakterstreng
MOVE  (20,10,E-)*2,S2,ZZ ZZZ ZZZ
ENDIF
```

11.5. DEFINISJONER

I definisjonsdelen av programmet vårt definerer vi alle felt som vi skal bruke fra input-filen. Hvis vi trenger å bruke noen arbeidsfelt, (work-felt) må de også defineres her. Vi må ha med en start-instruksjon der vi definerer hvordan den interne opptellingen skal foregå (start-instruksjonen MÅ stå først i programmet).

Hvis vi trenger å bruke FYLD-instruksjonen, skal den også være med her (Se FYLD).

Instruksjoner i definisjonsdelen (i riktig rekkefølge):

START
FIELD
GROUP
WORK
FYLD

Eksempel:

```
          START TYPE=B
KOMMUNE   FIELD (1,4,X)
PERSONNR  FIELD (5,11,X)
KJØNN     WORK  (1,X)
OVERSKR   GROUP (170)
NORSK     WORK  (85,X)
ENGELSK   WORK  (85,X)
```

Som vi ser av eksemplet, kan vi ha flere FIELD-instruksjoner. Vi kan også ha flere WORK- og GROUP-instruksjoner.

START er første instruksjon i programmet

Syntaks:

START TYPE=,NULDIV=,TXTFEJL=

Forklaring:

TYPE=B/P - Angir hvilket format den interne opptelling skal foregå i.

B = Binære fullord (max 9 siffer)

P = Pakket desimal (max 15 siffer)

Hvis vi skal telle 1 pr. record som leses, bør vi bruke binært format (TYPE=B). Hvis vi skal telle opp verdien til et felt (f eks. beløp og mengder), bør vi bruke pakket format (TYPE=P).

NULDIV=(A,B) Angir resultatet av divisjon med null
(Gjelder bare i SREGN og RREGN)

A - Angir resultat av 0/0.

B - Angir resultat av X/0, X ulik 0

A og B kan angis sum et tall, eller som 'I' (Invalid ==> ABEND)

Standardverdi for NULDIV=(0,I)

TXTFEJL=MAX

Angir det maksimale antall feil som aksepteres i txt-fil (samme feil kan telles flere ganger, 1 for hver gang teksten skal skrives ut til tabellen). Standardverdi er 10.

Eksempler:

```
START TYPE=P       <=== Vanlig START-instruksjon

START TYPE=P,NULDIV=(0,0)

START TYPE=B,NULDIV=(0,0),TXTFEJL=15
```

FIELD navngir bestemte felter i inputrecorden med startposisjon og lengde.

Syntaks:

FELTNAVN FIELD (STARTPOSISJON,LENGDE,TYPE)*ANTALL

Forklaring:

FELTNAVN: 1-8 Alfanumeriske karakterer. Første skal være bokstav, dog ikke Æ, Ø eller Å.

STARTPOSISJON: Feltets startposisjon i recorden (Hvis man skriver * blir startposisjonen første posisjon etter sistnevnte FIELD-spesifikasjon.)

LENGDE: Feltets lengde i bytes (antall tegn)

TYPE:

X = Alfanumerisk (lengde max 4096 tegn)

B = Binær (lengde 2 eller 4 bytes)

P = Pakket desimal (lengde max 16)

ANTALL: Indeksering (Array). Max = 500

Eksempel:

```
ALDGRP1  FIELD (10,5,X)
ALDGRP2  FIELD (15,5,X)
ALDGRP3  FIELD (20,5,X)
```

Disse tre instruksjonene kan også skrives slik:

```
ALDGRP   FIELD (10,5,X)*3
```

Da vil feltene hete ALDGRP(1), ALDGRP(2) og ALDGRP(3) når vi skal referere til dem i programmet vårt.

WORK Definerer et arbeidsfelt (Work-felt)

Syntaks:

FELTNAVN WORK (LENGDE,TYPE),STARTVERDI

Forklaring:

FELTNAVN: 1-8 Alfanumeriske karakterer. Første skal være bokstav.

LENGDE: Feltets lengde i bytes (antall tegn)

TYPE:

X = Alfanumerisk (lengde max 4096 tegn)
B = Binær (lengde 2 eller 4 bytes)
P = Pakket desimal (lengde max 16)

STARTVERDI: Konstant/SPACES/ZEROES/LOW-VALUES/HIGH-VALUES

Eksempler:

```
AARGANG   WORK  (4,X)
FYLKNAVN  WORK  (20,X),SPACES
TELLER    WORK  (4,X),1
BOKSTAV   WORK  (1,X),'B'
```

GROUP Definerer en rekke work-felter som en gruppe

Syntaks:

FELTNAVN GROUP (LENGDE),ANTALL

Forklaring:

FELTNAVN: 1-8 Alfanumeriske karakterer. Første skal være bokstav.

LENGDE: Feltets lengde i bytes

ANTALL: Indeksering (Array). Max = 500

Eksempel:

```
OVERSKR  GROUP (170)
NORSK    WORK  (85,X)
ENGELSK  WORK  (85,X)
```

Poenget med å bruke GROUP er at i det vi gir GROUP-feltet en verdi, vil vi samtidig automatisk gi WORK-feltene som hører til i gruppen (GROUP) verdier. Dette er spesielt nyttig ved bruk av FYLD og FIND (se disse og vedlegg IV).

Se også KORT/PARM for hvordan du kan hente inn konstanter utenfor programmet til work-felt i programmet.

NB! Husk at et arbeidsfelt alltid har sin verdi helt til den får en ny. Se vedlegg V for et eksempel på hvor galt det kan gå når vi ikke er klar over dette.

Innlesing av opplysninger fra JCL til work-felter i programmet. Dette gjøres for å slippe å endre selve programmet for hver gang det skal kjøres. KORT/PARM kan ha lengde på inntil 80 posisjoner.

Syntaks:

(1)
FELTNAVN WORK (LENGDE,TYPE),KORT(STARTPOSISJON,LENGDE)

(2)
FELTNAVN WORK (LENGDE,TYPE),PARM(STARTPOSISJON,LENGDE)

Eksempel:

```
//TELLING   EXEC  TAB1B,PARM='TXT=1986'
//INPUT     DD    DSN=TK414.S8019.KRL.FOLK,DISP=OLD
//TABELL    DD    SYSOUT=*
//KORT      DD    *
SEPTEMBER 3. KVARTAL
/*
//SYSIN     DD    *
          START TYPE=P
AARGANG   FIELD (1,4,X)
KOMMUNE   FIELD (5,4,X)
FNR       FIELD (9,11,X)
AAR       WORK  (4,X),PARM(1,4)
MÅNED     WORK  (9,X),KORT(1,9)
KVARTAL   WORK  (10,X),KORT(11,10)
:
:
          HDR   1,54,AAR
          HDR   3,24,MÅNED
          HDR   4,2,KVARTAL
```

Her vil '1986' bli hentet fra JCL (PARM='TXT=1986') og satt inn i arbeidsfeltet AAR, 'SEPTEMBER' hentes fra posisjon 1-9 fra filen KORT og vil bli satt inn i arbeisdfeltet MÅNED og KVARTAL vil få verien filen KORT har i posisjon 11-20: '3. KVARTAL'. Disse feltene brukes i dette tilfelle i overskriften til tabellen.

Jeg anbefaler å bruke PARM framfor KORT.

Tips: Når du lager program som skal kjøres jevnlig, med kun små endringer i programmet for hver gang (slik som å endre måneder og år), bør du prøve å trekke disse ut av programmet. Det finnes 2 måter å gjøre det på:

1. Bruk dataene i innfilen! Hvis årstall ligger på filen og programmet skal kjøres årlig, bruk det (se eksempel under HOP)!
2. Hvis dine små endringer ikke kan ordnes ved å bruke data fra innfilen, bruk PARM (se over)!

FYLD brukes til å lage en oppslagsfil. Oppslagsfiler brukes når vi trenger data som ikke ligger på input-filen vår. Det kan for eksempel være befolkningstall hvis vi skal lage tabeller med ratetall pr. 1000 innbyggere (Se vedelgg III for eksempel på dette.). Oppslagsfilen må ha et nøkkelfelt og et datafelt. Hvor disse er plassert i oppslagsfilen, skal du fortelle programmet med FYLD-instruksjonen. Du må også si hvor programmet kan finne filen (oppgi DDnavn). Ved hjelp av nøkkelfeltet vil vi ved hjelp av en FIND-instruksjon (se denne) finne fram til den riktige recorden i oppslagsfilen. Dataene som vi henter med FIND, kan vi så bruke i beregninger i programmet vårt.

NB! Oppslagsfiler må ikke forveksles med INPUT-filen. Oppslagsfiler bruker vi når vi trenger data fra en annen fil enn innfilen for å få laget tabellen vår. Vi kan ha opp til 10 oppslagsfiler i et program.

Syntaks:

DDNAVN FYLD KEY=,DATA=,MAX=

Forklaring:

DDNAVN: DDnavn på datasett (kan velges fritt, men må følge JCL-syntax)

KEY: (START,LENGDE,TYPE) Angir hvor i oppslagsfilen nøkkelfeltet er.

DATA: (START,LENGDE,TYPE) Angir hvor i oppslagsfilen dataene finnes.

MAX: Max antall records i en oppslagsfil. Vi kan godt sette dette tallet noe større enn antall records i oppslagsfilen, men vi kan ikke sette det mindre enn antallet.

Når vi bruker FYLD-instruksjonen må vi angi navnet på filen i JCL, og det DDnavnet vi bruker i programmet må vi også bruke i JCL.

Eksempel på JCL:

```
//FYLDFIL  DD  DSN=TK414.S8019.KRL.FYLD,DISP=SHR
```

DDnavnet (her: FYLDFIL) må korrespondere med DDnavnet i programmet.

Eksempler på FYLD; se SETUP, FIND og VEDLEGG IV.

11.6. SELEKSJON

Seleksjon brukes for å selektere ut de records vi skal bruke i programmet vårt. Hvis vi vet at den tabellen vi skal lage bare omfatter en del av de records som er på input-filen, bør vi sørge for å selektere bort de andre i denne delen av programmet. Da vil programmet vårt bli mer effektivt, fordi programmet ikke behøver å gjennomgå forbehandlingsfasen og opptellingsfasen for hele innfilen, men bare for de records du trenger.

Seleksjon kan vi også bruke ved testing av et program. Under programmeringen av tabellen er det lite hensiktsmessig å kjøre på hele datafilen. Derfor bør du alltid bruke kun en del av datafilen i programmeringsfasen. Det er tidsbesparende både for deg og maskinen (husk at tid er penger!). Den beste måten å begrense datamengden under programmeringen er å bruke TEST-instruksjonen (se denne). Men det er ikke alltid vi får sjekket nok bare ved hjelp av TEST. Vi har også en mulighet ved å bruke seleksjon av data. Dette vil være tidsbesparende, og selv om det ikke er like effektivt som å bruke TEST, vil ofte mulighetene for å sjekke tallene i tabellen være større. TEST og SELECT kan brukes samtidig, dette gir oss bedre muligheter for kontroll av tabellene før de kjøres på hele datamaterialet.

Til seleksjon kan vi bruke denne instruksjonen (bare 1):

SELECT

Eksempel:

```
SELECT    INPUT (LAND = 101-133,464,684 AND
                 (VARENR = 3902211,3902215 OR VERDI > 5000000))
```

SELECT Brukes for å velge (selektere) records

Syntaks:

SELECT INPUT NOT(BETINGELSE)

Forklaring:

BETINGELSE: Seleksjonskriterium, logisk uttrykk som skrives i parentes. Se logiske uttrykk.

Eksempler:

```
SELECT   INPUT (KOMMUNE = 0211)

SELECT   INPUT NOT(KOMMUNE = 0211)

SELECT   INPUT (KJØNN = 2 AND ALDER = 33)

SELECT   INPUT (MÅNED1 = MÅNED2)

SELECT   INPUT (VERDI >= 1000 AND VARENR = 0101010-2820100,
                                           3601001-3907999,
                                           8904001,9703900 AND
                                    LAND = 101-199)

SELECT   INPUT (FYLKE <= 16 AND FYLKE >= 4)
SELECT   INPUT (FYLKE = 4-16)
```

(De 2 eksemplene over gir samme resultat, men det lønner det seg å bruke det siste av dem)

```
SELECT   INPUT (INIT = 'AAA'-'KRL')
SELECT   INPUT (KODE NE 'B','K',' ',1-5)
```

11.7. FORBEHANDLING

I forbehandlingsdelen kan vi utføre vanlige programmeringsfunksjoner. Vi kan blant annet lage nye variable på grunnlag av tester, vi kan utføre regneoperasjoner, lete i tabeller (FIND, se dette), omkode data, hoppe nedover (eller oppover) i programmet (på grunnlag av tester) og vi kan kalle opp sub-program.

Disse instruksjonene kan vi bruke i forbehandlingen:

ABEND
CALL
CASE - ENDCASE
FIND
HOP
IF - ELSE - ENDIF
LOOP - ENDLOOP
SET
SOEG
STOP
TAEL
TEST

Det er 2 forskjellige forbehandlinger:

Forbehandling før 1. record leses

Forbehandling før opptelling (gjøres for hver lest record)

Forbehandling før 1. record gjør vi hvis vi vil beregne work-felt før vi begynner å lese fra input-filen. Se SETUP.

I TAB1 kan vi foreta en forbehandling for den første input-record som har en ny kombinasjon av verdier i forspaltekriteriene. Denne forbehandlingen kan benyttes hvis vi f. eks. vil beregne konstanter som gjelder for alle records i gruppen. Se IDSTART.

Den vanligste forbehandlingen er den som gjøres for hver innlest record. Dette gir oss store muligheter til å få tilpasse dataene i input-filen til tabellen vi skal lage, f. eks. kan vi kode om felt fra input-filen og utføre testing på dataene her.

SETUP - Instruksjonene som kommer mellom SETUP og MAIN vil bli utført før den første inputrecorden leses. SETUP gir mulighet for å hente inn verdier til og beregne work-felter. SETUP må avsluttes med STOP, dessuten må vi skrive MAIN i posisjon 1-4 på linja etter STOP. Dette må vi for å si at nå begynner forbehandlingen (MAIN). Se eksemplet under.

Eksempel:

```
BOSATTM  WORK  (8,X),ZEROES        *  LANDSTALL, BOSATTE, MENN
BOSATTK  WORK  (8,X),ZEROES        *  LANDSTALL, BOSATTE, KVINNER
NKL      WORK  (2,X),ZEROES        *  NØKKEL

BOSATT   FYLD  KEY=(1,2,X),DATA=(50,7,X),MAX=20
SETUP
         SET   NKL=10
         FIND  ARG=NKL,DATA=BOSATTM,DDNAVN=BOSATT
         SET   NKL=NKL+10
         FIND  ARG=NKL,DATA=BOSATTK,DDNAVN=BOSATT
         STOP
MAIN
```

Her hentes først data fra en file med FYLD. Deretter (i SETUP) finner programmet fram til de riktige dataene fra FYLD-filen ved hjelp av SET og FIND. Dette avsluttes med STOP.

IDSTART - Instruksjonene som kommer mellom SETUP og MAIN vil bli utført for den første input-recorden som har en ny verdi for kombinasjonen av forspaltekriterier.

IDSTART må avsluttes med STOP. IDSTART kan bare brukes i underprogrammet TAB1.

Eksempel:

```
         START TYPE=P
KOMMNR   (7,4,X)
NÆRING   (11,1,X)
ARBTAK   (20,7,X)
TELLER   (3,X)
SELECT   INPUT (KOMMNR NE SPACES)
IDSTART  SET   TELLER=TELLER+1
         STOP
MAIN     TEST  100
         SØJLE S1
         SØJLE S2,,ARBTAK
         FORSP KOMMNR,NOSUM
TABELL   FILE  BLKSIZE=17170,LRECL=17
         MOVE  (1,3,X),TELLER
         MOVE  (4,4,X),KOMMNR
         MOVE  (8,5,X)*2,S1
```

I dette eksemplet skal vi telle opp arbeidstakere og antall records fordelt på kommunenummer. I tillegg skal vi gi hver kommune et løpenummer som starter på 1 med første kommune og så øker med 1 for hver ny kommune.

For å telle opp antall records pr. kommune, lager vi det første SØJLE-direktivet (SØJLE S1) som gjør akkurat det. Når vi skal lage løpenummeret, benytter vi IDSTART. Den instruksjonen som står i IDSTART (SET TELLER=TELLER+1) blir gjort hver gang det skjer et skifte i verdien til forspaltekriteriet, dvs. for hvert nytt kommunenummer. Fordi vi kjører programmet med TAB1 vil vi få skrevet ut telleren med riktig verdi til hver kommune. Det er fordi vi får utskrift for hver ny verdi av forspaltekriteriene (Det gjør vi ikke i TAB1A og TAB1B).

11.7.3. Abend **ABEND**

ABEND bruker du når du vil at programmet skal bråstoppe (Abend) under eksekveringen. Denne kommandoen kan være nyttig å bruke i forbindelse med IF- eller CASE-tester. Vi bruker ABEND-instruksjonen for å avbryte programmet når noe ikke er som det skal være. Det kan være at vi ikke vil at programmet skal fortsette hvis dataene våre er helt gale eller det har inntruffet "umulige" kombinasjoer av data (f.eks. kvinner med prostata). For at programmet skal avbrytes når slike uheldige situasjoner oppstår, må vi ha programmert slik at vi tester på disse gale eller "umulige" situasjoner og gir instruksjonen ABEND inne i testen(e). Når en ABEND-instruksjon bli utført, vil programmet avbrytes med feilkode U900, og det vil stå i kjøreloggen hvilken instruksjon som forårsaket avbruddet, hvor mange records som er lest osv. Dette vil danne et godt grunnlag for å se hva som er feil i dataene og å rette dem opp.

Eksempel:

```
        IF     (KJØNN = 1 AND DIAGNOSE = 180) Mann med livmorkreft
        ABEND
        ENDIF

        IF     (ALDER < 16 AND EKTSTAT = 1) Under 16 og gift!!!
        ABEND
        ENDIF
```

11.7.4. Call **CALL**

CALL - kaller et Sub-program

Syntaks:

```
LABEL    CALL  'PGMNAVN',FELT1,FELT2,.....
```

Eksempel:

```
DATOEN   WORK  (8,X)          * Felt som datoen legges inn i
SKILCHAR WORK  (1,X),'-'      * Skillekarakter; kan velges fritt
:
SETUP
HENTDATO CALL  'HENTDATO',DATOEN,SKILCHAR
         STOP
MAIN
:
:
         HDR   1,(70,8),DATOEN  * Datoen blir slik: DD-MM-ÅÅ
:
```

Dette programmet henter subrutinen HENTDATO. Den ligger på samme bibliotek som Tab-programmene og kan derfor kalles opp uten å ta med mer JCL. Oversikt over subrutinene som ligger på dette biblioteket og hva de gjør kan du få ved å kontakte undertegnede.

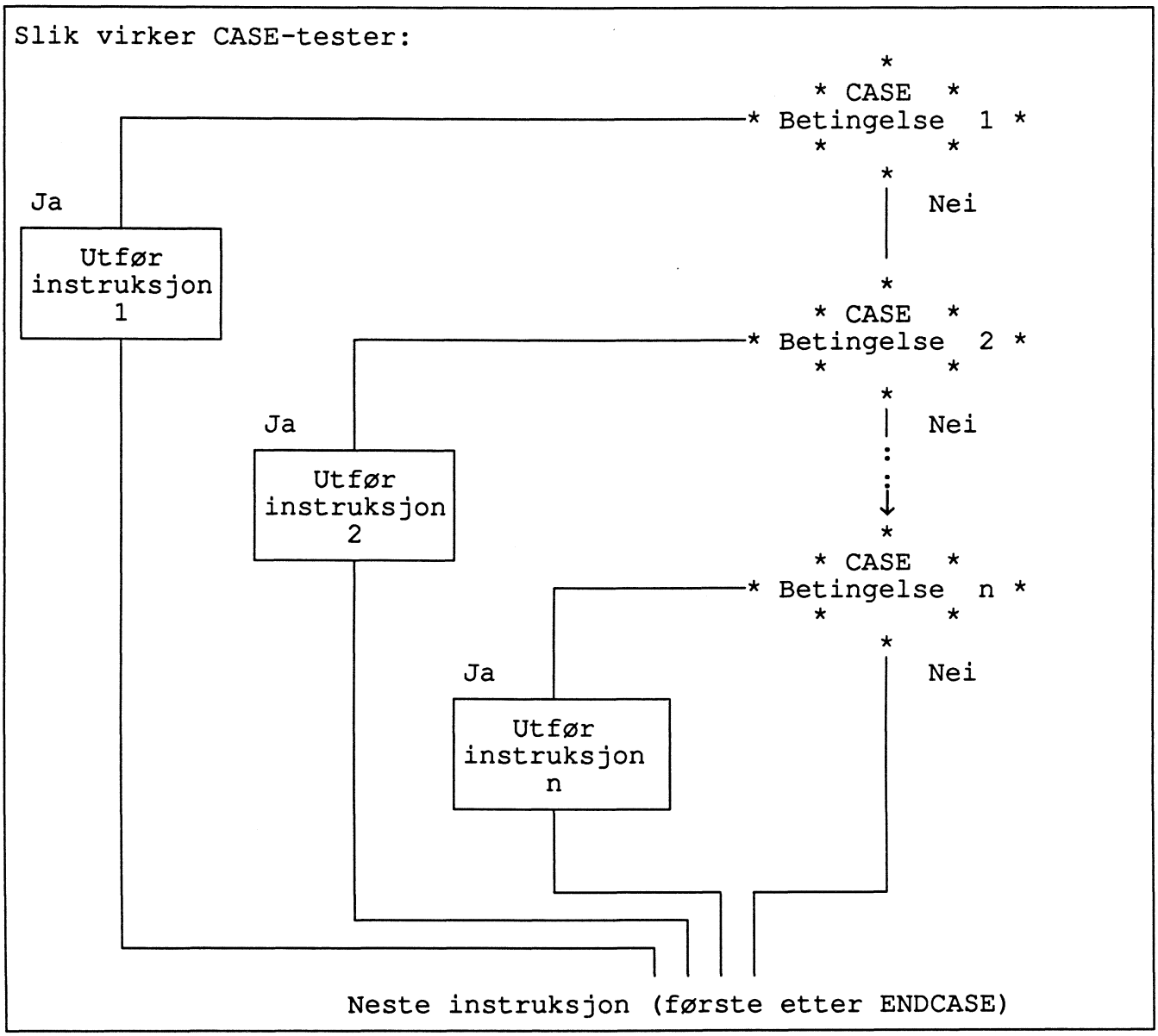
CASE er en IF-test med mulighet for fler en 2 utganger. Når en betingelse er oppfylt vil instruksjonene som kommer før neste CASE bli utført. Deretter vil programmet hoppe til første instruksjon etter ENDCASE. Programmet tester altså ikke om resterende CASE får sine betingelser oppfylt.

Vi kan ha nestede CASE (CASE inne i CASE) i opptil 40 nivåer.

Syntaks:

```
CASE  (BETINGELSE)
:
:
CASE  (BETINGELSE)
:
:
CASE
:
:
ENDCASE
```

Slik virker CASE-tester:



Når en betingelse i CASE-testen inntreffer, utføres de underliggende instruksjoner. Deretter hopper programmet ut av CASE-testen og utfører første instruksjon etter ENDCASE.

En CASE-test kan ha flere enn de tre betingelsene i diagrammet på forrige side, men er nok for å vise hvordan CASE-testen fungerer.

Du må være klar over at det går an å lage CASE-tester der det er slik at ingen av betingelsene slår til. Dette kan i verste fall gjøre at tabellen din vil inneholde gale tall. Vi forhindrer det ved å ta med en restgruppe slik som i eksemplet under.

```
CASE  (KJØNN = 1 AND SIVIL = 'U')
SET   A=1
CASE  (KJØNN = 1)
SET   A=2
CASE  (KJØNN = 2 AND SIVIL = 'U')
SET   A=3
CASE                    NB! Restgruppen
SET   A=4
ENDCASE
```

Her gjelder at betingelsene sjekkes nedover i programmet til den første sanne finnes. Når instruksjonene til denne er utført, fortsettes med instruksjonen etter ENDCASE.

Se under IF-instruksjonen når du bør bruke IF og når du bør bruke CASE.

FIND foretar tabelloppslag i en oppslagsfil dannet med FYLD.

Syntaks:

FIND ARG=,DATA=,SW=,DDNAVN=

Forklaring:

ARG=FELTNAVN Søkeargument til tabelloppslaget

DATA=FELTNAVN Her blir data fra oppslagsfilen lagt inn

SW=FELTNAVN (FELTNAVN skal defineres med LENGDE=1,TYPE=X)

SW=0 Hvis element ikke funnet
SW=1 Hvis element funnet

DDNAVN=FYLDFIL (DDnavnet på FYLD-tabellen)

Eksempel:

```
//0414KRL   JOB 8019,'Fyld og Find',MSGCLASS=X,CLASS=A,NOTIFY=0414KRL
//FINDEX    EXEC  TAB1A
//INPUT     DD    DSN=TK245.S0211.MAT5.G85K4.V00,DISP=OLD
//          DD    DSN=TK245.S0211.MAT5.G85K3.V00,DISP=OLD
//          DD    DSN=TK245.S0211.MAT5.G85K2.V00,DISP=OLD
//          DD    DSN=TK245.S0211.MAT5.G85K1.V00,DISP=OLD
//LANDSUM   DD    *
00 HELE LANDET 4145845
//TABELL    DD    SYSOUT=*
          START TYPE=P
AAR       FIELD (1,2,X)
KV        FIELD (3,1,X)
SKAFFE    FIELD (6,1,X)
OPPBL2    FIELD (16,8,X)
AARSTALL  WORK  (2,X)
LANDKODE  WORK  (2,X),0
INNB      WORK  (7,X)
LANDSUM   FYLD  KEY=(1,2,X),DATA=(16,7,X),MAX=1
SETUP     FIND  ARG=LANDKODE,DATA=INNB,DDNAVN=LANDSUM
          STOP
MAIN      SET   AARSTALL=AAR,ABS
TAELOP    SØJLE S1:S4,(KV = 1:2:3:4),OPPBL2
          FORSP SKAFFE,PRINT=3
          SREGN S1:S4=SN*10/INNB
TABELL    FILE  PRINT,OVERFLOW,MAXLIN=46
          HDR   1,1,'TABELL 14. ARBEIDSSØKERE UTEN ARBEIDSINNTEKT'
          HDR   2,12,'ETTER MÅTE Å SØKE PÅ.'
          HDR   3,12,'PROSENT AV LANDETS INNBYGGERE. 19'
          HDR   3,45,AARSTALL
          HDR   5,12,'1. KV.    2. KV.    3. KV.    4. KV.'
          MOVE  (7,10,E-)*4,S1, ZZZ ZZ9,9
```

Fordi vi ikke har opplysningen om landets innbyggere på innfilen, bruker vi FYLD for å legge tallet inn i et work-felt. Deretter, før vi leser første record fra input-filen (i SETUP), henter vi tallet til work-feltet. Deretter kan vi bruke work-feltets verdi til beregninger slik som vi gjør i SREGN i dette programmet.

Se Vedlegg IV for et annet eksempel på bruk av FIND-instruksjonen.

HOP brukes til å hoppe nedover eller oppover i programmet. Vi kan bare hoppe inne i forbehandlingen eller beregningsfasen, men ikke mellom dem. Hvis vi skal hoppe oppover i programmet må vi ta med PARM=LOOP når vi kaller opp programmet (se JCL i eksemplet).

```
LABEL1   HOP   LABEL2                ===>  hoppes alltid til LABEL2
LABEL1   HOP   NF,LABEL2            ===>  hoppes alltid til LABEL2,
                                          unntatt første gang
LABEL1   HOP   (BETINGELSE),LABEL2  ===>  hoppes hvis betingelse er
                                          sann
```

Eksempel:

```
//0414KRL   JOB 8019,'Hop-eksempel',MSGCLASS=X,CLASS=A,NOTIFY=0414KRL
//HOPPEX    EXEC  TAB1A,PARM=LOOP
//INPUT     DD    DSN=TK245.S0211.MAT5.G85K4.V00,DISP=OLD
//          DD    DSN=TK245.S0211.MAT5.G84K4.V00,DISP=OLD
//          DD    DSN=TK245.S0211.MAT5.G83K4.V00,DISP=OLD
//TABELL    DD    SYSOUT=*
          START TYPE=P
AAR       FIELD (1,2,X)
SKAFFE    FIELD (6,1,X)
OPPBL2    FIELD (16,8,X)
AARF      WORK  (2,X)
AARM      WORK  (2,X)
AARS      WORK  (2,X)
          HOP   NF,TAELOP
          SET   AARS=AAR
          SET   AARM=AAR-1,ABS
          SET   AARF=AAR-2,ABS
TAELOP
          SØJLE S1:S3,(AAR = AARF:AARM:AARS),OPPBL2
          FORSP SKAFFE,PRINT=3
          SREGN S1:S3=SN/100
TABELL    FILE  PRINT,OVERFLOW,MAXLIN=46
          HDR   1,1,'TABELL 14. ARBEIDSSØKERE UTEN ARBEIDSINNTEKT'
          HDR   2,12,'ETTER MÅTE Å SØKE PÅ'
          HDR   4,19,'19'
          HDR   4,21,AARF
          HDR   4,29,'19'
          HDR   4,31,AARM
          HDR   4,39,'19'
          HDR   4,41,AARS
          MOVE  (14,10,E-)*3,S1,ZZ ZZZ ZZZ
```

Dette programmet skal lage en tabell med en kolonne for hvert år. Årstall står på hver eneste record på input-filen. Problemet er å få riktig år i riktig kolonne uten å skrive noen årstall i programmet vårt (det ville bety at vi måtte endre programmet hver gang det skal kjøres med nye år). Det vi gjør er å lage programmet slik at det blir den første leste record fra innfilen som avgjør hvilke år som skal være med i tabellen. Dette gjør vi ved å bruke HOP. Ved å bruke HOP som vist i programmet over, vil forbehandlings-instruksjonene mellom HOP og TAELOP bare bli gjort for den første recorden som leses. Her settes årstall til de årene vi skal ha med. Disse brukes til å teste på i opptellingen (SØJLE), og de brukes til å lage overskrift (HDR).

IF brukes hvis instruksjoner bare skal utføres hvis en betingelse er oppfylt. Hvis betingelsen er oppfylt, vil alle instruksjoner til ELSE eller ENDIF bli utført. Hvis ELSE er med vil instruksjonene mellom ELSE og ENDIF bli utført hvis betingelsen ikke er oppfylt. Vi kan neste IF (IF inne i IF) i opptil 40 nivåer. Dessuten kan vi ha CASE og/eller LOOP både utenfor og inne i IF-strukturer (tilsammen opptil 40 nivåer).

Syntaks:

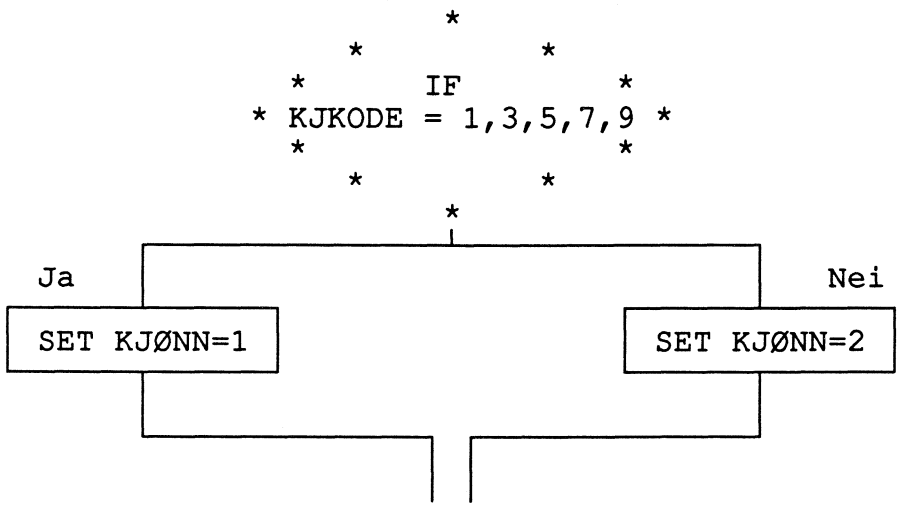
```
IF    (BETINGELSE)
:
ELSE
:
ENDIF
```

Se Logiske uttrykk for å se hvordan betingelser skal være.

Eksempel 1:

```
IF    (KJKODE = 1,3,5,7,9)
SET   KJØNN=1
ELSE
SET   KJØNN=2
ENDIF
```

Slik vil testen over bli utført:



Eksempel 2:

```
IF    (S1 NE 1,2)
MOVE  (36,10,E-),S3,ZZ ZZZ ZZZ
ELSE
MOVE  (36,10,X),'          :'
ENDIF
```

Når skal vi bruke IF og når skal vi bruke CASE?

Hvis det er mindre enn 3 utganger på en test, kan vi bruke IF.Er det 3 eller flere, bør vi bruke CASE. I eksempel 1 på forrige side er det 2 utganger; mann (1) eller kvinne (2). Det er 2 utganger også i eksempel 2.

Eksempel der CASE bør brukes:

```
CASE   (KJKODE = 1,3,5,7,9 AND SIVIL = 'U')
SET    KJSTATUS=1
CASE   (KJKODE = 1,3,5,7,9)
SET    KJSTATUS=2
CASE   (KJKODE = 0,2,4,6,8 AND SIVIL = 'U')
SET    KJSTATUS=3
CASE
SET    KJSTATUS=4
ENDCASE
```

I denne testen ser vi at det er 4 utganger, derfor bør CASE brukes.

Slik ville eksemplet over vært hvis vi brukte IF:

```
IF     (KJKODE = 1,3,5,7,9 AND SIVIL = 'U')
SET    KJSTATUS=1
ELSE
IF     (KJKODE = 1,3,5,7,9)
SET    KJSTATUS=2
ELSE
IF     (KJKODE = 0,2,4,6,8 AND SIVIL = 'U')
SET    KJSTATUS=3
ELSE
SET    KJSTATUS=4
ENDIF
ENDIF
ENDIF
```

Å bruke IF her ville bety mer skriving for oss, dessuten ville programmet vårt bli mindre oversiktlig.

Slik kunne vi også ha laget testen, men det lønner seg heller ikke fordi programmet må gå igjennom alle de 4 testene for hver record som leses (selv om bare den første slår til).

```
IF     (KJKODE = 1,3,5,7,9 AND SIVIL = 'U')
SET    KJSTATUS=1
ENDIF
IF     (KJKODE = 1,3,5,7,9 AND SIVIL NE 'U')
SET    KJSTATUS=2
ENDIF
IF     (KJKODE = 0,2,4,6,8 AND SIVIL = 'U')
SET    KJSTATUS=3
ENDIF
IF     (KJKODE = 0,2,4,6,8 AND SIVIL NE 'U')
SET    KJSTATUS=4
ENDIF
```

Instruksjonene mellom LOOP og ENDLOOP utføres så lenge betingelsen er oppfylt.

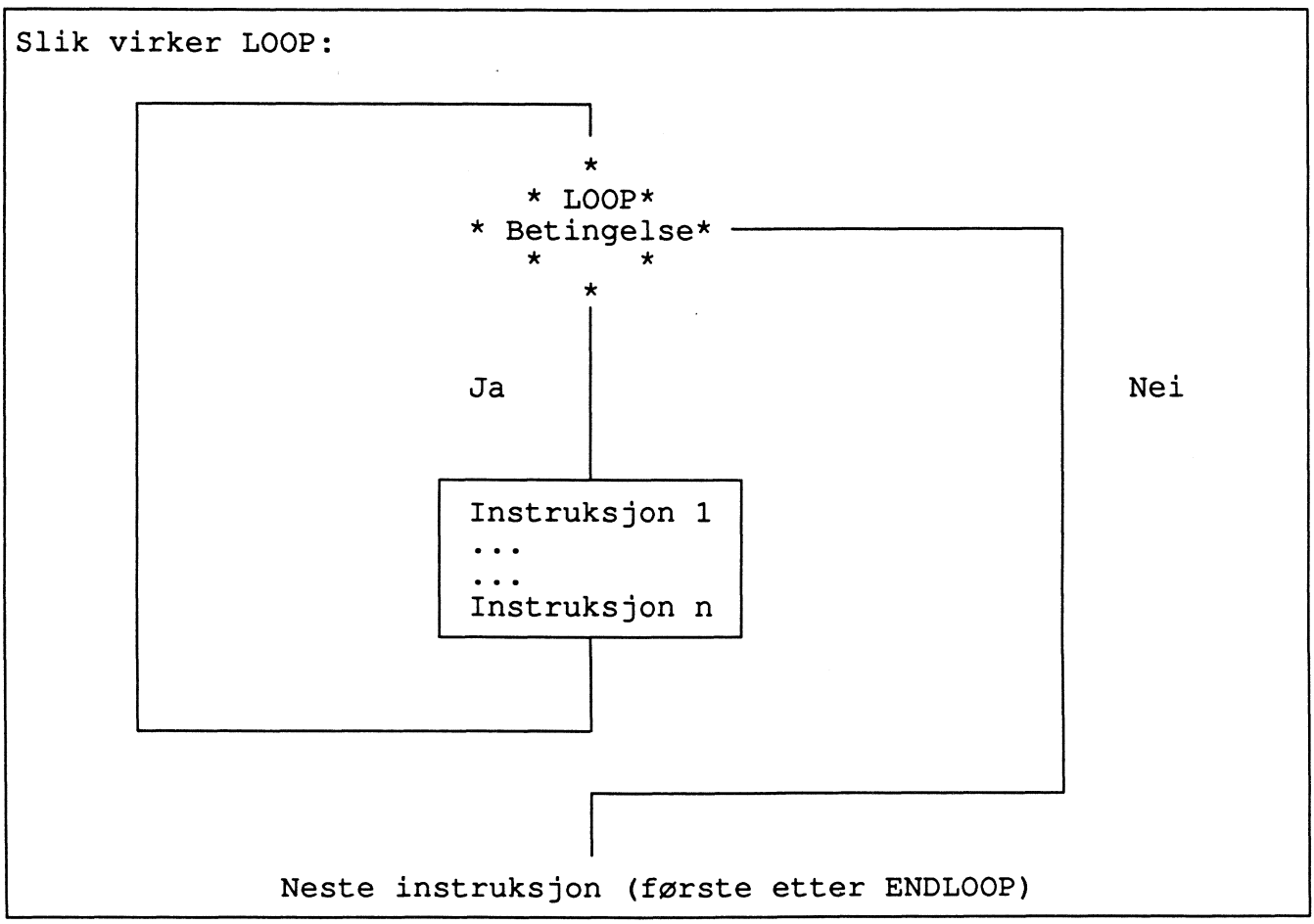
Syntaks 1:

```
LOOP  (BETINGELSE)
:
:
:
:
:
ENDLOOP
```

Eksempel:

```
SET    TELLER=1
LOOP   (TELLER <= 24)
SET    FELT(TELLER)=TELLER
SET    TELLER=TELLER+1
ENDLOOP
```

Instruksjonene i denne LOOP-en utføres 24 ganger.



Syntaks 2:

```
LOOP  TALLYX,STARTVERDI,SLUTTVERDI,KONSTANT
:
:
:
:
:
ENDLOOP
```

Forklaring:

TALLYX: TALLY0 - TALLY19. Dette er 20 systemdefinerte felt som er nullstilt ved start og ikke nullstilles ved innlesing av nye records. Disse kan vi f.eks bruke som teller til LOOP. Vi må bare huske på å sette inn verdier for det TALLY-feltet vi velger. Forskjellen på disse og vanlige work-felt er at vi ikke behøver å definere TALLYX-felt i definisjonsdelen av programmet, de blir definert av programmet selv om vi ikke har definert dem.

STARTVERDI: Startverdi for TALLYX

SLUTTVERDI: Sluttverdi for TALLYX

KONSTANT: Endring i TALLYX for hvert gjennomløp

Eksempel (gir samme resultat som eksemplet i Syntaks 1, men krever mindre skriving):

```
LOOP  TALLY1,1,24,1
SET   FELT(TALLY1)=TALLY1
ENDLOOP
```

Strukturer med IF/ELSE/ENDIF, CASE/ENDCASE og LOOP/ENDLOOP kan inneholde andre strukturer.

Eksempel:

```
IF
LOOP
:
ENDLOOP
:
ELSE
IF
:
ENDIF
:
ENDIF
```

SET foretar beregninger; +, -, * og / og flytter felt. Alle felt kan være indekserte (definert som en tabell (Array)). Ved binære regne-operasjoner må ikke verdien av noen felt være større enn ca. 2,1 * 10**9 (2**31 - 1). Ved pakket desimal er grensen 10**15 -1. Disse grensene gjelder også ved mellomresultater.

NB! Utregningene i SET utføres fra venstre mot høyre!

Syntaks:

LABEL SET FELTNAVN(ROUNDED)=Parameter OPERATOR Parameter,SIGN/ABS

Forklaring:

FELTNAVN: Feltet resultatet av SET flyttes til.

ROUNDED: (Kan forkortes til R) gir avrunding av alle divisjoner.

Parameter: Kan være tally-felter, et felt i input-recorden, et arbeidsfelt eller en konstant.

OPERATOR: + - * /

SIGN/ABS: SIGN gjør at eventuelt fortegn bevares, mens ABS gir absolutte tall. Standardverdi: ABS.

Eksempel:

```
SET   A=5
SET   A=B*C-4,SIGN
SET   A=B*10/C
```

NB!
SET regner med heltallsaritmetikk, dvs. at tallene kuttes eller rundes av etter hver regneoperasjon (se eksempler under). Uttrykk inne i parenteser utføres før parentesene løses opp.

Eksempler: Hvis B=2 og C=4 blir

```
SET  A=B/C*10        gir A=0 (2/4 = 0.5 som kuttes til 0, 0*10 = 0)
SET  A(R)=B/C*10     gir A=10 (2/4 = 0.5 som rundes av til 1, 1*10 = 10)
SET  A=B*10/C        gir A=5 (2*10 = 20, 20/4 = 5)
SET  A=(B/C)*10      gir A=0 (2/4 = 0.5 som kuttes til 0, 0*10 = 0)
SET  A=B/(C*10)      gir A=0 (4*10 = 40, 2/40 =0.05 som kuttes til 0)
SET  A=B+C/10        gir A=0 (2+4 = 6, 6/10 = 0.6 som kuttes til 0)
SET  A(R)=B+C/10     gir A=1 (2+4 = 6, 6/10 = 0.6 som rundes av til 1)
SET  A=B+C/10+B      gir A=2 (2+4 = 6, 6/10 = 0.6 som kuttes til 0,
                              0 + 2 = 2)
SET  A(R)=B+C/10+B gir A=3 (2+4 = 6, 6/10 = 0.6 som rundes av til 1,
                              1 + 2 = 3)
SET  A=B/(C-4)       gir Abend U0111 (4-4 = 0, 2/0 er umulig!)
```

SOEG foretar omkoding ved tabelloppslag. Tabellen lager du selv i SOEG-instruksjonen (se under).

Syntaks:

```
SOEG  TILFELT=FRAFELT,(TILKODE1,(INTERVALL,...INTERVALL),
                       TILKODE2,(INTERVALL,...INTERVALL),
                       :
                       :
                       TILKODEn,(INTERVALL,...INTERVALL),
                       RESTKODE)
```

Eksempel:

```
SOEG  LANDOMR=LAND,(1,(100-199),      *  Europa
                    2,(200-399),      *  Afrika
                    3,(400-599),      *  Asia
                    4,(600-699),      *  Nord-Amerika
                    5,(700-799),      *  Sør-Amerika
                    6,(800-899),      *  Oseania
                    9)                *  Ellers
```

```
Hvis    LAND = 100-199
blir LANDOMR = 1,
hvis    LAND = 200-399
blir LANDOMR = 2,
hvis    LAND = 400-599
blir LANDOMR = 3,
hvis    LAND = 600-699
blir LANDOMR = 4,
hvis    LAND = 700-799
blir LANDOMR = 5,
hvis    LAND = 800-899
blir LANDOMR = 6,
hvis    LAND = Restgruppen
blir LANDOMR = 9
```

STOP brukes til å stoppe forbehandlingen av en record (uten at opptelling er foretatt, hvis du ikke har brukt TAEL-funksjonen). Programmet vil så starte med forbehandling av neste record.

Den vanligste bruken av STOP er i forbindelse med TAEL-instruksjonen. Når vi bruker TAEL må STOP være med (se TAEL).

STOP brukes også til å avslutte instruksjoner i SETUP. Her brukes den for å si at SETUP er ferdig, slik at programmet kan begynne å lese records fra innfilen og starte med forbehandling og opptelling.

I Tab1 må vi også bruke STOP til å avslutte IDSTART (se denne).

Når vi bruker SETUP, IDSTART og/eller TAEL i programmet vårt, betyr det at vi endrer litt på strukturen i programmet. De instruksjonene som er med i SETUP, blir utført før det leses en eneste record fra inputfilen til tabellprogrammet. Instruksjonene som er med i IDSTART utføres hver gang kombinasjonen av forspaltekriteriene får en ny verdi. Disse må avsluttes med STOP slik at det er mulig å komme tilbake til programmets egentlige struktur.

Når vi bruker TAEL, hopper vi til opptellingsfasen (et hopp for hver TAEL). Det betyr at vi bryter med strukturen i programmet. Ved å bruke STOP i forbindelse med TAEL avslutter vi forbehandlingen for den aktuelle record og går tilbake til programmets egentlige struktur uten å foreta noen ny opptelling (dvs. leser neste record).

Eksempler på STOP; se SETUP, IDSTART, TAEL og VEDLEGG IV.

TAEL gir mulighet for å telle opp verdier for en lest record flere ganger, en for hver TAEL. Det vil bli hoppet til opptellingsfasen en gang for hver TAEL vi har med. Etter at det er telt vil programmet hoppe til første instruksjon etter TAEL. Når TAEL brukes, må STOP være med (se eksemplet).

Eksempel:

```
KOMMNR   FIELD (40,4,X)       *   KOMMUNENUMMER
ALDER    FIELD (67,3,X)       *   ALDER
ALDGRP   WORK  (1,X),ZEROES   *   ALDERSGRUPPER 1 = 16-74, 2 = 25-66
MAIN     IF    (ALDER = 16-74)
         SET   ALDGRP=1
         TAEL
         ENDIF
         IF    (ALDER = 25-66)
         SET   ALDGRP=2
         TAEL
         ENDIF
         STOP
TAELOP   SØJLE S1
         FORSP ALDGRP,NOSUM
         FORSP KOMMNR,NOSUM
```

I dette eksemplet skal det telles opp data for 2 forskjellige aldersgrupper fordelt på kommunenummer. Som vi ser overlapper disse 2 aldersgruppene hverandre. Det betyr at data fra 1 record fra input-filen skal telles 2 ganger (når alder er fra og med 25 til og med 66 år). Da må vi bruke TAEL. Ved å bruke TAEL slik som i eksemplet over, vil følgende skje:

Hvis den første IF-testen slår til, vil feltet ALDGRP settes til 1. Deretter følger TAEL, det betyr at vi hopper til opptellingsfasen i programmet (SØJLE). Her vil opptelling skje, og når den er ferdig vil vi fortsette med første instruksjon etter TAEL. Det er en ny test, og hvis den slår til, vil ALDGRP settes til 2, og så vil en ny TAEL utføres, dvs. at vi hopper til opptellingsfasen igjen og teller en gang til.

Ved å bruke TAEL bryter vi med strukturen i programmet, vi hopper fram og tilbake.

Se også Vedlegg IV for et eksempel på en vanlig bruk av TAEL-instruksjonen.

Programmet avsluttes når testinstruksjonene er gjennomført N ganger. Det betyr i praksis at vi kjører programmet vårt på en testfile som består av de N første records av input-filen som behandles av programmet. Det betyr at kjøringen avsluttes etter behandlingen av N selekterte records.

Det lønner seg å bruke TEST når vi skal teste ut nye programmer. Som oftest må vi kjøre programmet flere ganger før vi har fått tabellen til å se ut som vi vil, og inntil da bør vi ha med TEST. Når vi er fornøyd med programmet vårt og skal kjøre det på hele input-filen må vi huske på å fjerne TEST-instruksjonen.

Hvis dataene ligger på tape, bør du sørge for å få lagt noen testdata ut på disk når du lager og tester programmer. Hvis du kjører testkjøringer fra tape, må du huske på at for hver gang du sender en jobb til eksekvering (SUBMIT) må driftskontoret montere tapen. Det skal ikke mange feilkjøringer med tape til før du får driftskontoret på nakken. Derfor bør du ha en liten testfil på disk, da slipper du å plage driftskontoret unødig, og dessuten vil eksekveringen gå fortere.

Syntaks:

TEST N

Eksempel:

```
          START TYPE=B
FNR       FIELD (1,11,X)
KJKODE    FIELD (9,1,X)
SIVSTAT   FIELD (15,1,X)
KOMMUNE   FIELD (235,4,X)
KJSTAT    WORK  (1,X)
AAR       WORK  (4,X),PARM(1,4)
SELECT    INPUT (SIVSTAT = 1,2)
          TEST  200
          CASE  (KJKODE = 1,3,5,7,9 AND SIVSTAT = 1)
          SET   KJSTAT=1
          CASE  (KJKODE = 1,3,5,7,9)
          SET   KJSTAT=2
          CASE  (KJKODE = 0,2,4,6,8 AND SIVSTAT = 1)
          SET   KJSTAT=3
          CASE
          SET   KJSTAT=4
          ENDCASE
          SØJLE S1
          FORSP KOMMUNE,NOSUM
          FORSP KJSTAT,NOSUM
TABELL    FILE  BLKSIZE=17292,LRECL=11
          MOVE  (1,4,X),KOMMUNE
          MOVE  (5,1,X),KJSTAT
          MOVE  (6,6,X),S1
```

TEST bør komme som første instruksjon i forbehandlingen (I dette tilfelle mellom SELECT og CASE). Her vil eksekveringen avsluttes når programmet har behandlet 200 records som har SIVSTAT lik 1 eller 2.

11.8. OPPTELLING I SØJLER

Opptellingen skjer litt forenklet ved at det dannes en tallmatrise i storage (det indre lager). Denne matrisen omfatter forspalten og alle søjlene som er definert. For hver ny record som leses inn til programmet vil det bli talt i denne matrisen. Hvis verdien av forspaltekriteriene er ny, vil matrisen bli utvidet, hvis verdien har forekommet før, vil denne linjen i matrisen bli addert opp.

Følgende instruksjoner kan brukes i opptellingen:

SØJLE
SGRP - SGEND

Hvis vi i START-instruksjonen har definert TYPE=B, kan vi definere 1-1024 kolonner (søjler) der hver kolonne kan ha max 9 sifre + fortegn. Ved TYPE=P kan vi ha 1-512 kolonner med max 15 sifre + fortegn i hver kolonne.

På grunnlag av det høyeste søjle-nummer vi har definert i programmet vårt, blir det allokert plass i det interne minnet (storage). Det betyr at hvis vi i et program har definert de 3 søjlene S1, S100 og S200, vil det bli satt av plass til 200 søjler i intrenminnet. Derfor lønner det seg å definere søjlene fra S1 og så øke med 1 for hver ny søjle som skal defineres.

I opptellingen kan vi telle opp en konstant verdi eller vi kan telle opp innholdet av et felt (definert med FIELD el. WORK). Innholdet av et felt som skal telles opp må være numerisk, hvis ikke vil vi få ABEND (ABnormal ENDing) med kode U0107 (Se feilmeldinger).

SØJLE brukes til å definere hva som skal telles opp og eventuelle betingelser som skal gjelde for opptellingen.

Syntaks:

(1) SØJLE SN,(BETINGELSE),VERDI/FELTNAVN

(2) SØJLE SN1:SN2,(BETINGELSE),VERDI/FELTNAVN

(3) SØJLE SN1:SN2,,FELTNAVN(INDEKS)

(4) SØJLE SN1:SN2,,TALLYX,VERDI/FELTNAVN

(1) Brukes for å få telt opp i en SØJLE. Det telles i søjlen hvis betingelsen er sann (eller utelatt), med angitt verdi. Utelates VERDI eller FELTNAVN telles det med 1.

Forklaring:

SN: Søjle N, der N er tall fra 1 til 1024

VERDI: Et tall.

FELTNAVN: Et felt fra inputfilen eller et work-felt. Feltets verdi telles.

Eksempel 1:

```
SØJLE S1,(KJØNN = 1),PERSONER
SØJLE S2,,FISK
SØJLE S3
SØJLE S4,,1
SØJLE S5,(KJØNN = 1)
```

I søjle S1 telles det antallet i feltet personer hvis kjønn = 1, i S2 telles det verdien til feltet fisk, i S3 telles det med 1 for hver record, S4 er identisk med S3 og i S5 telles 1 for hver record der kjønn = 1

Eksempel 2:

```
SØJLE S1,(KOMMUNE = 0101)
SØJLE S2,(KOMMUNE = 0102)
SØJLE S3,(KOMMUNE = 0211-0301,0402)
SØJLE S4,(KOMMUNE NE 0101,0102,0211-0301,0402)
```

Det telles 1 pr. record i søjle S1 hvis kommune = 0101, i S2 hvis kommune = 0102, i S3 hvis kommune = 0211-0301 eller 0402 og i S4 i alle andre tilfelle.

Eksempel 3:

```
SØJLE S1,(KOMMUNE < 0801),KATTER
SØJLE S2,(KOMMUNE = 0801-1299),KATTER
SØJLE S3,(KOMMUNE = 1401-1599),KATTER
SØJLE S4,(KOMMUNE > 1599),KATTER
```

Det telles antall katter for kommunene 0101-0799 i søjle S1, for kommunene 0801-1299 i S2, for kommunene 1401-1599 i S3 og for resten av kommunene i S4

(2) Brukes til å få telt i en eller flere søjler, avhengig av innholdet av et felt.

Syntaks:

SN1: Den første søjlen vi skal telle opp

SN2: Den siste søjlen vi skal telle opp. Kolon (:) mellom søjlene forteller at vi skal telle opp søjlene fra og SN1 til og med SN2.

BETINGELSE: (FELTNAVN = INTERVALL,INTERVALL:INTERVALL,..INTERVALL:..)

FELTNAVN: Det feltet vi skal teste mot for å se hvilken søjle vi skal telle opp i.

INTERVALL: De verdier vi skal sjekke FELT mot. Det telles i den første søjlen testen slår for. Kolon (:) brukes for å skille mellom søjler. Hvert kolon angir et skille.

Eksempel 1:

```
SØJLE S1:S4,(KOMMUNE = 0101:0102:0211-0301,0402:)
```

NB!
Vær klar over at SØJLE-instruksjonen med denne syntaksen fungerer som en CASE-test. Det betyr at når det er blitt telt i en av søjlene i instruksjonen, vil programmet hoppe til neste instruksjon i programmet uten å sjekke om betingelsen er oppfylt for flere søjler enn den ene.

Det telles i søjle S1 hvis kommune = 0101, S2 hvis kommune = 0102, S3 hvis kommune er fra og med 0211 til og med 0301 og 0402 og i S4 i alle andre tilfelle. Gir samme resultat som eksempel 2 for syntax (1) og bør derfor brukes isteden.

Eksempel 2:

```
SØJLE S1:S4,(KOMMUNE < 0801:1401:1599:)
```

Dette eksemplet gir samme resultat som eksempel 3 fra syntax (1), men det gir mye mindre skriving.

Eksempel 3:

Hvordan man **IKKE** skal gjøre; samme betingelse i 2 søjler.

```
SØJLE S1:S5,(ALDER = 0-24:25-66:67-74:25-74:80-140)
```

Med denne syntaksen av søjle-instruksjonen vil det altså utføres en test tilsvarende en CASE-test, der det testes helt til en betingelse har slått til for deretter å hoppe til neste instruksjon. Det betyr at det i eksemplet over ikke vil bli telt opp noe i søjle 4 fordi betingelsen for å telle opp i denne også gjelder for søjle 2 eller 3.

Slik må eksempel 3 gjøres for at det skal bli riktig:

```
SØJLE S1:S3,(ALDER = 0-24:25-66:67-74)
SØJLE S4,(ALDER = 25-74)
SØJLE S5,(ALDER = 80-140)
```

(3) Brukes til å få telt i en serie av søjler. Feltene som telles opp må være definert som indekserte, se eksemplet.

Eksempel:

```
BELØP    FIELD (10,6,P)*10
         :
         :
         SØJLE S11:S20,,BELØP(1)

     BELØP(1) telles i SØJLE S11
     BELØP(2) telles i SØJLE S12
           :
           :
     BELØP(10) telles i SØJLE S20
```

(4) Brukes til å få telt i en av flere søjler avhengig av innholdet av et TALLYX-felt.

Eksempel:

```
        SØJLE S10:S22,TALLY8

   ===> Telling i SØJLE S10 hvis TALLY8=1
        Telling i SØJLE S11 hvis TALLY8=2
                            :
                            :
        Telling i SØJLE S22 hvis TALLY8=13
```

SGRP avgrenser en gruppe søjler og betingelser for å telle opp i disse

Syntaks:

(1)
SGRP (BETINGELSE),ALL
(2)
SGRP SANTAL,(BETINGELSE),ALL

(1) Brukes for å avgrense en gruppe søjler. ALL gjør at det telles i alle søjlene i SGRP hvis betingelsen er oppfylt. Hvis ALL utelates, forlates SGRP når det er telt i 1 SØJLE (som en CASE der vi ikke tester videre når en av testene har slått til).

Eksempel:

```
SGRP  (KJØNN = 1),ALL
SØJLE S1:S7,(ALDER = 0-15:16-19:20-24:25-49:50-66:67-74:)
SGEND
```

```
===> Hvis 'KJØNN = 1' telles det: i søjle S1 hvis alder er 0-15
                                  i søjle S2 hvis alder er 16-19
                                           :
                                  i søjle S7 hvis alder er over 74
```

Slik ville eksempel 1 se ut hvis vi ikke hadde brukt SGRP:

```
SØJLE S1,(KJØNN = 1 AND ALDER = 0-15)
SØJLE S2,(KJØNN = 1 AND ALDER = 16-19)
SØJLE S3,(KJØNN = 1 AND ALDER = 20-24)
SØJLE S4,(KJØNN = 1 AND ALDER = 25-49)
SØJLE S5,(KJØNN = 1 AND ALDER = 50-66)
SØJLE S6,(KJØNN = 1 AND ALDER = 67-74)
SØJLE S7,(KJØNN = 1 AND ALDER > 74)
```

Eksempel 2: Sgrp uten betingelse

```
SGRP
SØJLE S1,(FYLKE = 01,03 AND INNBYGG = 10000-50000)
SØJLE S2,(FYLKE = 04,07)
SØJLE S3,(FYLKE = 10,12)
SØJLE S4                          <=== Resten av kommunene
SGEND
```

Slik ville S4 se ut hvis vi ikke brukte SGRP:

```
SØJLE S4,(FYLKE NE 04,07,10,12 AND
         (FYLKE NE 01,03 OR INNBYGG NE 10000-50000))
```

Det er lettere å se at det i S4 dreier seg om resten av kommunene hvis vi bruker SGRP. Når vi bruker SGRP som i eksempel 2, lønner det seg å kommentere at siste SØJLE skal inneholde en rest.

(2) Brukes til å velge opptelling i en av flere søjlegrupper.

SANTAL angir antall søjler i hver søjlegruppe.
BETINGELSE angir hvilken gruppe det skal telles i.

Eksempel 3:

```
SGRP  7,(KJØNN = 1:2),ALL
SØJLE S1:S7,(ALDER = 0-15:16-19:20-24:25-49:50-66:67-74:)
SGEND
```

```
===> Hvis KJØNN = 1 telles det: i søjle S1 hvis alder er  0-15
                                i søjle S2 hvis alder er 16-19
                                        :
                               i søjle S7 hvis alder er over 74

     Hvis KJØNN = 2 telles det: i søjle S8 hvis alder er  0-15
                                i søjle S9 hvis alder er 16-19
                                        :
                                i søjle S14 hvis alder er over 74
```

Vi kan på samme måte som IF og CASE også neste SGRP i opptil 40 nivåer.

Eksempel 4:

```
SGRP  (KOMMUNE = 0301),ALL
SGRP  7,(KJØNN = 1:2),ALL
SØJLE S1:S7,(ALDER = 0-15:16-19:20-24:25-49:50-66:67-74:)
SGEND
SGEND
```

Dette er det samme som eksempel 3, men opptellingen vil bare skje hvis kommunenummeret er 0301

11.9. BESKRIVELSE AV FORSPALTEN

Til å beskrive forspalten vår bruker vi følgende instruksjoner:

TABEL
TOTAL
FORSP

I tabellen vår må vi ha minst et forspaltekriterium. Summen av TABEL- og FORSP-instruksjoner må ikke være over 13.

TABEL er et slags forspaltekriterium, men på et høyere nivå. Når vi bruker TABEL-instruksjonen, vil vi få ut en tabell for hver verdi feltet vi bruker i TABEL har.

Eksempel:

Vi skal lage næringstabeller; en tabell pr. fylke.

```
          START TYPE=P
FYLKE     FIELD (1,2,X)
NÆRING    FIELD (6,2,X)
OMSETN    FIELD (18,6,X)
FYLKNAVN  WORK  (20,X)
SELECT    INPUT (FYLKE = 01,02)
          SØJLE S1,,OMSETN
          TABEL FYLKE,TXTFIL=FYLKTXT,TXTKEY=(1,2,X),TXT=(4,20),
                WORK=FYLKNAVN
          TOTAL TXT='Hele fylket'
          FORSP NÆRING,TXTFIL=NAERTXT,TXTKEY=(1,2,X),TXT=(1,27),
                PRINT=1
TABELL    FILE  PRINT,MAXLIN=80,OVERFLOW
          HDR   1,1,'Tabell 1   Omsetning i forskjellige næringer.'
          HDR   1,47,FYLKNAVN
          MOVE  (30,10,Z-),S1
```

Dette programmet vil gi en tabell pr. fylke med næring i forspalten:

```
Tabell 1   Omsetning i forskjellige næringer. Østfold

Hele fylket                        300000

61 Engros- og agenturhandel        100000
62 Detaljhandel                    200000


Tabell 1   Omsetning i forskjellige næringer. Akershus

Hele fylket                       1000000

61 Engros- og agenturhandel        501000
62 Detaljhandel                    499000
```

Som vi ser over, har vi mulighet til å erstatte tallkoder med tekster. Vi har mange muligheter til å redigere forspalten. Disse angis med parametre i TABEL og/eller FORSP.

TABEL beskriver oppsplitting i tabeller. TABEL kan ikke brukes i underprogammet TAB1 (men i TAB1A og TAB1B).

Det kan brukes 1-9 tabellkriterier, men summen av FORSP- og TABEL-kriterier må ikke overstige 13.

Syntaks:

(1)

```
TABEL FELTNAVN,PAGE/PAGE1/CPAGE,SKIP=,PRINT=,TOTKEY=,
      SELECT=/SUPRESS=
```

(2)

```
TABEL FELTNAVN,TXTFIL=/MEMBER=,TXTKEY=,TXT=,PRINT=/WORK=,
      PAGE/PAGE1/CPAGE,NOSUM/CSUM,SKIP=,PRINTGRP=,
      TOTKEY=,SEPCHAR=,CONTXT=,SELECT=/SUPRESS=,FPAGE,
      NOLEAD,TYPE=ALL
```

Syntaks (1) Beskriver tabellnivå uten txt-fil

Syntaks (2) Beskriver tabellnivå med txt-fil

Forklaring:

FELTNAVN: Navnet på feltet som inneholder forspaltekriteriet (Kan ikke være et TALLY-felt).

TXTFIL=DDname angir DD-navn på txt-filen

MEMBER=Membernavn anvendes når txt-filen er member i et partisjonert datasett.

TXTKEY=(START,LENGDE,TYPE) angir identens plassering og format i txt-recorden.

TXT=(START,LENGDE) angir hvor i txt-recorden teksten finnes.

TOTKEY=XXXX angir ident for txt-record med totaltekst (2) eller totalident (1).

TOTTXT='Totaltekst' angir tekst for total.

PRINT=START (1)
PRINT=(START,LENGDE) (2)

Angir startposisjon og lengde for tabelltekst i forspalten.
Hvis du ikke bruker tekstfil må både START og LENGDE tas med.

WORK=FELTNAVN brukes for å få flyttet txt/ident fra txt-filen inn i et work-felt, som kan brukes i overskriften (HDR).

SKIP=(M,N) angir spacing av første txt-linje. N kan ha verdier fra 1 til 9 (Standardverdi er 2)

PAGE angir at det skal skiftes side ved ny tabell.

PAGE1 gjør at det skiftes side og at sidenr nullstilles (Neste side blir side 1).

CPAGE

A) CPAGE angir at det skal skrives ut så mange totale tabeller som mulig på hver side.

B) CPAGE=XX gjør at det ikke foretas linjeskift hvis det er XX eller flere linjer igjen på siden.

CONTXT='Tekst' kan brukes sammen med 'CPAGE' til å angi en tekst som skrives ut som siste linje på en side, fordi det er nødvendig med sideskift midt i en gruppe.

SELECT=/SUPRESS= brukes til å selektere/undertrykke utskrift av spesielle linjer. Hvis 'SELECT/SUPRESS' brukes på høyere nivå, vil også utskrift av linjer på lavere nivå bli undertrykt.

SELECT=(INTERVALL,INTERVALL,...)

SUPRESS=(INTERVALL,INTERVALL,...)

Eksempel: SELECT=(2-8,12)

11.9.2. Total **TOTAL**

TOTAL modifiserer utskrift av total

Syntaks:

TOTAL TXT=,SKIP=,PRINT=,FPAGE

Forklaring:

TXT='Totaltekst' (Standardverdi = 'Total')
NB! Teksten kan ikke være lenger enn 50 posisjoner.

SKIP=N angir antall linjeskift før utskrift (N = 1-3, Standardverdi = 3)

PRINT=Printposisjon (Standardverdi = 1)

FPAGE angir for TAB1 og TAB1A at det skal skiftes side før utskrift av sumlinje på dette nivå (etter for TAB1B)

Eksempel:

```
TOTAL TXT='I alt .......',SKIP=2
```

FORSP beskriver forspaltekriterier og utseende av forspalten.

Syntaks:

(1) FORSP FELTNAVN,PRINT=,PAGE/CPAGE,NOSUM/CSUM,SKIP=,
PRINTGRP=,NEWLINE,SEPCHAR=,CONTXT=,
SELECT=/SUPRESS=,FPAGE,NOLEAD,SUMTXT

(2) FORSP FELTNAVN,TXTFIL=/MEMBER=,TXTKEY=,TXT=,
PRINT=/WORK=,PAGE/CPAGE,NOSUM/CSUM,SKIP=,
PRINTGRP=,NEWLINE,SEPCHAR=,CONTXT=,SUMTXT=,
SELECT=/SUPRESS=,FPAGE,NOLEAD,TYPE=ALL

(3) FORSP FELTNAVN,NOSUM/CSUM,SELECT/SUPRESS

Syntaks (1) Beskriver forspalte uten tekstfil

Syntaks (2) Beskriver forspalte med tekstfil

Syntaks (3) Brukes for sekvensiell utfil

Forklaring:

FELTNAVN: Navnet på feltet som inneholder forspaltekriteriet (Kan ikke være en TALLY).

TXTFIL=DDname angir DD-navn på tekstfilen

MEMBER=Membernavn anvendes når tekstfilen er member i et partisjonert datasett.

TXTKEY=(START,LENGDE,TYPE) angir identens plassering og format i tekstrecorden.

TXT=(START,LENGDE) angir hvor i txt-recorden teksten finnes.

PRINT=(START,LENGDE) (1)
PRINT=START (2)

Angir startposisjon og lengde for forspaltetekst.

SKIP=(M,N) angir hvor mange linjeskift (spacing) som skal utføres før utskrift av forspaltetekst. M og N kan ha verdier fra 1 til 9. Antall blanke linjer vi får blir 1 mindre enn verdiene vi setter.

For elementnivå angir:

M - Antall linjeskift før 1. linje etter brudd
N - Antall linjeskift før øvrige linjer
(Standardverdi: SKIP=(2,1))

For overordnet nivå angir:

M - Antall linjeskift før overskriftlinje (NEWLINE)
N - Antall linjeskift før første sumlinje
(Standardverdi: SKIP=(2,2)

PAGE angir at det skal skiftes side ved brudd i dette forspaltenivå.

CPAGE

A) Hvis vi ikke vil at sideskift skal komme midt i utskriften av linjer på et forspaltenivå, bruker vi CPAGE. Hvis det ikke er plass til alle disse linjene på siden, vil de bli overført til neste side.

B) CPAGE=XX gjør at det ikke foretas sideskift hvis det er XX eller flere linjer igjen på siden.

FPAGE angir at det skal skiftes side før utskrift av sumlinje på dette nivå (TAB1 og TAB1A) eller etter (TAB1B).

SUMTXT er den tekst som vil bli skrevet ut i sumlinje istedenfor feltnavn/tekst fra txt-fil (Gjelder TAB1 og TAB1A).

NOSUM angir at det ikke skal dannes sumlinje.

CSUM angir at den overordnede sumlinje bare dannes hvis det er forskjellige verdier av feltnavn på dette nivå.

PRINTGRP=N kan bare brukes på laveste forspaltenivå. Brukes for å gjøre utskriften lettere å lese. Linjene som skrives ut på laveste nivå i tabellen, vil bli delt opp i grupper på N (dette tallet velger du selv) linjer. Den første linjen i hver gruppe vil skrives ut etter det antall linjeskift som er angitt med M i SKIP, de øvrige som angitt ved N.

SEPCHAR='K' angir at det skal skrives ut en forspaltelinje som inneholder den angitte karakter ved brudd i det pågjeldende kriterium. SEPCHAR kan bare anvendes i forbindelse med NEWLINE.

CONTXT='Tekst' kan brukes sammen med CPAGE til å angi en tekst som skrives ut som siste linje på en side når det er nødvendig med sideskift midt i en gruppe.

NEWLINE angir at det skal skiftes linje etter utskrift av teksten på dette nivå, og denne teksten vil ikke bli skrevet ut på lavere nivå.

NOLEAD angir i forbindelse med NEWLINE at det ikke skal skrives ut ledende tekst, bare tekst for sumlinje (Gjelder ikke TAB1B).

SELECT=/SUPRESS= brukes til å selektere/undertrykke utskrift av spesielle linjer. Hvis SELECT/SUPRESS brukes på høyere nivå, vil også utskrift av linjer på lavere nivå bli undertrykt.

```
SELECT=(INTERVALL,INTERVALL,...)
SUPRESS=(INTERVALL,INTERVALL,...)
```

Eksempel: SELECT=(1-5,7)

TYPE=ALL kan bare brukes på laveste forspaltenivå. Det angir at utskrift skal styres av forspaltetekstfilen, dvs. at alle linjene i fospaltetekstfilen skal skrives ut, uavhengig om de får data eller ei. Hele forspaltetekstfilen vil bli skrevet for hver verdi eventuelle høyere forspaltenivåer har.

WORK=FELTNAVN. FELTNAVN er navnet på et group- eller work-felt med TYPE=X og lengde lik det som er angitt i TXT=. Før utskriving og beregning av linjer på dette nivå, flyttes innholdet fra TXT= til FELTNAVN, som det kan refereres til av HDR, MOVE og SREGN.

Eksempler på bruk av en del av parametrene til FORSP-instruksjonen:

(Dette er programmet med JCL og inn- og utfiler. Det er bare parametrene i FORSP og om det er TAB1, TAB1A eller TAB1B som kjøres, som endres)

```
//0414KRLB JOB 8019,'Forsp-eksempel',MSGCLASS=X,CLASS=A,NOTIFY=0414KRL
//FORSP    EXEC TAB1B
//INPUT    DD  *
A1B10024
A1B20033
A2B10009
//TABELL   DD  SYSOUT=*
//ATEKST   DD  *
A1Norge
A2Sverige
//BTEKST   DD  *
B1Katter
B2Rever
B3Elger
B4Rådyr
B5Jerver
B6Gauper
//SYSIN    DD  *
         START TYPE=P
A        FIELD (1,2,X)              *   A1
B        FIELD (3,2,X)              *   B1,B2
TALL     FIELD (5,4,X)              *   24,33,09
TAELOP   SØJLE S1,,TALL
         FORSP A,PRINT=1
         FORSP B,PRINT=4
TABELL   FILE  PRINT,OVERFLOW,MAXLIN=46
         HDR   1,1,'Test av FORSP-instruksjonen'
         MOVE  (14,4,Z-),S1
```

Programmet over vil lage denne tabellen:

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

A1 B1          24
A1 B2          33

A1             57

A2 B1           9

A2              9


TOTAL          66
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

TOTAL          66

A1             57

A1 B1          24
A1 B2          33

A2              9

A2 B1           9
```

Under følger eksempler på forspalten til det samme programmet som på forrige side, bortsett fra at vi endrer litt på forspalten. Først ser vi hvordan forspalten ser ut, deretter hvordan tabellen vil se ut.

```
        FORSP A,PRINT=1,NOSUM
        FORSP B,PRINT=4
```

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

A1 B1           24
A1 B2           33

A1              57

A2 B1            9

A2               9
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

A1              57

A1 B1           24
A1 B2           33

A2               9

A2 B1            9
```

```
         FORSP A,PRINT=1,NOSUM
         FORSP B,PRINT=4,NOSUM
```

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

A1 B1           24
A1 B2           33

A2 B1            9
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

A1 B1           24
A1 B2           33

A2 B1            9
```

Som vi ser over, blir resultatet det samme med TAB1, TAB1A og TAB1B. Da bruker vi TAB1 hvis input-filen er sortert på A og B, og TAB1A hvis den ikke er sortert. Dette fordi TAB1 er mest effektivt av Tab-programmene, deretter følger TAB1A og så TAB1B.

```
         FORSP A,PRINT=1
         FORSP B,PRINT=4,CSUM
```

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

A1 B1           24
A1 B2           33

A1              57

A2 B1            9


TOTAL           66
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

TOTAL           66

A1              57

A1 B1           24
A1 B2           33

A2 B1            9
```

```
FORSP A,PRINT=1,NEWLINE
FORSP B,PRINT=4
```

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

A1

   B1          24
   B2          33

A1             57

A2

   B1           9

A2              9


TOTAL          66
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

TOTAL          66

A1             57

   B1          24
   B2          33

A2              9

   B1           9
```

```
FORSP A,PRINT=1,NEWLINE,NOLEAD
FORSP B,PRINT=4
```

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

   B1          24
   B2          33

A1             57

   B1           9

A2              9


TOTAL          66
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

TOTAL          66

A1             57

   B1          24
   B2          33

A2              9

   B1           9
```

```
FORSP A,PRINT=1,NEWLINE,SUMTXT='I ALT'
FORSP B,PRINT=4
```

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

A1

   B1          24
   B2          33

I ALT          57

A2

   B1           9

I ALT           9


TOTAL          66
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

TOTAL          66

A1             57

   B1          24
   B2          33

A2              9

   B1           9
```

```
FORSP A,PRINT=1,NEWLINE
FORSP B,PRINT=4,CSUM
```

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

A1

   B1          24
   B2          33

A1             57

A2

   B1           9


TOTAL          66
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

TOTAL          66

A1             57

   B1          24
   B2          33

A2

   B1           9
```

```
FORSP A,PRINT=1,NEWLINE,SUMTXT='I ALT'
FORSP B,PRINT=4,CSUM
```

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

A1

   B1          24
   B2          33

I ALT          57

A2

   B1           9


TOTAL          66
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

TOTAL          66

A1             57

   B1          24
   B2          33

A2

   B1           9
```

Forspalten med txt-fil:

```
FORSP A,TXTFIL=ATEKST,TXTKEY=(1,2,X),TXT=(3,10),PRINT=1
FORSP B,TXTFIL=BTEKST,TXTKEY=(1,2,X),TXT=(3,10),PRINT=4
```

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

NorKatter      24
NorRever       33

Nor            57

SveKatter       9

Sve             9


TOTAL          66
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

TOTAL          66

Norge          57

NorKatter      24
NorRever       33

Sverige         9

SveKatter       9
```

```
      FORSP A,TXTFIL=ATEKST,TXTKEY=(1,2,X),TXT=(3,10),PRINT=1,
            NEWLINE
      FORSP B,TXTFIL=BTEKST,TXTKEY=(1,2,X),TXT=(3,10),PRINT=4
```

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

Norge

   Katter      24
   Rever       33

Norge          57

Sverige

   Katter       9

Sverige         9


TOTAL          66
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

TOTAL          66

Norge          57

   Katter      24
   Rever       33

Sverige         9

   Katter       9
```

```
      FORSP A,TXTFIL=ATEKST,TXTKEY=(1,2,X),TXT=(3,10),PRINT=1,
            NEWLINE
      FORSP B,TXTFIL=BTEKST,TXTKEY=(1,2,X),TXT=(3,10),PRINT=4,
                  TYPE=ALL
```

```
------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

Norge

   Katter      24
   Rever       33
   Elger        -
   Rådyr        -
   Jerver       -
   Gauper       -

Norge          57

Sverige

   Katter       9
   Rever        -
   Elger        -
   Rådyr        -
   Jerver       -
   Gauper       -

Sverige         9


TOTAL          66
```

```
---------- TAB1B ----------

Test av FORSP-instruksjonen

TOTAL          66

Norge          57

   Katter      24
   Rever       33
   Elger        -
   Rådyr        -
   Jerver       -
   Gauper       -

Sverige         9

   Katter       9
   Rever        -
   Elger        -
   Rådyr        -
   Jerver       -
   Gauper       -
```

```
FORSP A,TXTFIL=ATEKST,TXTKEY=(1,2,X),TXT=(3,10),PRINT=1,
      NEWLINE
FORSP B,TXTFIL=BTEKST,TXTKEY=(1,2,X),TXT=(3,10),PRINT=4,
      TYPE=ALL,PRINTGRP=2
```

------ TAB1 og TAB1A ------

Test av FORSP-instruksjonen

Norge	
Katter	24
Rever	33
Elger	-
Rådyr	-
Jerver	-
Gauper	-
Norge	57
Sverige	
Katter	9
Rever	-
Elger	-
Rådyr	-
Jerver	-
Gauper	-
Sverige	9
TOTAL	66

---------- TAB1B ----------

Test av FORSP-instruksjonen

TOTAL	66
Norge	57
Katter	24
Rever	33
Elger	-
Rådyr	-
Jerver	-
Gauper	-
Sverige	9
Katter	9
Rever	-
Elger	-
Rådyr	-
Jerver	-
Gauper	-

11.10. BEREGNING AV SØJLER

Under beregning av søjler kan følgende instruksjoner brukes:

ABEND
CALL
CANCEL
CASE - ENDCASE
DOPAGE
FIND
GETPST
HOP
IF - ELSE - ENDIF
LOOP
PRCT
SET
SOEG
SREGN

Når vi skal foreta beregning på søjler, bør vi huske på forskjellen mellom TAB1 og TAB1A/TAB1B. I TAB1 dannes og skrives den enkelte linje for hvert brudd i forspaltekriteriene. TAB1A og TAB1B kjøres i 2 faser; i første fase leses inputfilen og forbehandling og opptelling (fordeling i linjer og kolonner, dvs. lage en matrise) foretas. I andre fase skjer beregning på matrisen (SREGN) og redigering og utskrift utføres.

Hvis vi i SREGN skal regne med arbeidsfelter, vil disse i TAB1A og TAB1B inneholde verdien arbeidsfeltene fikk ved behandlingen av den siste recorden fra input, dvs. at et arbeidsfelt i SREGN alltid vil inneholde en konstant. I TAB1 vil innholdet av arbeidsfeltet ha forskjellig verdi for hvert brudd i forspaltekriteriene (jfr. flytdiagrammene).

Under SREGN kan det tenkes at vi vil behandle elementærlinjer (det lavestenivået) og sumnivåer (overliggende nivåer) på forskjellig måte. Da kan vi benytte oss av det systemdefinerte feltet TALLY0.

Tally0

TALLY0 er et systemdefinert felt som viser hvilket nivå som behandles. Dette hjelper oss hvis vi skal behandle forskjellige nivåer på forskjellig måte. Dette feltet gis verdi før eventuell beregning av søjler og vi kan derfor teste på dette feltet under SREGN og ved utskrift.

Verdier for TALLY0:

0 ===> Elementærnivå (laveste nivå)

1 ===> 1. sumnivå

2 ===> 2. sumnivå

osv.

SREGN brukes til å foreta beregning på søjler. Vi kan også regne ut nye søjler. Utregninger foregår fra høyre mot venstre, med unntak av divisjoner som utføres tilslutt.

NB! Divisjoner i SREGN utføres til slutt! Dette i motsetning til SET, da regnes det fra venstre mot høyre uansett.

Syntaks 1:

SREGN SX=Parameter OPERATOR Parameter....

Eksempel:

```
SREGN S7=S1+S2+S3+S4+S5+S6  *  S7 = Sum av S1 til S6
SREGN S7=S1:S6              *  S7 = Sum av S1 til S6
SREGN S8=S7*100/S1:S6       *  S1:S6 Beregnes før divisjonen!
SREGN S1=S1-(S2:S5)         *  S1 = S1 - (Sum av S2 til S5)
```

Som vi ser (S7=S1:S6) fungerer : som summeringstegn for søjler. Du angir Søjlenummeret til den første og den siste søjlen du skal summere, adskilt med : og programmet summerer verdiene i søjlene fra og med den første til og med den siste søjlen. Det sparer oss for en del skriving.

Syntaks 2:

SREGN SX:SY=Parameter OPERATOR Parameter....

Eksempel:

```
SREGN S1:S5=SN*100/S6  *S1, S2, S3, S4 og S5 regnes som % av S6
```

Eksemplet over tilsvarer disse 5 SREGN-instruksjonene:

```
SREGN S1=S1*100/S6
SREGN S2=S2*100/S6
SREGN S3=S3*100/S6
SREGN S4=S4*100/S6
SREGN S5=S5*100/S6
```

NB!

Divisjoner i SREGN er alltid med avrunding til nærmeste heltall. Resultatet (også ved mellomregning) er et heltall.

Vi kan bruke et nivå med parenteser i SREGN.

PRCT gjør kolonner med absolutte tall om til kolonner med prosent av totalen til en fritt valgt kolonne. Hvis søjle-verdien er et negativt tall, vil prosenten også bli negativ. PRCT kan ikke brukes i underprogrammet TAB1 (men i TAB1A og TAB1B).

Syntaks:

PRCT SX:SY/SZ,DEC=N

Forklaring:

SX:SY Søjlene prosentene skal regnes på.

SZ Søjlen prosenten skal regnes av. Prosentene regnes ut fra totalen til denne søjlen. Hvis vi skriver SN blir prosentene regnet ut av søjlens egne totalsum.

DEC= Antall desimaler

Eksempler:

```
        PRCT  S1:S6/S7,DEC=3
        PRCT  S1:S6/SN,DEC=1
        PRCT  S15/S8,DEC=2
        PRCT  S10/SN,DEC=3
```

I det første eksemplet blir SØJLE S1 regnet ut som verdien til S1 i prosent av totalen til S7. S2 til S6 regnes ut på tilsvarende måte som S1. Det andre eksemplet viser hvordan vi regner ut prosent av søjlens total for S1 til S6. Eksempel 3 og 4 viser hvordan vi lager prosent av en søjle ut fra henholdsvis en annen søjle (eks. 3) og søjlens egne total (eks. 4).

Resultatet av prosentberegningen av en søjle skrives ut til den samme søjlen. Det betyr at hvis vi skal ha både absolutte tall og prosenttall i tabellen vår, må vi lage de søjlene vi skal lage prosenttall av 2 ganger (i 2 forskjellige søjler) før vi regner ut prosentene (ellers mister vi de absolutte tallene).

Eksempel:

```
          :
          SØJLE S2:S6,,FOLK(1)
          SØJLE S7:S11,,FOLK(1)
          FORSP KOMMUNE,PRINT=1
          SREGN S1=S2:S6
          PRCT  S7:S11/S1,DEC=2
TABELL    FILE  PRINT,OVERFLOW
          HDR   (1-1),(1-72)
TABELL 3. BEFOLKNING. ABSOLUTTE TALL OG PROSENT
          MOVE  (10,10,E-)*6,S1,ZZ ZZZ ZZZ
          MOVE  (70,10,E-)*5,S7,ZZZ ZZ9,99
```

11.10.3. Cancel **CANCEL**

CANCEL brukes for å undertrykke utskriften av en linje. Verdiene til linjen vil telle med i totalen. Hvis CANCEL brukes på et annet nivå enn det laveste, vil linjene som ligger på nivåer under den linjen som undertrykkes også undertrykkes (hvis vi tester på verdien av TALLY0, vil bare de linjer på det nivået som TALLY0 indikerer bli undertrykt).

Eksempel 1. Linjer med varenummer 2914101 eller 2914350 vil ikke bli skrevet ut, men verdiene deres vil være med i totalsummen.

```
FORSP VARENR,PRINT=1
IF    (VARENR = 2914101,2914350)
CANCEL
ENDIF
```

Eksempel 2. Linjer på laveste nivå (TALLY0 = 0) der verdien (S1) er mindre enn 100000 vil ikke bli skrevet ut, men verdiene vil være med i totalen.

```
SØJLE S1,,VERDI
FORSP VARENR,PRINT=1
IF    (S1 < 100000 AND TALLY0 = 0)
CANCEL
ENDIF
```

11.10.4. Dopage **DOPAGE**

DOPAGE brukes for å få sideskift på bestemte steder

Eksempel:

```
FORSP FYLKE,TXTFIL=FYLKTXT,TXTKEY=(1,2,X),TXT=(1,26),PRINT=1,
      NEWLINE
FORSP KOMMNR,TXTFIL=KOMMTXT,TXTKEY=(1,4,X),TXT=(1,26),PRINT=1
IF    (KOMMNR = 0826,1114,1560,1743 OR
      (FYLKE = 3,6,14,19 AND TALLY0 = 1))
DOPAGE
ENDIF
```

Dette eksemplet vil gi sideskift før utskrift av fylke 3, 6, 14 og 19 og kommunenummer 0826, 1114, 1560 og 1743.

11.10.5. Getpst **GETPST**

GETPST gjør at det systemdefinerte feltet TALLY19 vil inneholde antall resterende linjer på siden.

Eksempel:

```
FORSP FYLKE,TXTFIL=FYLKTXT,TXTKEY=(1,2,X),TXT=(1,26),PRINT=1,
      NEWLINE
FORSP KOMMNR,TXTFIL=KOMMTXT,TXTKEY=(1,4,X),TXT=(1,26),PRINT=1
GETPST
IF    (TALLY0 = 1 AND TALLY19 < 5)
DOPAGE
ENDIF
```

Her vil det foretas sideskift hvis det er mindre enn 5 linjer igjen på siden når fylkessummene skal skrives ut.

11.11. UTSKRIFT

Vi kan skrive ut tabellen vår til to forskjellige typer filer; tabellfil (ferdig tabell med overskrifter og tekster legges ut på en fil, klar til utskriving) og sekvensiell fil (tallene i tabellen legges ut på en vanlig datafil).

Ved sekvensiell utskrift kan vi bare skrive til 1 fil, ved tabellfiler kan vi ha opp til 5 utfiler.

I utskriftsfasen kan vi bruke disse instruksjonene:

FILE
HDR
MOVE

CASE - ENDCASE
IF - ELSE - ENDIF
LIN

[i forbindelse med MOVE]

HDR må være med når vi skriver til tabellfil, men skal ikke være med når vi skriver til sekvensiell fil. Vi velger om vi skal skrive tabellen vår til en tabellfil eller sekvensiell fil i FILE-instruksjonen (se denne).

Vi kan ikke bruke nestede IF og/eller CASE i forbindelse med MOVE; det betyr at en slik test må avsluttes før vi kan gjøre en ny (vi kan ha 1 CASE utenfor 1 IF, men det er alt).

Forskjellen på å skrive til en tabellfil og en sekvensiell fil:

Tabellfil:

```
          START TYPE=P
KOMMUNE   FIELD (25,4,X)
FYLKE     FIELD (25,2,X)
FOLK      FIELD (30,6,X)*10
          SØJLE S1:S10,,FOLK(1)
          FORSP FYLKE,PRINT=1
TABELL    FILE  PRINT,OVERFLOW
          HDR   1,1,'TABELL 1. BEFOLKNING I FYLKENE'
          MOVE  (10,6,E-)*10,S1,ZZ ZZ9
```

Sekvensiell fil (5 første linjer er like):

```
          FORSP FYLKE,NOSUM
MATRISE   FILE  LRECL=62,BLKSIZE=2542
          MOVE  (1,2,X),FYLKE
          MOVE  (3,6,X)*10,S1
```

Når vi lager tabellfil, definerer vi i FORSP-instruksjonen hvor i tabellen forspaltekriteriene skal komme. Ved utskrift til sekvensiell fil må vi flytte ut forspaltekriteriene med MOVE-instruksjoner (vi angir ikke noe om utskrift i FORSP-instruksjonen).

FILE definerer outputfilen (tabellfilen).

Syntaks 1 (tabellfil):

Brukes for å beskrive tabellfil (max 5 pr. kjøring)

DDNAVN FILE PRINT,MAXLIN=NN,OVERFLOW,LRECL=NNN,SKIP1=N

Forklaring:

MAXLIN=NN Max antall linjer pr. side i tabellen som lages. Standardverdi er MAXLIN=60.

> Tips: Sett MAXLIN=46 til de vanlige printerne (8" arklengde) og MAXLIN=84 til laserskriveren (RMT15, A4-ark) hvis vi forminsker utskriften til 8 linjer pr. tomme (tabellbredden bør da være max 115 posisjoner).

OVERFLOW: Hvis vi tar med OVERFLOW, vil det bli skrevet ut stjerner (*) hvis tallet i en celle er større enn plassen som er avsatt til tall i cellen. Det lønner seg alltid å ha med OVERFLOW.

LRECL=NNN Recordlengde for utfilen. Denne kan du bruke når du skal ha tabellen din ut på en fil.

> Tips: Sett LRECL=130 hvis tabellen din skal skrives ut på laserskriveren RMT15. Da slipper du problemer med at skriveren ikke er noe glad i filer som er bredere.

SKIP1=N der N kan være 1, 2 eller 3. SKIP1 brukes til å angi hvor mange blanke linjer det skal være mellom overskriften og tabellen. Vær klar over at du får en blank linje mindre enn det tallet du angir (SKIP1 = 2 gir deg en linje mellom overskrift og tabell).

Syntaks 2 (sekvensiell fil):

Gjelder for utskrift til sekvensiell fil (Max 1 pr. kjøring)

DDNAVN FILE LRECL=NNN,BLKSIZE=NNNN,SUM

Forklaring:

LRECL=NNN	Recordlengde for utfilen
BLKSIZE=NNNN	Blokkstørrelse for utfilen
SUM	Angir at summer skal tas med i utfilen

HDR gir overskrift til en tabell.

Syntaks 1:

HDR (STARTLINJENR-SLUTTLINJENR),(STARTPOSISJON-SLUTTPOSISJON)

Når vi bruker syntaks 1, 'tegner' vi hele overskriften. Vi lager en ramme, og innenfor rammen kan vi skrive hva vi vil. Du må være klar over at denne overskriften blir akkurat slik som du 'tegner' den. Størrelsen på rammen (antall linjer og posisjoner) bestemmer du selv i HDR-instruksjonen.

NB! Rammen starter i posisjon 1 i første linje etter HDR.

Eksempel:

```
       HDR   (1-5),(1-50)
Tabell 2. Snille kursdeltakere fordelt etter kjønn
--------------------------------------------------

              Gutter              Jenter
          ----------------    ----------------
```

Syntaks 2:

HDR LINJENR,START/(START,LENGDE),
KONSTANT/FELTNAVN/SIDENR/DATO

Eksempel:

```
       HDR   1,1,'Tabell 1. Snille medhjelpere'
       HDR   3,64,DATO
       HDR   11,37,AAR
```

SIDENR og DATO er systemdefinerte felt som vi kan referere til i overskriften vår uten at de er definert i definisjonsdelen.

Tips:
Det lønner seg alltid å bruke syntaks 1, unntatt når vi skal skrive verdien av et felt ut i overskriften, da må vi bruke syntaks 2.

MOVE Flytter felt, søjler og konstanter ut til den enkelte tabellinje. Vi kan ikke ha nestede IF eller CASE-setninger i forbindelse med MOVE.

Syntaks:

MOVE (STARTVERDI,LENGDE,TYPE)*ANTALL,SØJLENR/FELTNAVN/KONSTANT/
SPACES/ZEROES/LOW-VALUES/HIGH-VALUES/P10,MASKE

Forklaring:

STARTPOSISJON: Startposisjon feltet får i tabellutskriften.

LENGDE: Lengden til feltet (evt. søjlen el. konstanten).

TYPE:
- X ===> Alfanumerisk felt (lengde max 4096)
- B ===> Binært felt (lengde 2 eller 4)
- P ===> Pakket desimal-felt (lengde max 16)

Bare ved utflytting av søjler og numeriske felt:
- Z ===> Nullundertrykkelse av foranstilte nuller
- Z- ===> Som Z, men hvis null (før evt. divisjon) skrives - i siste posisjon
- ZB ===> Som Z, men hvis null (før evt. divisjon) skrives blanke
- E ===> Feltet redigeres som angitt MASKE
- E- ===> Feltet redigeres som angitt MASKE, men null behandles som ved Z-
- EB ===> Som E-, men null behandles som ved ZB

ANTALL: Skriver ut flere søjler eller felt vha en MOVE-instruksjon (like mange som antall). Feltene må være definert som indekserte, se FIELD. Søjler blir behandlet som indekserte felt. Max antall felt eller søjler som kan skrives ut med 1 MOVE er 500.

P10: Lovlige verdier for P10: 10, 100, 1000, 10000 og 100000. Gjør at innholdet av feltet divideres med P10 og rundes av før utskriving.

MASKE: Brukes til å editere numeriske data. Masken skal inneholde like mange tegn som du skal flytte ut i MOVE-instruksjonen. Følgende tegn kan brukes:

- Z ===> Foranstilte nuller fjernes
- 9 ===> Tallene skrives ut uten nullundertrykkelse
- \+ ===> Fortegn skrives ut (+ må stå først eller sist i masken)
- \- ===> Negative fortegn skrives ut, ellers blank (- må stå først eller sist i masken)

Alle andre tegn (unntatt parenteser) kan brukes som fyllkarakterer (inkl. +, - og blank). Fyllkarakterer som skal skrives ut etter tallet vil bare bli skrevet ut hvis tallet er negativt.

SPACES/ZEROES/LOW-VALUES/HIGH-VALUES: Kan kun brukes når TYPE er X. Disse verdiene vil bli flyttet ut. Max LENGDE er da 256.

SIGN/ABS: Kan kun brukes når TYPE er X. ABS gir absolutte tall, SIGN medfører at eventuelt fortegn bevares.

Eksempler:

```
MOVE   (12,10,X),S1,SIGN                    <=== A
MOVE   (12,10,Z),S1                         <=== B
MOVE   (12,10,Z)*4,S1                       <=== C
MOVE   (12,10,Z-)*4,S1                      <=== D
MOVE   (12,10,E-)*4,S1,ZZZ ZZ9,99           <=== E
MOVE   (12,10,E-)*4,S1,-ZZ ZZZ,99           <=== F
MOVE   (12,10,E-)*4,S1,+ZZ ZZ9,99           <=== G
MOVE   (12,10,E-)*4,S1/1000,+ZZ ZZ9,99      <=== H
MOVE   (1,4,X),KOMMNR                       <=== I
MOVE   (40,2,X),' :'                        <=== J
```

Hvis søjle S1 har verdien -537, S2 er 553428, S3 er 0, S4 er 12 og KOMMNR er 0301, blir utskriften slik (bare resultatet av MOVE-instruksjonen for 1 linje i tabellen):

```
     ----+----1----+----2----+----3----+----4----+----5----+

A                 000000053P
B                       537
C                       537     553428          0         12
D                       537     553428          -         12
E                     5,37   5 534,28          -       0,12
F                    -5,37   5 534,28          -        ,12
G                    -5,37  +5 534,28          -      +0,12
H                    -0,01      +5,53          -       0,00
I    0301
J                                                :
     ----+----1----+----2----+----3----+----4----+----5----+
```

A og I brukes mest når vi skal skrive til sekvensielle filer.

Hvis vi i MOVE-instruksjonen bruker type E (f. eks. (11,5,E)), må vi ha med en maske som viser hvordan dataene skal skrives ut, se tabellen nedenfor.

Tall	Maske						
	Z ZZZ	ZZZ,9	ZZ9,9	-ZZ9,9	+ZZ9,9	ZZ9,9+	ZZZZX
+2377	2 377	237,7	237,7	237,7	+237,7	237,7+	2377
+0003	3	,3	0,3	0,3	+0,3	0,3+	3
+0014	14	1,4	1,4	1,4	+1,4	1,4+	14
-2377	2 377	237,7	237,7	-237,7	-237,7	237,7-	2377X
-0003	3	,3	0,3	-0,3	-0,3	0,3-	3X
-0014	14	1,4	1,4	-1,4	-1,4	1,4-	14X
0		,0	0,0	0,0	+0,0	0,0+	

LIN brukes for å få en ny linje innenfor samme verdi av forspalten. Vi får en ny linje for hver LIN vi tar med. LIN kan bare brukes når vi skal lage tabellfiler, ikke når vi skal skrive til sekvensiell fil. Når vi bruker LIN, vil forspalteteksten bare komme med på den første linjen.

Eksempel:

```
          START TYPE=P
AARGANG   FIELD (1,4,X)
FYLKE     FIELD (5,2,X)
KOMM3     FIELD (7,1,X)
BEFOLKN   FIELD (9,6,X)
AAR       WORK  (4,X),SPACES
BYLAND    WORK  (12,X),SPACES
          SET   AAR=AARGANG
          IF    (KOMM3 = 0)
          SET   BYLAND='BYKOMMUNER  '
          ELSE
          SET   BYLAND='LANDKOMMUNER'
          ENDIF
          SØJLE S1:S4,(BEFOLKN < 5000:10000:20000:),BEFOLKN
          SØJLE S5,,BEFOLKN
          SØJLE S6:S9,(BEFOLKN < 5000:10000:20000:)
          SØJLE S10
          SØJLE S11:S14,(BEFOLKN < 5000:10000:20000:),BEFOLKN
          SØJLE S15,,BEFOLKN
          TOTAL TXT='HELE LANDET',PRINT=1
          FORSP BYLAND,PRINT=1,SKIP=(2,2)
          SREGN S16=S1*100/S6
          SREGN S17=S2*100/S7
          SREGN S18=S3*100/S8
          SREGN S19=S4*100/S9
          SREGN S20=S5*100/S10
          PRCT  S11:S15/S5,DEC=2
TABELL    FILE  PRINT,MAXLIN=46,OVERFLOW
          HDR   1,30,'BEFOLKNINGSTABELL'
          HDR   (2-7),(16-87)
*********************************************************
* KOMMUNER ETTER ANTALL INNBYGGERE. VVVV                *
*********************************************************
*          *  5000-   *  10000-  *          *          *
* < 5000   *  9999    *  19999   * > 20000  * I ALT     *
*********************************************************
          HDR   4,66,AAR
          MOVE  (27,1,X),' '
          LIN
          MOVE  (1,26,X),'ANTALL KOMMUNER ..........'
          MOVE  (27,12,Z-)*5,S6
          LIN
          MOVE  (1,26,X),'GJ.SNITT ANT. INNB. ......'
          MOVE  (27,12,E-)*5,S16,Z ZZZ ZZ9,99
          LIN
          MOVE  (1,26,X),'PROSENT AV BEFOLKNINGEN ..'
          MOVE  (27,12,E-)*5,S11,Z ZZZ ZZ9,99
          LIN
          MOVE  (1,26,X),'INNBYGGERE TOTALT ........'
          MOVE  (27,12,Z-)*5,S1
```

12. TAB2

Tab2 adskiller seg fra Tab1 spesielt med hensyn til oppbyggingen av forspalten.

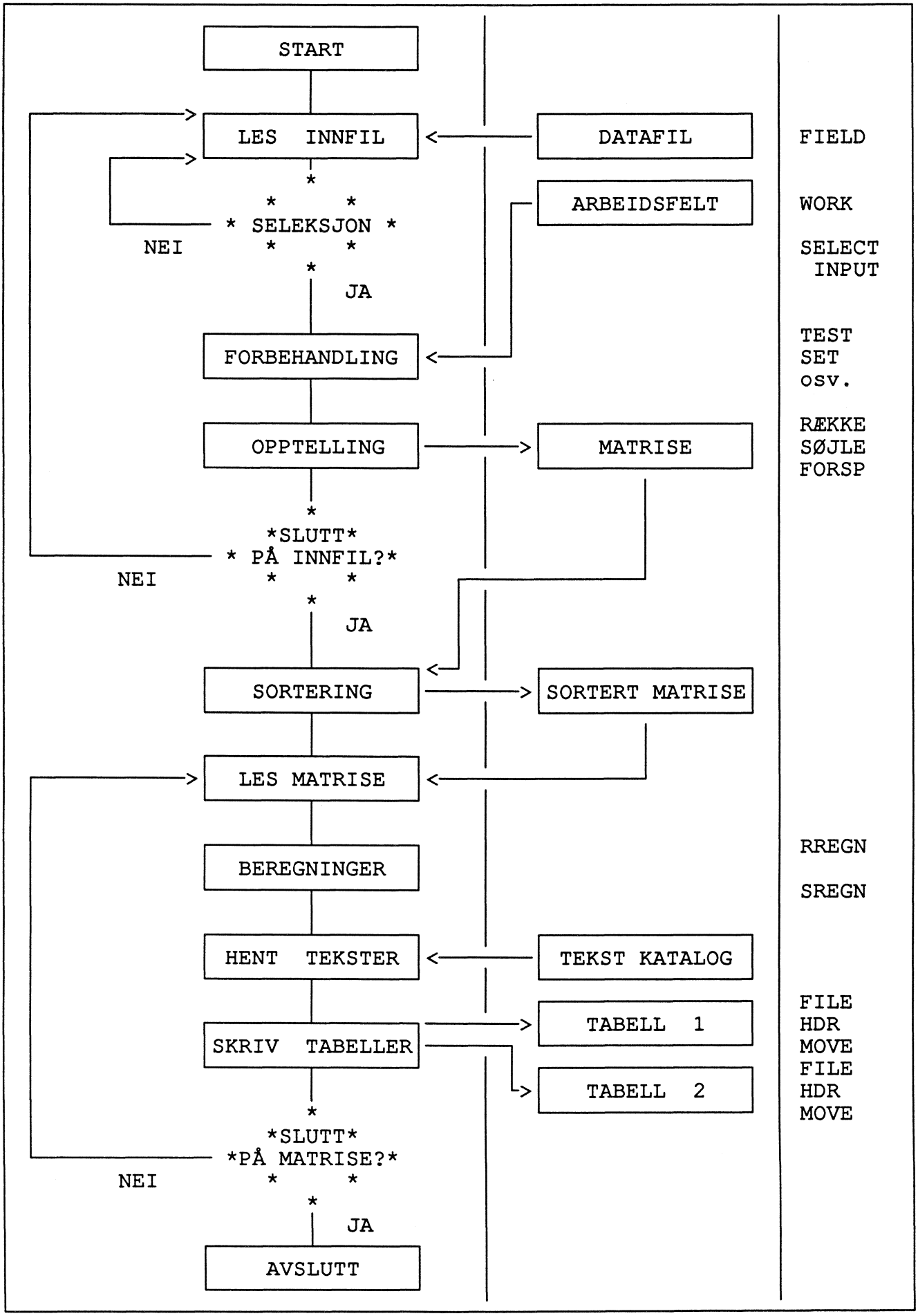
I Tab2 har forspalten for den enkelte tabell en helt fast oppbygging, og betingelsen for opptelling i en 'RÆKKE' kan angis for rekken spesielt. Det behøver altså ikke å være noen sammenheng mellom rekkene.

Det er også mulig å danne beregnede rekker ved RREGN som tilsvarer SREGN.

De instruksjonene som er like for Tab1 og Tab2, er bare tatt med i avsnittet om Tab1. Alle disse kan brukes i Tab2 på samme måte som i Tab1. Dette gjelder ikke CANCEL, DOPAGE, GETPST, LIN, TAEL og TOTAL som bare kan brukes i Tab1.

Input-filen til tabellprogrammet må være sekvensiell, og recordformatet kan være fast, variabelt eller udefinert. De felt som benyttes i programmet må ligge innenfor de 4096 første posisjoner på recorden.

Den enkelte tabell kan ha 1-9999 rækker (linjer) og 1-1000 søjler (kolonner) hvis TYPE=B eller 1-500 søjler hvis TYPE=P (TYPE angis i START-instruksjonen).


START
LES INNFIL
DATAFIL
FIELD
ARBEIDSFELT
WORK
* SELEKSJON *
NEI
SELECT
INPUT
JA
FORBEHANDLING
TEST
SET
osv.
OPPTELLING
MATRISE
RÆKKE
SØJLE
FORSP
SLUTT
* PÅ INNFIL?*
NEI
JA
SORTERING
SORTERT MATRISE
LES MATRISE
BEREGNINGER
RREGN
SREGN
HENT TEKSTER
TEKST KATALOG
SKRIV TABELLER
TABELL 1
TABELL 2
FILE
HDR
MOVE
FILE
HDR
MOVE
SLUTT
PÅ MATRISE?
NEI
JA
AVSLUTT

12.1. OPPTELLING I RÆKKER

Som i TAB1A og TAB1B kjøres programmet i 2 faser; den første er forbehandling og opptelling, den andre er beregninger og utskrift. Forskjellen mellom TAB1A/TAB1B og TAB2 ligger først og fremst i hvordan vi definerer forspalten vår. I TAB2 må vi definere hver eneste linje vi skal skrive ut til tabellen vår. Dette gjør vi med følgende instruksjoner:

CALL
CASE - ENDCASE
FIND
HOP
IF - ELSE - ENDIF
LOOP - ENDLOOP
RGRP - RGEND
RÆKKE
SET
SOEG

Instruksjonene RÆKKE og RGRP-RGEND tilsvarer SØJLE og SGRP-SGEND bortsett fra at de behandler linjer, ikke kolonner.

Vi kan angi at det skal telles med en verdi både i RÆKKE og SØJLE. Hvis vi gjør det vil alltid verdien som er angitt i SØJLE bli telt, dvs. at verdi i SØJLE har høyest prioritet.

Eksempel:

```
RÆKKE R1,,2
RÆKKE R2,,4
RÆKKE R3
SØJLE S1
SØJLE S2,,3
SØJLE S3,,5
```

Disse instruksjonene gir følgende tabell for 1 lest record:

	S1	S2	S3
R1	2	3	5
R2	4	3	5
R3	1	3	5

RÆKKE-instruksjonen er en del av forbehandlingen. Den er en slags avansert utgave av TAEL-instruksjonen i Tab1. Det betyr at vi kan bruke forbehandlingsinstruksjoner sammen med (og før og etter) RÆKKE.

Fordi vi i Tab2 må definere hver eneste linje vi skal lage, blir det mer programmeringsarbeid å lage en tabell i Tab2 enn i Tab1. Derfor bør vi alltid prøve å finne ut om en tabell kan lages i Tab1 før vi begynner å programmere i Tab2. Tab2 krever dessuten mer ressurser enn Tab1, og det er også en grunn til å prøve Tab1.

RÆKKE angir hvilke linjer det skal telles i, hvilke betingelser som eventuelt gjelder for opptellingen og hva som eventuelt skal telles.

Syntaks:

(1) RÆKKE RN,(BETINGELSE),VERDI

(2) RÆKKE RN1:RN2,(BETINGELSE),VERDI

(3) RÆKKE RN1:RN2,TALLYX,VERDI

Dette er helt analogt med SØJLE-instruksjonen (se denne).

Det telles i rækken hvis betingelse er oppfylt.

Eksempler:

```
RÆKKE R1
RÆKKE R2:R3,(KJØNN = 1:2)
```

Det telles i linje R1 for hver lest record, i linje 2 og 3 hvis kjønn er henholdsvis 1 og 2.

```
RÆKKE R1,(REDFYS = 1)
RÆKKE R2,(REDFYS = 1 AND KJØNN = 1)
RÆKKE R3,(REDFYS = 1 AND KJØNN = 2)
```

Helt likt det første eksemplet, bortsett fra at vi har fått en betingelse til for hver linje; REDFYS = 1.

```
RÆKKE R1:R3,(NÆRING2 = 2:3:5)
RÆKKE R4:R11,(NÆRING3 = 21:22:23:31:32:35:36:41)
RÆKKE R12:R20,(ALDER <= 14:19:24:29:34:39:44:80:)
```

Det telles i linje R1, R2 eller R3 hvis NÆRING2 er henholdsvis 2, 3 eller 5. R4 til R11 telles avhengig av hvilken verdi NÆRING3 har. For R12 til R20 gjelder at for R12 telles hvis alder er 0-14 år, R13 15-19 år, R14 20-24 år, R15 25-29 år, R16 30-34 år, R17 35-39 år, R18 40-44 år, R19 45-80 år og R20 81 år og over.

```
SET   TALLY7=FYLKE
RÆKKE R1:R20,TALLY7
```

Her setter vi først feltet TALLY7 lik fylkesnummer. Deretter teller vi opp i 20 linjer der innholdet av TALLY7 (dvs. fylkesnummeret) bestemmer hvilken linje vi skal telle i. Vi vil da få en linje for hvert fylkesnummer og linje 13 (ugyldig fylkesnummer) vil bli blank.

Fordi RÆKKE-instruksjonen er en forbehandlingsinstruksjon kan vi bruke RÆKKE sammen med IF, CASE og de andre forbehandlingsinstruksjonene.

RGRP avgrenser en gruppe med rækker. Rgrp tilsvarer Sgrp (se dette) bortsett fra at det gjelder rækker, ikke søjler.

Syntaks:

(1)
RGRP (BETINGELSE),ALL
(2)
RGRP RANTAL,(BETINGELSE),ALL

(1) Brukes for å avgrense en gruppe rækker. ALL gjør at det telles i alle rækkene i RGRP hvis betingelsen er oppfylt. Hvis ALL utelates, forlates RGRP når det er telt i 1 RÆKKE (som en CASE der vi ikke tester videre når en av testene har slått til).

Eksempler:

```
RGRP  (KJØNN = 1),ALL
RÆKKE R1:R7,(ALDER <= 15:19:24:49:66:74:)
RGEND
```

===> Hvis 'KJØNN = 1' telles det: i R1 hvis alder er mindre enn 16
i R2 hvis alder er 16-19
:
i R7 hvis alder er over 74

```
RGRP  (REDFYS = 1),ALL
RÆKKE R1
RÆKKE R2:R3,(KJØNN = 1:2)
RGEND
```

Eksemplet over gjør det samme som det andre eksemplet med RÆKKE.

(2) Brukes til å velge opptelling i en av flere søjlegrupper.

RANTAL angir antall rækker i hver rækkegruppe.
BETINGELSE angir hvilken gruppe det skal telles i.

```
RGRP  11,(FYLKE = 1:2:3:4:5:6:7:8:9:10),ALL
RÆKKE R1:R9,(ALDER <= 14:19:24:29:34:39:44:80:)
RÆKKE R10:R11,(KJØNN = 1:2)
RGEND
```

Her vil det bli dannet 110 linjer; 11 linjer for hvert av de 10 fylkene vi har med. R1 til R11 for Østfold (FYLKE = 01), R12 til R22 for Akershus osv. helt til Vest-agder (FYLKE = 10)

12.2. BESKRIVELSE AV FORSPALTEN **(TAB2)**

Det er hvordan vi definerer forspalten som skiller Tab1 fra Tab2. I RÆKKE-instruksjonen har vi definert hvilke linjer tabellen skal bestå av. I definisjonen av forspalten har vi to muligheter; den ene er å definere hvilke linjer som skal skrives ut. Da kan vi få skrevet ut linjer fra linje x til linje y, og linjenummeret vil da komme ut i forspalten. Den andre muligheten er å styre utskriften av linjene ved hjelp av en tekst-fil. I denne tekstfilen må det ligge en henvisning til hvilket linjenr en tekstlinje skal gjelde. I forspaltedirektivet må vi da beskrive hvordan denne filen ser ut.

Tekst-filen må inneholde følgende felt; linjetype, linjenr, tekst og styretegn til skriveren.

Linjetype

I tekst-filen kan vi ha to forskjellige slags linjer; elementærlinje og tekstlinje. Om det er den ene eller den andre typen avgjør vi med å skrive henholdsvis E og T (bokstavene må være store) i posisjon 1 i tekst-filen vår. En elementærlinje (E) må ha med en referanse til en linje, fordi det er elementærlinjene som skal få tall. Den andre linjetypen, tekstlinjen (T), skal som navnet tilsier, bare ha tekst.

Linjenummer

Linjenummer refererer til hvilken linje (RÆKKE) teksten hører til. Skal bare brukes til linjetype E (elementærlinje). I tekst-filen skal det stå minst 1 blank posisjon etter linjenummeret. Linjenumrene som er med i tekst-filen, må også defineres i programmet, vha RÆKKE slik at vi kan få en forbindelse mellom tekstfilen og programmet. Det er viktig å være nøye med definisjon av linjer (RÆKKE) og at tekstlinjene refererer til riktige linjer, slik at teksten blir riktig i forhold til tallene som står utenfor denne.

Tekst

Kort og godt den teksten vi skal ha ut i forspalten vår.

Styretegn til skriver

Ved hjelp av styretegn til skriveren kan vi fortelle programmet vårt hvor mange linjer vi skal hoppe over når tabellen skrives ut før linjen som styretegnet står på skrives ut.

Til å beskrive forspalten kan vi bruke følgende instruksjoner:

TABEL
FORSP

Også i Tab2 kan vi lage flere tabeller med lik forspalte (vha. TABEL). Vi kan ha opp til 8 TABEL-instruksjoner i et program.

TABEL beskriver oppsplitting i tabeller.

Syntaks:

(1)

TABEL FELTNAVN,TOTKEY=

(2)

TABEL FELTNAVN,TXTFIL=/MEMBER=,TXTKEY=,TXT=,
TOTKEY=/TOTTXT=,WORK=

(1) Beskriver tabellnivå uten tekstfil

(2) Beskriver tabellnivå med tekstfil

Forklaring:

FELTNAVN: Navnet på feltet som inneholder forspaltekriteriet (Kan ikke være en TALLY).

TXTFIL=DDname angir DD-navn på txt-filen

MEMBER=Membernavn anvendes når txt-filen er member i et partisjonert datasett.

TXTKEY=(START,LENGDE,TYPE) angir identens plassering og format i txt-recorden.

TXT=(START,LENGDE) angir hvor i txt-recorden teksten finnes.

TOTKEY=XXXX angir ident for txt-record med totaltekst (2), eller totalident (1).

TOTTXT='Totaltekst' angir tekst for total.

WORK=FELTNAVN brukes for å få flyttet txt/ident fra txt-filen inn i et arbeidsfelt som kan brukes i overskriften (HDR).

FORSP i Tab2 brukes ikke på samme måte som i Tab1. I Tab2 brukes den til enten å angi hvilke linjer som skal skrives ut (syntaks 1), eller til å beskrive tekst-filen som inneholder forspalteteksten (syntaks 2).

Syntaks 1:

(1)

```
FORSP RX:RY
```

Brukes når det ikke anvendes txt-fil.

Det lages 1 linje for hvert nummer i intervallet fra X til Y.

Syntaks 2:

(2)

```
FORSP TXTFIL=,TXT=(START,LENGDE),ASA=START,RAKNR=START,
      PRINT=START
```

Forklaring:

ASA=START angir startposisjon i txt-recorden hvor styretegnet til skriveren er angitt.

Styretegn til skriveren angir antall linjeskift før utskrift av linjen.

+	===> Ikke noe linjeskift (det skrives på den samme linjen)
	===> Et linjeskift (det skrives på den neste linjen)
0	===> Dobbelt linjeskift (en blank linje settes inn)
-	===> Trippelt linjeskift (to blanke linjer settes inn)
1	===> Sideskift

RAKNR=START angir posisjon i txt-recorden hvor linjenummeret starter.

Linjenummeret knytter teksten til en linje definert med RÆKKE-instruksjonen

PRINT=START angir hvor på tabellutskriften teksten skal skrives ut (Standardverdi: PRINT=1).

12.3. BEREGNING AV RÆKKER

Under beregning av rækker kan vi bruke disse instruksjonene:

CALL
PRCT
RREGN
SREGN

Som vi ser, kan vi ikke bruke på langt nær så mange instruksjoner i beregningsfasen i Tab2 som i Tab1.

Med RREGN kan vi gjøre beregninger på linjer med de 4 alminnelige regneartene (+, -, *, /). RREGN kan vi bruke før og/eller etter SREGN-instruksjoner, men ikke mellom.

Under beregninger med RREGN, SREGN og PRCT kan divisjon med 0 forekomme. Vi kan selv bestemme hvilket resultat 0-divisjonen skal få. Dette gjør vi i START-instruksjonen med parameteren NULDIV. Der angir vi hva resultatet av 0/0 og X/0 (X ulik 0) skal bli. Resultatet kan bli et tall eller at vi får abend (se START-instruksjonen).

Hvis vi lager nye rækker med RREGN, vil disse bli betraktet som elementærlinjer. Det betyr at linjer dannet ved RREGN også kan ha tekster i en tekst-fil.

12.3.1. Rregn **RREGN**

RREGN brukes til å foreta beregninger på enkeltlinjer (rækker). RREGN er analogt med SREGN, bortsett fra at beregninger gjøres på rækker og at det ikke kan benyttes parenteser i RREGN.

RREGN kan utføres før eller etter SREGN, eventuelt både før og etter. Hvis vi skal beregne prosent-rækker/søjler, er det viktig at summer er dannet før prosenter regnes ut.

Linjer dannet med RREGN oppfattes som elementærlinjer i forbindelse med tekst-filer.

Slik kan vi gjøre i SREGN, men ikke i RREGN:

```
RREGN R1:R5=RN/100
```

Her må vi gjøre slik:

```
RREGN R1=R1/100
RREGN R2=R2/100
RREGN R3=R3/100
RREGN R4=R4/100
RREGN R5=R5/100
```

Vi kan ikke regne ut nye linjer vha felt, kun vha linjenr og konstanter.

Ikke slik (verdien til feltet innbygg er beregnet i programmet):

```
RREGN R1=R1*100/INNBYGG
```

Men slik (vi må vite den eksakte verdien av innbyggere på forhånd):

```
RREGN R1=R1*100/4145845
```

PRCT gjør kolonner med absolutte tall om til kolonner med prosent av totalen til en fritt valgt kolonne.

Syntaks:

PRCT SX:SY/SZ,DEC=N,TOTRAK=NNN

Forklaring:

SX:SY Søjlene prosentene skal regnes på.

SZ Søjlen prosentene skal regnes av. Prosentene regnes ut fra totalen til denne søjlen. Hvis vi skriver SN, blir prosenten regnet ut av søjlens egen totalsum.

DEC= Antall desimaler

TOTRAK=NNN Linjenummeret til den linjen som inneholder totalen. Totalrekken må være definert med RÆKKE eller RREGN. Hvis vi ikke bruker TOTRAK vil totalen av samtlige linjer regnes ut, og denne vil bli brukt til prosentberegningen. Hvis du bruker flere PRCT-instruksjoner i et program, må alle ha med TOTRAK, men række- numrene behøver ikke å være like.

Eksempel:

```
PRCT  S1/S7,DEC=3,TOTRAK=11
PRCT  S1/S7,DEC=3
```

For den første av disse to PRCT blir SØJLE S1 regnet ut som verdien til S1 i prosent av totalen til S7. Totalen befinner seg i linje 11 (TOTRAK=11). For den andre PRCT blir det regnet ut en sum som prosentene skal regnes av. Denne summen er totalsummen av alle linjene. Denne forskjellen er det viktig å kjenne til.

Som i Tab1 kan vi velge om vi skal lage en tabellfil eller en sekvensiell fil. Hvis vi skriver tabellen vår til en sekvensiell fil, vil tekstene fra en eventuell tekst-fil komme i posisjon 1 og utover (så langt som teksten er definert). Alle felt må flyttes ut til tabellen med MOVE-instruksjoner.

Vi kan lage opp til 5 tabellfiler eller 1 sekvensiell fil i et TAB2-program.

Disse instruksjonene kan vi bruke i utskriftsfasen:

FILE
HDR
MOVE

CASE - ENDCASE
IF - ELSE - ENDIF

[i forbindelse med MOVE]

HDR-instruksjonen må være med når vi lager tabellfiler, men når vi skal lage en sekvensiell fil skal vi ikke ha med HDR.

Vi kan ikke bruke nestede IF og/eller CASE i forbindelse med MOVE; det betyr at en slik test må avsluttes før vi kan gjøre en ny (vi kan ha 1 CASE utenfor 1 IF, men det er alt).

I beregningsfasen og spesielt i utskriftsfasen kan vi ofte ha bruk for å gjøre forskjellige ting på forskjellige linjer. Da kan vi bruke det systemdefinerte feltet TALLY0.

Tally0

Tally0 er et systemdefinert felt som før utskrift av den enkelte linje vil inneholde elementærlinjenes rækkenummer. Dette kan vi bruke hvis vi skal gjøre tester på den enkelte række før utskrift.

Eksempel:

```
CASE   (TALLY0 = 01-10)
MOVE   (12,10,Z-)*3,S1
CASE   (TALLY0 = 11-20)
MOVE   (14,8,E-)*2,S4,+ZZZZ9,9
ENDCASE
```

For linje 1 til 10 vil kolonne S1, S2 og S3 bli skrevet ut, mens for linje 11 til 20 vil vi få skrevet ut kolonne S4 og S5 der tallene får en desimal.

```
//          EXEC TAB2
//INPUT     DD DSN=PP414.S8019.TABKURS.DATAFIL2,DISP=SHR
//AARFIL    DD *
1984
1985
1986
1987
//FORSPTXT  DD *
E01 1 HELE LANDET ......
T   0 AV DETTE:
T      BOSATT I BY MED
T      MER ENN 100 000
E02    INNBYGGERE ......
E03 0 01 ØSTFOLD .......
E04   02 AKERSHUS ......
E05   03 OSLO ..........
E06   12 HORDALAND .....
E07   16 SØR-TRØNDELAG .
E08   20 FINNMARK ......
//TABELL    DD SYSOUT=*
          START TYPE=P
AARGANG   FIELD (1,4,X)
FYLKE     FIELD (5,2,X)
INNB      FIELD (13,6,X)
AAR       WORK  (4,X),0
          RÆKKE R1
          RÆKKE R2,(INNB > 100000)
          RÆKKE R3:R8,(FYLKE = 1:2:3:12:16:20)
          SØJLE S1,,INNB
          TABEL AARGANG,TXTFIL=AARFIL,TXTKEY=(1,4,X),TXT=(1,4),
                WORK=AAR
          FORSP TXTFIL=FORSPTXT,TXT=(7,18),ASA=5,RAKNR=2,PRINT=1
 TABELL   FILE  PRINT,OVERFLOW,MAXLIN=46
          HDR   1,1,'INNBYGGERE.'
          HDR   1,13,AAR
          MOVE  (23,10,Z-),S1
```

Dette programmet gir denne tabellen (en slik tabell pr. år):

```
INNBYGGERE. 1985

HELE LANDET ......      4145845

AV DETTE:
 BOSATT I BY MED
 MER ENN 100 000
 INNBYGGERE ......       788842

01 ØSTFOLD .......       235039
02 AKERSHUS ......       386278
03 OSLO ..........       447351
12 HORDALAND .....       397614
16 SØR-TRØNDELAG .       246407
20 FINNMARK ......        76650
```

VEDLEGG

VEDLEGG I: FEILMELDINGER (ABENDKODER) TIL TAB-PROGRAM

System-abends:

S001 1. 2 filer med forskjellige recordlengde er forsøkt konkatenert (lest sammen). Filene skal ha samme recordlengde.
2. 2 filer med forskjellig blokkstørrelse der den filen med minst blokkstørrelse er lest først er forsøkt konkatenert (lest sammen). Dette går ikke, den filen som har minst blokkstørrelse skal ikke leses først. Bytt om rekkefølgen på filene.
3. Innfilen er tom. Sjekk om det er riktig datasett. Kan være lagd to ganger. Det riktige vil da ikke være katalogisert (NOT CATLG 2).
4. Du har brukt DOPAGE i TAB1A, og i tillegg har du enten 512 søjler og TYPE=P eller 1024 søjler og TYPE=B. DOPAGE er ikke lov i TAB1A, bruk TAB1B isteden.

S013 1. Du har prøvd å lage en fil (i JCL) der blokkstørrelsen ikke er et multiplikat av recordlengden. Det må den være (gjelder filer med fast recordlengde (F,FB,FBA)).
2. Du har prøvd å lese et member som ikke eksisterer. Sjekk membernavn på tekstdatasettene dine.

S0C1 1. Hvis du har mer enn 9 TABEL-instruksjoner i programmet ditt vil du få denne feilkoden.
2. Programmet vil stoppe hvis det blir gjort forsøk på å regne med ikke-numeriske felt. I TAB1 får du feilkode S0C1, i TAB1A, TAB1B og TAB2 får du feilkode U0107 (se denne). Du får ikke ut noen kjørelogg fra Tab, men du får ut tabellen så langt den kom før ABEND. Lag en test i forbehandlingen av dataene som tester på hva som står i feltet som forårsaker ABEND, og gjør det om til et numerisk felt.
3. Du har prøvd å flytte ut et indeksert felt der indeksen er større enn antallet indekserte felter. Tabellen vil være skrevet ut til der du prøver å flytte ut feltet som gir feilkoden. Sjekk indeksene dine, du har gjort en feil slik at indeksen din er blitt for stor.
4. SYSIN mangler. Maskinen har ikke funnet programmet ditt. Sett inn en slik linje før programmet ditt:
```
//SYSIN     DD *
```

S0C4 1. Feil recordlengde på en fil som kopieres inn som en del av programmet. Filbeskrivelsen av innfilen kan ligge på en egen fil. Denne filen skal ha samme recordlengde som den filen programmet ditt ligger på har. Feilen kan oppstå når du bruker COPY-instruksjonen.
2. Hvis du bruker for mange LIN-instruksjoner, kan denne feilen oppstå (grensen er et sted mellom 20 og 32).
3. Du har prøvd å lese etter recordens slutt i en FYLD-fil. Sjekk riktig recordlengde og at data du skal fylle opp ligger innenfor maks. recordlengde.

System-abends (forts):

S737 Hvis RC=24 har du oppgitt et member i JCL som ikke finnes på fila. Sjekk partisjonerte datasett som er brukt i JCL!

S80A For liten region spesifisert. REGION=1024K eller REGION=2048K settes i EXEC-kortet (EXEC TAB1A,REGION=1024K).
S878 For liten region spesifisert. REGION=1024K eller REGION=2048K settes i EXEC-kortet (EXEC TAB1A,REGION=1024K).

S90A For liten region spesifisert. REGION=2048K eller REGION=4096K settes i EXEC-kortet (EXEC TAB1A,REGION=2048K).

SA03 Ingen opptelling foretatt.
1. Hvis STOP er med og ikke TAEL, vil det ikke skje noen opptelling. Derfor fjern STOP.
2. Hvis ingen records selekteres, dvs. det ikke blir noen data å lage tabell av, vil du få denne meldingen. Sjekk at logikken i SELECT er riktig.

User-abends:

U0000 Som regel betyr denne at det er en systemfeil, og den har en egen abend-kode (F. eks. S0C1, SA03). Sjekk disse!

U0004-
U4092 Hvis du bruker DOPAGE i TAB1A vil du få varierende feilkoder alt etter hvor mange søjler du har i programmet. Bruker du TYPE=B vil feilkoden være 4 ganger antall søjler i programmet ditt. Med TYPE=P vil feilkoden være 8 ganger antall søjler. Ikke spør meg hvorfor!! Bruk TAB1B hvis du må bruke DOPAGE.

U0100 DD-statement mangler. Du har angitt et DD-navn i programmet som ikke står i JCL-en din. Det aktuelle DD-statementet står etter: DD-statement missing. Sjekk JCL!

U0104
1. Tekstdatasettet ditt er tomt. Sjekk tekstdatasettene.
2. Du har under i forbindelse med SREGN prøvd å gi et felt i innfilen en verdi uten at feltet er et forspaltekriterium. Dette er ikke lov fordi innfilen ikke er tilgjengelig under SREGN. Felt fra innfilen som er forspaltekriterier kan derimot gis verdi under SREGN.
3. Du prøvd å telle opp et felt som ligger utenfor recorden. Sjekk filbeskrivelsen din.

U0106 Hvis summen av TABEL -og FORSP-instruksjoner er mer enn 13 vil du få denne feilkoden.

U0107
1. Forsøk på å utføre regneoperasjoner med ikke numeriske felt. Lag en test i forbehandlingen av dataene som tester på hva som står i feltet som forårsaker ABEND, og gjør det om til en numerisk verdi.
2. Hvis du har prøvd å regne med et felt som ligger utenfor recorden (f.eks prøve å regne med et felt som starter i posisjon 37, mens inputfilens recordlengde er 29), vil denne meldingen dukke opp. Sjekk at filebeskrivelsen stemmer overens med inputfilen og/eller at du har riktig innfil i JCL-en din.
3. Du har angitt en indeks som ikke er numerisk. Sjekk hvilke felt som brukes som indekser og sørg for at de alltid er numeriske.
4. Hvis du har prøvd å hoppe (HOP) fra beregningsfasen (SREGN) til forbehandlings- eller opptellingsfasen vil programmet gå ad undas. HOP kan bare brukes innen forbehandlingen eller i beregningsfasen.
5. Du har prøvd å flytte ut et felt som ikke er numerisk med en MOVE-instruksjon som krever numeriske felt. Dvs. MOVE med type lik Z, Z-, ZB, E, E-, eller EB (eks; MOVE (7,4,Z-),FELT). Sjekk hvilke verdier feltet du flytter ut har, og sørg for at de alltid er numeriske.
6. Hvis du prøver å flytte en ikke-numerisk verdi til et pakket felt, vil programmet avbrytes med kode U0107.
7. Hvis du prøver å regne med felt fra input-filen i SETUP, går det galt. I SETUP er ikke input-filen tilgjengelig, da hele poenget med SETUP er å utføre instruksjoner før vi begynner å lese inputfilen. Derfor dropp instruksjoner som bruker felt fra inputfilen i SETUP!

U0107 (forts)

8. Hvis du har regnet med arbeidsfelt i SREGN og har brukt TYPE=B parameteren i START-instruksjonen går det galt. Dette gjelder bare hvis arbeidsfeltene dine ikke er binære. Derfor, definer arbeidsfeltene dine binære.

U0108

Et aggregert tall er blitt større enn grensen for binære tall (2**31 - 1 el. ca 2.1 * 10**9). Hvis du får større tall enn dette, kan du ikke bruke TYPE=B (oppgis i START-instruksjonen). Bruk TYPE=P isteden.

U0109

1. Tallet du prøver å telle opp er for stort for binær opptelling (TYPE=B). Prøv TYPE=P isteden. Se U0108.
2. Divisjonen 0/0 eller N/0 (N er et heltall) er forsøkt utført og lagt ut i et binært felt. Verdier som skal legges ut i binære felt må være tall. Se U0111

U0110

1. Du har prøvd å flytte et tall til et pakket felt som ikke har vært stort nok til tallet. Øk feltets lengde. Eksempel: Et pakket felt som er definert 3 posisjoner (3,P) kan inneholde tall fra -99999 til +99999. Prøver du å flytte større tall til feltet vil du få U0110. Da må du øke feltets lengde til 4 (4,P).
2. Tallet du prøver å telle opp er for stort også for opptelling med pakket desimal (TYPE=P, se U0108 & U0109). Tab kan dessverre ikke regne med tall som er større enn 10**15).

U0111

Divisjonen 0/0 eller N/0 (N er et heltall) er blitt gjort i forbehandlingen. Du bør teste om nevneren er lik 0, og i så fall gi en verdi til kvotienten (resultatet av divisjonen) istedenfor å utføre divisjonen når nevneren er lik 0.

U0117

Invalid index. Du har prøvd å referere til en index som ligger utenfor den array'en du har definert. Sjekk tellere til array-indexene dine.

U0125

Divisjon med 0 i SREGN. Sett NULDIV-parameteren i START-instruksjonen (se denne).

U0173

Koden til en betingelse tar mer enn 4K. Grensen går på 4K. Prøv å forenkle betingelsen.

U0900

1. Du har prøvd å kjøre TAB1 på en fil som ikke er sortert på forspaltekriteriene (Dette er et krav for å kjøre program med TAB1). Kjør programmet med TAB1A istedenfor TAB1.
2. Du har brukt ABEND-instruksjonen, og den har slått til. Sjekk kjøreloggen og programmet ditt.

User-abends (forts):

U0901

1. Feil i sortworkområde. Til TAB2 skal det hete SRT1WK01.
2. For liten region spesifisert. REGION=1024K eller REGION=2048K settes i EXEC-kortet (EXEC TAB2,REGION=1024K).
3. Driftstekniske problemer. Feilen vil bli rettet av driftskontoret.
4. Programmet går ikke når du bruker TAB2's JCL-prosedyre. Det kan hende at det går hvis du ikke bruker denne, men kaller opp TAB2-programmet uten JCL-prosedyren. Da går det kanskje bra (jeg aner ikke hvorfor, JCL-en vil bli den samme i begge tilfeller!).

U0903 Flere records i en FYLD-tabell enn du har oppgitt som max. Øk MAX= i FYLD-instruksjonen.

U0904 Ikke plass til FYLD-fil(ene). Grensen er 500K. Prøv å redusere dataene i FYLD-filene.

U0906

1. Mer enn 10 feil i txt-datasett. Som oftest er det koder som ikke finnes i txt-datasettet. Disse kodene og hvilket txt-datasett det gjelder vil være angitt. Vi kan enten rette txt-datasettet eller input-filen, eller vi kan øke antall tillatte txt-feil (dette gjøres i START-instruksjonen).
2. For mange linjer i txt-datasettet med samme kode. Hvis det er mer enn 1 linje med tekst pr. kode, må du ha med NEWLINE i FORSP-instruksjonen din.

U0999

1. Loaded program ikke kompatibelt med prosessor. Hvis du har prøvd å kjøre TAB1 og får denne meldingen, kan du prøve å kjøre om igjen med TAB1A. Feilen skyldes at du har brukt enten TABEL eller PRCT eller at du har brukt et work-felt som forspaltedirektiv. Det får du ikke lov til i TAB1.
2. Selkort (SELKRIT) har ugyldig verdi eller ikke verdi i det hele tatt. Sjekk SELKRIT.
3. Feil bruk av HOP-instruksjonen. Hvis du har prøvd å bruke HOP i utskriftsfasen, vil det skjære seg. Du kan ikke bruke HOP der.
4. Du har gitt en ugyldig parameterverdi til en subrutine som du kaller opp. Sjekk disse!
5. Gal bruk av LIN-instruksjonen. Du har ikke lov til å ha en LIN-instruksjon før første MOVE-instruksjon. Fjern den første LIN.
6. GETPST i TAB1A. Beklager, men det går nok ikke. Prøv TAB1B isteden!

NB! Listen er absolutt ikke fullstendig. Jeg tar gjerne imot meldinger om nye koder og andre feiltyper til de feilkodene som er i denne listen.

Kristian Lønø

VEDLEGG II: HVORDAN STYRE FERDIGE TABELLER TIL EN FIL?

Det første du må gjøre er å opprette en fil til tabellene dine. Dette kan gjerne være et partisjonert datasett (PDS), dvs. at filen har membere. Vi lager et PDS ved først å gå inn i punkt 3.2 fra hovedmenyen. Der velges OPTION A (allokering av datasett). Dessuten må du oppgi navnet på filen du skal lage. Dette er vist her:

```
------------------------------ DATA SET UTILITY -------------------------------
OPTION  ===> A

   A - Allocate new data set                C - Catalog data set
   R - Rename entire data set               U - Uncatalog data set
   D - Delete entire data set               S - Data set information (short)
   blank - Data set information

ISPF LIBRARY:
   PROJECT ===>
   GROUP   ===>
   TYPE    ===>

OTHER PARTITIONED OR SEQUENTIAL DATA SET:
   DATA SET NAME  ===> 'TK414.S8019.KRL.TABELLER'
   VOLUME SERIAL  ===>          (If not cataloged, required for option "C")

DATA SET PASSWORD ===>          (If password protected)
```

Vi skal lage en fil som heter TK414.S8019.KRL.TABELLER. Husk at filnavnet skal stå i fnutter (''). Etter å ha skrevet inn de nødvendig opplysninger i dette skjermbildet trykker du [Enter]. Da vil du få opp et skjermbilde som, når du har tastet inn alle nødvendige opplysninger, skal se slik ut:

```
------------------------ ALLOCATE NEW DATA SET -------------------------------
COMMAND ===>

DATA SET NAME: TK414.S8019.KRL.TABELLER

   VOLUME SERIAL      ===> SSB904      (Blank for authorized default volume) *
   GENERIC UNIT       ===>             (Generic group name or unit address) *
   SPACE UNITS        ===> BLOCK       (BLKS, TRKS, or CYLS)
   PRIMARY QUANTITY   ===> 200         (In above units)
   SECONDARY QUANTITY ===> 50          (In above units)
   DIRECTORY BLOCKS   ===> 10          (Zero for sequential data set)
   RECORD FORMAT      ===> FBA
   RECORD LENGTH      ===> 133
   BLOCK SIZE         ===> 23142
   EXPIRATION DATE    ===>             (National format or blank)

   ( * Only one of these fields may be specified)
```

Vi har angitt 10 Directory Blocks, dette gjør at datasettet blir partisjonert. Hvis du ikke vil ha et PDS, skriver du 0 i dette feltet. Trykk [Enter] når skjermbildet er riktig utfylt. Du vil nå få en melding om at datasettet er allokert (i øvre høyre hjørne).

Neste steg blir å endre i JCL-en til programmet slik at tabellen styres til filen vi har laget istedenfor til kjørerapporten (SYSOUT).

La oss se på hvordan JCL-en må endres for å styre en tabell ut på filen vår. Under viser vi JCL der tabellen styres til kjørerapporten:

```
//0414KRLW JOB (8019),'Tabell til Sysout',MSGCLASS=X,
//         CLASS=A,MSGLEVEL=(0,0)
//OPPG7    EXEC TAB2
//INPUT    DD   DSN=TAB.DIV(DATAFIL1),DISP=OLD
//TABELL   DD   SYSOUT=*
//SYSIN    DD   *
```

Det er ikke store endringer som må gjøres:

```
//0414KRLW JOB (8019),'Tabell til fil',MSGCLASS=X,
//         CLASS=A,MSGLEVEL=(0,0)
//OPPG7    EXEC TAB2
//INPUT    DD   DSN=TAB.DIV(DATAFIL1),DISP=OLD
//TABELL   DD   DSN=TK414.S8019.KRL.TABELLER(TABELL7),DISP=OLD
//SYSIN    DD   *
```

Legg merke til at vi har oppgitt et member i tillegg til datasettnavnet. Det må vi gjøre siden vi lagde et PDS. Fordelen med det er at du nå kan lage andre tabeller som legges ut på samme datasett, du gir dem bare andre membernavn. Da slipper du å lage filer i et sett.

NB!
Det er viktig at FILE-instruksjonen din har med både LRECL=133 og BLKSIZE=23142 når du bruker recordlengde på 133 slik som under her.

```
TABELL   FILE  PRINT,OVERFLOW,MAXLIN=90,LRECL=133,BLKSIZE=23142
```

Glemmer du dette vil filen din få endret blokkstørrelsen til 1995.

Vi kan ikke legge ut tabeller til flere membere i samme PDS i samme kjøring. Det lar seg dessverre ikke gjøre (vi vil få den første tabellen som skal skrives ut i alle membere).

Når du har kjørt programmet og lagt tabellen ut på et member i PDS-et ditt, kan du lese og editere på den ferdige tabellen hvis det skulle være nødvendig. Når all redigering av tabellen og laserkoder er lagt inn, gjenstår det bare å få skrevet ut tabellen. Det gjør du ved å velge punkt 3.6 fra hovedmenyen. Ferdig utfylt bør skjermbildet du får se slik ut (det som står med uthevet skrift må du skrive inn):

```
--------------------------  HARDCOPY UTILITY  ----------  MEMBER TABELL7 SAVED
OPTION  ===> PK


   PK - Print/punch and keep data set
   PD - Print/punch and delete data set



DATA SET NAME ===> 'TK414.S8019.KRL.TABELLER(TABELL7)'
  VOLUME SERIAL     ===>              (If not cataloged)
  DATA SET PASSWORD ===>              (If password protected PDS)

SYSOUT CLASS      ===> A
LOCAL PRINTER ID ===>

JOB STATEMENT INFORMATION: (If not to local printer, verify before proceeding)
  ===> //O414KRLW JOB (8019),'Til laser (RMT15)',MSGCLASS=X,
  ===> //         CLASS=A,MSGLEVEL=(0,0)
  ===> /*JOBPARM LINECT=0
  ===> /*ROUTE PRINT RMT6
```

Etter å ha trykket [Enter] vil skjermbildet se slik ut:

```
--------------------------  HARDCOPY UTILITY  ----------------  JCL GENERATED
OPTION  ===> PK

   PK - Print/punch and keep data set
   PD - Print/punch and delete data set
   CANCEL - Exit without submitting job

Enter END command to submit job.

DATA SET NAME ===> 'TK414.S8019.KRL.TABELLER(TABELL7)'
  VOLUME SERIAL     ===>              (If not cataloged)
  DATA SET PASSWORD ===>              (If password protected PDS)

SYSOUT CLASS      ===> A

JOB STATEMENT INFORMATION:
      //O414KRLW JOB (8019),'Til Oslo  (RMT6)',MSGCLASS=X,
      //         CLASS=A,MSGLEVEL=(0,0)
      /*JOBPARM LINECT=0
      /*ROUTE PRINT RMT6
```

Du kan nå enten skrive inn et nytt filnavn for å få skrevet ut en tabell til, eller du kan trykke [PF3]. Dette vil sende tabellene(e) dine til skriveren.

Dette bør medføre at du vil få dine vakre tabeller pent skrevet ut.

VEDLEGG III: OPPTELLINGSSYSTEMET I TAB

Hvis man skal lage store tabeller i TAB, er det nødvendig å ha forståelse for hvordan opptellingen i TAB foregår. Med store tabeller forstås i denne sammenheng tabeller med mer enn 100 000 tabellceller (søyler x linjer) hvis TYPE = B, 50 000 tabellceller hvis TYPE = C.

I opptellingsfasen bruker TAB et areal på 600K, 600 * 1 024 bytes. For hver linje i tabellen inneholder arealet et pekerelement og selve linjen.

Pekerelementet består av to felt på hver 4 bytes.
1. felt 1 er en peker (adresse) til linjen.
2. felt 2 er en teller, funksjonen vil bli gjort rede for senere.

Linjen består av:
1. Identifikasjonen, som består av samtlige felter som er nevnt i TABELL- og FORSP-instruksjonene.
   Lengden av identifikasjonen er summen av disse feltenes lengde, rundet opp til nærmeste multipla av 4.
2. Søylene, antall søyler bestemmes av det høyeste søylenummeret som brukes i tabellen.
   Hver søyle er enten 4 bytes (TYPE=B) eller 8 bytes (TYPE=P).

Antallet linjer, kan så beregnes som

600 * 1 024 / (8+a+(b*c)),

der

- a er lengden av forspaltefeltene rundet opp til multipla av 4
- b er største søylenummer og
- c er 4 for TYPE=B og 8 for TYPE=P.

Dette antallet underskrives i forbindelse med kjørerapporten som MAX ANTALL LINJER I STORAGE.

OPPTELLINGSFASEN

Under selve opptellingen i TAB skjer lokaliseringen av den linje som skal telles i forbindelse med TAEL, for TAB2 ved hver REKKE hvis betingelsene er oppfylt.

Først testes om det allerede er dannet (finnes) en linje med samme identifikasjon som den linjen det skal telles i. Er dette tilfellet brukes denne linjen.

Hvis det ikke finnes noen linje, dannes en ny linje i tabellen, dvs. det opprettes et nytt pekerelement og identifikasjon settes på linjen og søylene nullstilles. Deretter brukes denne linjen.

Slik fortsetter prosessen så lenge det er ledige plasser i tabellen.

Hvis det ikke er mer plass i tabellen når en ny linje skal dannes, begynner programmet å bruke eksternt lager (disk), ved å overføre linjer til sort (datasett som programmet bruker under eksekveringen).

For å bestemme hvilke linjer som skal overføres til sort, brukes 2. felt i pekerelementet.

Fra det tidspunkt hvor tabellen er mer enn 75% fylt opp, brukes dette feltet som en teller, idet det telles for hver gang linjen velges for opptelling.

Når tabellen er full, overføres først alle linjer som ikke har vært referert, dvs. hvis teller er < 1.

Hvis ikke dette frigir minst 1/8 av linjene, fortsettes det med linjer der teller er < 2.

Deretter fortsettes det på samme måte med grensene 4, 8, 16, 32 osv. inntil minst 1/8 av linjene er frigitt.

Deretter nullstilles telleren for alle linjene som er igjen i tabellen, og opptellingen fortsetter som beskrevet ovenfor, herunder regelen om opptelling hvis tabellen er fylt med mer enn 75%.

Blir tabellen full igjen, gjennomføres igjen overføring til sort, og så fremdeles, inntil hele inputfilen er behandlet.

Har sort vært brukt under opptelling, overføres de linjer som fortsatt er i tabellen til sort på slutten av opptellingen.

En konsekvens av denne teknikken er at det på sortdatasettet kan finnes mange linjer med samme identifikasjon, skrevet ut under hver sin "opprydding" av tabellen.

I verste fall kan antallet linjer på sortdatasettet bli like stort som antallet records i inndatasettet og mange ganger større enn antallet linjer i tabellen.

KONSEKVENSER AV OPPTELLINGSTEKNIKKEN

Den vesentligste konsekvensen av den valgte opptellingsteknikken er at det er en sannhet med modifikasjoner når det påstås at TAB ikke krever at inputfilen er sortert, selv om dette er sant for de fleste TAB-kjøringer.

For å belyse konsekvensene, skal vi her se på et par konkrete eksempler.

Eks. 1.
Input 300 000 records
Forspalte med 30 000 elementærlinjer,
max. 5 000 linjer i storage

I dette eksemplet vil opptellingsalgoritmen fungere, idet de 4 000 aktive linjene reelt vil bli i tabellen, mens de siste 26 000 linjene vil bli overført løpende til sort, inkludert mange dobbeltgjengere, men antallet linjer på sortdatasettet vil ikke overskride 64 000 linjer, eller mye mer enn det dobbelte av antallet elementærlinjer, som vil være tilfredsstillende, og mer optimalt enn presortering av inputfilen.

Eks. 2.
Input 300 000 records
Forspalte med 10 000 elementærlinjer.
max 5 000 linjer i storage.

Fordelingen av records mot elementærlinjene er jevn, dvs. at det telles mellom 10 og 60 records i 90% av elementærlinjene.

I dette eksemplet vil det bli problemer fordi tabellen vil bli fort fylt opp (etter litt mer enn 10 000 records), og det vil ikke være noen opplagte kandidater for utskifting. Sjansen for at den linjen som skal telles er i storage, vil når tabellen er nesten full være ca. 50% dvs. for annenhver record dannes en ny elementærlinje, så tabellen blir fort full. Resultatet vil bli at det overføres mellom 200 000 og 300 000 records til sortdatasettet, dvs. ca. 25 ganger antallet elementærlinjer. Er disse linjene enda lengre enn de originale inputrecords blir resultatet en mye større sortering enn hvis input hadde vært presortert.

Det er også her nødvendig å gjøre det klart, at mens en sortering av innfilen før tabellprogrammet kjøres kan utnytte hele regionen som internt lager, dvs. ca. 800K i praksis, må sortering under TAB bruke et langt mindre internt arbeidslager (ca. 60K). Dette gjør denne noe langsommere ved store sorteringer.

Eks. 3.

Som eksempel 2, men
forspalten er oppbygd av to felt A og B.
A har et verdisett på 20 verdier, B et verdisett på 500 verdier, dvs. at produktet av A og B er 10 000 (antallet av elementærlinjer.

Input er sortert på A, men ikke på B.

I dette tilfellet fungerer algoritmen optimalt. For hvert skift i A bygges det opp 500 linjer i tabellen. Etter 7 skift av A overskrides grensen på 7%, hvilket medfører at opptelling i 2. pekerfelt begynner.

Etter at de første 5 000 linjer er dannet og tabellen derfor er fylt opp (11. verdi av A), vil alle de elementene som knytter seg til de første 6 verdiene av A blir overført til sort.

Når tabellen igjen blir full, vil elementer som knytter seg til verdien av A som er passert, bli overført og så videre. Resultatet vil bli at sort-datasettet vil bestå av 10 000 linjer, som i det aktuelle eksemplet vil være korrekt sortert.

Dette eksemplet viser betydningen av delvis sortert input.

Nettopp dette viser at sortering, helt eller delvis på et eller flere forspalte-felter, ofte gjør eksekveringen av programmene mer effektiv.

Skal man f.eks. lage en serie av tabeller, hvor en variabel inngår i samtlige forspalter, f.eks. en serie tabeller fordelt på kommunekode sammen med forskjellige andre kriterier, vil sorteringen på denne variabel ofte kunne løse problemene for samtlige tabeller.

Det er i denne forbindelse uten betydning om det inputfilen er sortert på, er det høyeste forspalte-kriterie i alle tabeller.

PARM='SIZE=nnnK'.

I situasjoner der antallet linjer i tabellen overskrider antallet linjer som kan være i storage med mindre enn 50%, kan man sette størrelsen av det interne tabellareal opp gjennom parameteren SIZE=nnnK, hvor nnn max kan være 999.

På denne måten kan antallet linjer som kan være i tabellen økes med ca. 65%.

Tabeller med få søyler.

Hvis man har store tabeller med få søyler, kan den måten linjene bygges opp på i tabellarealet få meget negative konsekvenser.

Hvis man f.eks. har en tabell med 2 binære søyler (i alt 8 bytes), og et identifikasjonsfelt-felt med en lengde på 16 bytes, vil dette bety at det i tabellen brukes 24 (16+8) bytes til id og peker for hver gang den bruker 8 bytes til det egentlige innhold, søylene. Resultatet blir altså at 24/32 eller 2/3 av den interne tabell brukes til å lagre identer og peker, kun en tredjedel til å lagre egentlige tabelldata.

I slike tilfeller, må man hvis det er mulig, starte med å danne en grunntabell, hvor et (evt. flere) forspalte kriterier omdannes til søylekriterier, og siden omforme denne grunntabellen.

Har man f.eks. en forspaltevariabel med 10 mulige verdier, som man istedet legger på søylenivå i eksemplet, således at vi får 20 søyler, endres forholdet mellom identer/pekere og søyler til 24/104 eller ca. 1/4, dvs. 3/4 nå inneholder egentlige tabelldata.

TYPE=.

En annen vesentlig parameter å observere i denne sammenheng er TYPE-parameteren (angis i START-instruksjonen). Er det problemer med plassen, bør TYPE=B alltid brukes, med mindre det kan forventes at verdien av noen tabellceller, inkludert totaler, kommer til å overstige 2**31 eller ca. 1,6 milliarder, idet TYPE=B kun bruker 4 bytes pr. søyle, mot 8 bytes pr. søyle for TYPE=P.

VEDLEGG IV: BRUK AV FYLD OG FIND TIL Å LAGE RATETABELLER

Å lage ratetabeller i TAB krever at man behersker TAB ganske godt. Vi skal nå se på hvordan vi lager en tabell for utskrivninger fra sykehus pr. 1000 innbyggere. La oss anta at vi skal lage en tabell som ser ut som denne her (tallene er ikke riktige, men det er ikke poenget nå):

Tabell 1. Utskrivinger fra somatiske sykehus etter alder, kjønn og bostedsfylke. Pr. 1000 innbyggere. 1988

Kjønn og fylke	I alt	Alder						
		0-9	10-19	20-39	40-59	60-69	70-79	80-
BEGGE KJØNN								
HELE LANDET	5,102	3,785	2,225	4,610	4,282	7,179	10,311	13,438
Østfold	5,185	3,188	2,062	4,139	4,607	7,233	12,268	13,976
Akershus	4,551	3,486	1,790	4,462	4,095	6,903	9,619	14,134
Oslo	5,339	3,343	2,029	4,256	3,701	6,751	10,654	17,204
Hedmark	5,486	3,908	2,541	4,942	4,677	7,475	10,055	12,099
Oppland	4,719	2,909	2,159	4,305	4,248	6,975	8,677	9,468
Buskerud	5,285	3,339	2,349	4,993	4,769	7,161	9,816	13,380
Vestfold	4,912	3,507	2,151	4,336	4,282	7,693	9,322	11,990
Telemark	4,936	4,157	2,822	4,441	4,000	6,567	9,476	9,043
Aust-Agder	5,164	3,760	2,189	5,096	4,190	6,955	10,060	13,607
Vest-Agder	5,235	3,588	2,316	4,961	5,160	7,313	10,116	12,291
Rogaland	4,439	3,655	1,947	4,461	3,570	6,276	9,139	12,196
Hordaland	4,775	4,253	2,009	4,092	4,108	6,620	10,224	12,231
Sogn og Fjordane .	5,798	4,255	2,254	5,053	4,902	8,325	11,728	13,678
Møre og Romsdal ..	5,158	4,630	2,132	4,797	3,783	6,963	10,009	14,437
Sør-Trøndelag	4,734	3,441	2,251	4,484	3,828	6,390	8,992	13,227
Nord-Trøndelag ...	5,809	3,200	2,691	6,041	4,800	8,623	10,269	14,697
Nordland	6,169	5,480	3,040	5,375	5,349	8,382	12,737	12,779
Troms	5,277	3,486	2,392	4,512	4,528	8,584	12,029	16,805
Finnmark	6,160	4,149	2,464	6,158	5,739	10,653	13,714	11,236
MENN								
HELE LANDET	4,462	4,265	1,965	2,283	4,183	8,240	12,056	15,991
Østfold	4,797	3,513	1,964	2,474	4,867	7,853	14,349	16,102
Akershus	3,846	3,843	1,528	2,064	3,827	7,820	11,429	19,267
Oslo	4,470	4,020	1,941	1,970	3,712	8,415	12,388	19,272
Hedmark	5,158	4,441	2,339	2,956	4,528	8,756	11,271	15,064
Oppland	3,827	3,174	1,511	1,915	3,600	7,421	8,827	11,218
Buskerud	4,579	3,547	2,127	2,518	4,477	8,176	10,832	17,247
Vestfold	4,333	4,135	1,771	2,269	3,836	8,720	11,387	15,004
Telemark	4,374	4,320	1,892	2,298	3,995	6,885	12,315	11,307
Aust-Agder	4,662	4,175	1,506	2,917	4,171	8,253	11,757	18,732
Vest-Agder	4,649	5,080	1,931	2,164	5,066	9,531	11,182	13,619
Rogaland	3,694	4,312	1,716	1,768	3,577	6,637	11,067	15,654
Hordaland	4,330	4,460	1,985	2,183	4,413	7,918	12,023	14,321
Sogn og Fjordane .	5,586	5,998	2,146	2,648	4,092	11,445	13,675	18,173
Møre og Romsdal ..	4,796	4,876	2,143	2,726	3,904	8,163	12,540	18,171
Sør-Trøndelag	4,038	3,864	2,273	2,121	3,951	6,629	10,446	14,493
Nord-Trøndelag ...	4,868	3,098	1,989	3,484	4,712	8,607	11,336	15,552
Nordland	5,408	5,913	2,657	2,430	5,151	9,656	15,120	15,601
Troms	4,330	3,927	2,115	1,784	4,129	9,887	14,044	16,791
Finnmark	5,362	5,074	2,069	3,232	5,316	11,864	17,085	16,272
KVINNER								
HELE LANDET	5,729	3,281	2,497	7,073	4,383	6,222	9,027	12,122
Østfold	5,561	2,849	2,164	5,890	4,346	6,671	10,781	12,874
Akershus	5,250	3,105	2,064	6,905	4,364	6,025	8,214	11,439
Oslo	6,103	2,632	2,119	6,483	3,690	5,438	9,654	16,500
Hedmark	5,811	3,347	2,752	7,055	4,827	6,272	9,022	10,110
Oppland	5,609	2,630	2,852	6,881	4,902	6,553	8,555	8,305
Buskerud	5,975	3,119	2,578	7,575	5,066	6,235	9,026	11,238
Vestfold	5,468	2,851	2,545	6,494	4,728	6,776	7,857	10,549
Telemark	5,485	3,987	3,798	6,725	4,006	6,277	7,300	7,770
Aust-Agder	5,659	3,320	2,910	7,379	4,211	5,785	8,790	10,770
Vest-Agder	5,809	1,989	2,712	7,967	5,254	5,374	9,310	11,646
Rogaland	5,180	2,978	2,190	7,325	3,563	5,952	7,685	10,448
Hordaland	5,212	4,034	2,034	6,156	3,795	5,471	8,953	11,221
Sogn og Fjordane .	6,016	2,429	2,368	7,795	5,795	5,288	10,086	10,723
Møre og Romsdal ..	5,520	4,370	2,120	7,099	3,657	5,867	8,067	12,354
Sør-Trøndelag	5,415	3,003	2,228	7,007	3,702	6,176	7,917	12,582
Nord-Trøndelag ...	6,760	3,305	3,443	8,838	4,890	8,638	9,402	14,154
Nordland	6,939	5,024	3,449	8,653	5,557	7,213	10,873	11,197
Troms	6,249	3,025	2,684	7,523	4,962	7,344	10,497	16,813
Finnmark	7,007	3,182	2,885	9,456	6,236	9,469	11,191	8,152

Som grunnlag for denne tabellen har vi to filer;

A. Pasientdatafil
B. Middelfolkemengdefil

Pasientdatafilen har en record pr. utskrivning fra somatiske sykehus, mens middelfolkemengdefilen har en record pr. kommune * kjønn * alder med antall innbyggere.

Vårt første problem er at datene ligger på 2 filer. TAB kan som kjent ikke ha mer enn én input-fil. Vi får også et problem med at middelfolkemengdefilen ikke har samme aggregeringsnivå som tabellen vi skal lage har. Det trenger vi for å få regnet ut utskrivninger pr. 1000 innbyggere.

Det første problemet vil vi løse ved å hente inn innbyggertallene som en oppslagsfil (FYLD-fil). Da vil vi få pasientdatafilfilen som inputfil og middelfolkemendgefilen som oppslagsfil.

Problem 2 løser vi ved å lage 2 TAB-program som skal lage en oppslagsfil som er skreddersydd for tabellen vi skal lage. Hvorfor må vi lage 2 program for å få laget en oppslagsfil? Svaret på dette ligger i den ferdige tabellens utseende. Hvis vi ser litt nærmere på den ferdige tabellen, ser vi tabellen kan deles i to tabeller, en med totaltall for begge kjønn, og en med tall fordelt på kjønn. Tabellen bryter med det vanlige hierarkiet til TAB1. Det er kanskje ikke så lett å se med en gang, men faktum er at TAB ikke uten videre lager fylkesfordeling for begge kjønn. For å få til dette må vi bruke TAEL-instruksjonen. Trikset vi bruker er å simulere at 'Begge kjønn' skal ligge på samme nivå som 'Menn' og 'Kvinner' (selv om 'Begge kjønn' egentlig ligger et nivå høyere). Dette gjør det enkelt å få med fylkesfordeling også for 'Begge kjønn'. Vi skal altså bruke TAEL-instruksjonen for å få 'Begge kjønn' ned på samme nivå samme 'Menn' og 'Kvinner'. Som sagt så skaper altså denne lille finessen ved tabellen enkelte problemer (eller skal vi si merarbeid) for oss. I alle fall må vi først lage en oppslagsfil for kjønn * fylke. I denne sørger vi også for å få med sumtall for hele landet for menn og for kvinner. Vi kaller 'Hele landet' fylke 00, og den koden gir vi 'Hele landet' i den første oppslagfilen. Programmet som lager den første oppslagsfilen finner du på neste side.

```
//O414KRLÆ JOB (8019),'Oppslagsfil',MSGLEVEL=(2,0),
//         CLASS=A,MSGCLASS=X,NOTIFY=O414KRL,TIME=2
//*************************************************************
//* PROGRAM SOM LAGER OPPSLAGSFIL FOR KJØNN * FYLKE.          *
//* HELE LANDET FOR FYLKESKODE 00                             *
//* SKRIVER RESULTATET TIL EN TK-FIL FOR Å SJEKKE TALLENE     *
//*************************************************************
//TAB1     EXEC TAB1B
//INPUT    DD DSN=P6216.S0137.G185A1A1.G8800.V00,DISP=OLD
//TABELL   DD DSN=TK414.S0137.KRL.INNB1,DISP=OLD
         START TYPE=P
KOMMNR   FIELD (1,4,X)
FYLKE    FIELD (1,2,X)
KJØNN    FIELD (5,1,X)
ALDER    FIELD (6,3,X)
ANTALL   FIELD (10,7,X)  NB! Siste siffer er desimal!!
         SØJLE S1,,ANTALL
         SØJLE S2:S8,(ALDER = 0-9:10-19:20-39:
                              40-59:60-69:70-79:),ANTALL
         FORSP KJØNN,NOSUM
         FORSP FYLKE
TABELL   FILE  LRECL=67,BLKSIZE=6700,SUM
         MOVE  (1,1,X),KJØNN
         MOVE  (2,2,X),FYLKE
         MOVE  (4,8,X)*8,S1
         IF    (TALLY0 = 1)      HELE LANDET SKAL FÅ FYLKE = 00
         MOVE  (2,2,X),'00'
         ENDIF
```

Dette programmet vil gi oss en fil som ser slik ut:

```
10020821180026757100312372006547000046226500200432501340795005069 80
10101167595001389200017763500350400002719550011908000080945000286 60
10202033060002674600031370500652890004980400017721000093070000306 85
10302129600002313400021641500766475004821900022336500153335000564 80
: (vi hopper over noen fylker her)
12000381470000487650006236000128960000855550003157000018295000059 65
20021273690025449200298437506185580045080050222413001826645010000 35
20101207615001326650017038000333210002702700013131500113525000562 50
20202051345002505750029990500640445004961150018527000119955000590 80
20302419720002203200021238500786620004850600028318500266180001659 70
: (vi hopper over noen fylker her)
22000359065000465850005847500114340000727550003221500024510000101 85
```

Vi har nå laget en befolkningsfil med kjønn og fylke som kjennemerker. Denne filen har også med totalsummer for hele landet for hvert kjønn. Disse har fått fylkeskode 00. Tallene er med en desimal. Hvert kjønn * fylke har antall personer fordelt etter de aldersgrupper som tabellen vi skal lage har. Med dette programmet har vi funnet innbyggertall for hvert kjønn og alle fylker (+ 'Hele landet') som skal brukes i tabellen vår. Men vi må også finne innbyggertall for sum 'Begge kjønn' for hvert fylke (dette er også en del av den ferdige tabellen vår). Vi tar da utgangspunkt i filen vi nettopp har laget (ingen vits i å bruke den opprinnelige (store) befolkningsfilen når alle data vi trenger ligger på den vi nettopp laget). Programmet blir nesten likt det forrige:

```
//O414KRLØ JOB (8019),'Oppslagsfil for fylke',MSGLEVEL=(2,0),
//         CLASS=A,MSGCLASS=X,NOTIFY=O414KRL
//*****************************************************
//* PROGRAM SOM LAGER OPPSLAGSFIL FOR BEGGE KJØNN * FYLKE  *
//* HELE LANDET FÅR FYLKESKODE 00, BEGGE KJØNN FÅR KODE 0  *
//* SKRIVER RESULTATET TIL EN TK-FIL FOR Å SJEKKE TALLENE  *
//*****************************************************
//TAB1X    EXEC TAB1B
//INPUT    DD DSN=TK414.S0137.KRL.INNB1,DISP=OLD
//TABELL   DD DSN=TK414.S0137.KRL.INNB2,DISP=OLD
         START TYPE=P
KJØNN    FIELD (1,1,X)
FYLKE    FIELD (2,2,X)
ANTALL   FIELD (4,8,X)*8
         SØJLE S1:S8,,ANTALL(1)
         FORSP FYLKE
TABELL   FILE  LRECL=67,BLKSIZE=6700
         MOVE  (1,1,X),'0'   BEGGE KJØNN FÅR KODE 0
         MOVE  (2,2,X),FYLKE
         MOVE  (4,8,X)*8,S1
```

Etter å ha kjørt dette programmet har vi en fil som ser slik ut:

```
000420948700522063006108095127325800913065504228455031674400150701 5
001023752100027158500348015006836100054222500250395001944700008491 0
002040844050051803500613610012933350099415500362480002130250008976 5
003045493200045166000428800015530950096725000506550004195150022245 0
: (vi hopper over noen fylker her)
020007405350009535000120835002433000015831000063785000428050001615 0
```

Filen inneholder en record pr. fylke. Tallene er sumtall for 'Begge kjønn', og koden for 'Begge kjønn' er 0. Denne filen inneholder innbyggertall for den første tredelen av tabellen vi skal lage.

Nå gjenstår det bare å lage tabellen. Vi skal bruke de to filene vi har laget som en oppslagsfil. Denne skal vi slå opp i når vi skal regne om de absolutte tallene til å være ratetall pr. 1000 innbyggere. Vår oppslagsfil skal være sortert på kjønn * fylke. Dette fordi kjønn og fylke er de nøklene (kjennemerkene) vi skal bruke i oppslaget (oppslagsfilen må være sortert etter oppslagsnøklene). Dette får vi til ved å konkatenere (lese sammen) de to filene våre med den vi lagde sist som den første. Programmet kommer på neste side:

```
//O414KRLØ JOB (8019),'Tabell 1',MSGLEVEL=(2,0),
//         CLASS=A,MSGCLASS=X,NOTIFY=O414KRL,TIME=2
//TAB1XX   EXEC TAB1B,TIME=2
//INPUT    DD DSN=P4216.S0137.G184A3A2.G8800.V00,DISP=SHR
//INNB     DD DSN=TK414.S0137.KRL.INNB2,DISP=OLD
//         DD DSN=TK414.S0137.KRL.INNB1,DISP=OLD
//TABELL   DD SYSOUT=*,COPIES=1
//KJØNN    DD *
0BEGGE KJØNN
0
0HELE LANDET
1MENN
1
1HELE LANDET
2KVINNER
2
2HELE LANDET
//FYLKTXT  DD DSN=TAB.DIV(FYLKTXTS),DISP=SHR
//SYSIN    DD *
         START TYPE=P,NULDIV=(0,0)
FYLKE    FIELD (23,2,X)
KJØNN    FIELD (27,1,X)
ALDER    FIELD (28,3,X)
AAR      FIELD (50,2,X)
INNBYG   GROUP (64)
PERS     WORK  (8,X)*8
KJFY     WORK  (3,X)
AARGANG  WORK  (2,X)
INNB     FYLD  KEY=(1,3,X),DATA=(4,64,X),MAX=60
SELECT   INPUT (KJØNN = 1-2 AND FYLKE = 01-20)
         SET   AARGANG=AAR
         TAEL
         SET   KJØNN=0
         TAEL
         STOP
         SØJLE S1
         SØJLE S2:S8,(ALDER = 0-9:10-19:20-39:40-59:60-69:70-79:)
         FORSP KJØNN,TXTFIL=KJØNN,TXTKEY=(1,1,X),TXT=(2,15),
               NEWLINE,PRINT=1,NOSUM
         FORSP
FYLKE,TXTFIL=FYLKTXT,TXTKEY=(1,2,X),TXT=(3,25),PRINT=3
         IF    (TALLY0 = 1)
         SET   FYLKE=0
         ENDIF
         SET   KJFY=KJØNN*100+FYLKE
         FIND  ARG=KJFY,DATA=INNBYG,DDNAVN=INNB
         SREGN S9=S1*10000000/PERS(1)
         SREGN S10=S2*10000000/PERS(2)
         SREGN S11=S3*10000000/PERS(3)
         SREGN S12=S4*10000000/PERS(4)
         SREGN S13=S5*10000000/PERS(5)
         SREGN S14=S6*10000000/PERS(6)
         SREGN S15=S7*10000000/PERS(7)
         SREGN S16=S8*10000000/PERS(8)
```

```
TABELL   FILE  PRINT,MAXLIN=84,OVERFLOW
         HDR   (1-7),(1-67)
Tabell 1. Utskrivinger fra somatiske sykehus etter alder, kjønn og
          bostedsfylke. Pr. 1000 innbyggere. 19XX
-------------------------------------------------------------------
                                                          Alder
Kjønn og fylke          I alt
-------------------------------------
                                    0-9   10-19   20-39   40-59
60-6
-------------------------------------------------------------------
         HDR   (1-7),(68-120)


--------------------


---------------------
9   70-79      80-
---------------------
         HDR   2,48,AARGANG
         MOVE  (22,8,E-)*8,S9,ZZZ9,999
*        MOVE  (120,3,X),KJFY         * Hva er KJFY?
*        LIN
*        MOVE  (5,9,X),'Pasienter'    * Absolutte tall, pasienter
*        MOVE  (22,8,E-)*8,S1,ZZZZZZZZ
*        LIN
*        MOVE  (5,10,X),'Innbyggere' *  Absolutte tall, innbyggere
*        MOVE  (22,8,E-)*8,PERS(1)/10,ZZZZZZZZ
         IF    (FYLKE = 20 AND KJØNN = 2 AND TALLY0 = 0)
         LIN
         MOVE  (1,1,X)*88,'-'
         ENDIF
```

Vi bruker FYLD-instruksjonen til å lage en oppslagsfil. I FYLD sier vi i hvilke posisjoner nøkkelfeltet og oppslagsdataene står i. For å gjøre oppslaget bruker vi FIND-instruksjonen. For å få få gjort et oppslag må vi ha en oppslagsnøkkel, vi må vite hvilken kode vi skal søke etter i oppslagsfilen. Siden vår oppslagsnøkkel består av to felt fra innfilen, er vi nødt til å legge dataene fra disse to feltene inn i et arbeidsfelt. Dette arbeidsfeltet bruker vi så til oppslaget vårt.

Legg merke til at vi gjør oppslaget i SREGN. Det er først her vi har fått laget en matrise som har de absolutte tallene vi skal regne rater ut fra. Før vi skal gjøre oppslag mot data på høyeste nivå (sum 'Hele landet', må vi sette FYLKE til 0. Dette fordi vi har valgt kode 0 som fylkeskode for 'Hele landet'. Problemet er at på dette nivået vil fylkeskoden "henge igjen" fra det siste fylket på nivået under (Finnmark, kode 20). Hvis vi gjør oppslag på oppslagsfilen vår uten å rette opp fylkeskoden, vil vi få hentet tall for Finnmark der vi skulle ha tall for 'Hele landet', hvilket vi gi feil ratetall. Grunnen til at fylkeskoden "henger igjen" er at vanligvis vil det ikke ha noen mening å gjøre operasjoner på fylke når vi er på et nivå over fylke, derfor er det ikke noe innebygget i TAB-programmet som nullstiller fylkeskoden her. Det betyr at vi må endre koden selv. Her er det altså snakk om å få hentet ut innbyggertall for 'Hele landet', som jo er på et nivå over Fylke, derfor gir vi Fylke koden 00, som det har i oppslagsfilen vår.

For å få med fylkesfordeling på sumtallene for 'Begge kjønn', har vi brukt instruksjonen TAEL to ganger; den første sørger for opptelling til ett av kjønnene, den andre til sum 'Begge kjønn'. Sum 'Begge kjønn' får vi telt opp ved å gi kjønn kode 0 etter første TAEL. Vi vil da få telt opp med kjønn lik 0 hver gang, i tillegg til opptelling for det kjønnet den leste recorden har, hvilket gir oss sumtall for 'Begge kjønn'. Dette kalles å legge sumtall på samme nivå som de andre tallene. At de er på samme nivå (for tabellprogrammet) ser vi av tekstkatalogen til kjønn, der er 'Begge kjønn' på lik linje med 'Menn' og 'Kvinner'. Dette er trikset for å få undersummer (her fylkestall) også for sumnivåer (her 'Begge kjønn').

FYLD og FIND kan egentlig bare brukes til å gjøre oppslag med en nøkkel, og vi kan bare hente ut et felt fra oppslagsfilen. Vi har sett hvordan vi kan få til å bruke mer enn 1 nøkkel. For å hente ut mer enn et felt, bruker vi GROUP-instruksjonen. En GROUP definerer en gruppe med arbeidsfelter (WORK). Når vi bruker GROUP, vil vi hente dataene fra oppslagsfilen og legge resultatet i GROUP-feltet. Når dette er gjort, vil arbeidsfeltene under gruppen ha fått verdier som vi kan bruke hver for seg. Dette skulle gå ganske klart fram i programmet over.

De siste fire programlinjene lager den stiplede linjen nederst på tabellen.

Husk på å bruke TEST-instruksjonen når du programmerer, selv om det ikke alltid er like enkelt å få sjekket dataene når du gjør det. For å sjekke tallene i dette programmet kan vi for eksempel kjøre tabellen for Aust-Agder og Finnmark (de to fylkene med færrest innbyggere).

Når du skal lage ratetabeller, lønner det seg nesten alltid å få skrevet ut de absolutte tallene under programmeringsfasen. Dette kan vi gjøre ved å ta med kommentarlinjene nederst i det siste programmet. Disse sørger for å skrive ut de absolutte tallene i tillegg til ratene. Det gir deg gode muligheter til å sjekke tallene. Når tallene er sjekket og godkjent, fjerner du (kommenterer vekk) programlinjene som skriver ut de absolutte tallene.

Når vi har fått til alt dette, kan vi snekre sammen de tre jobbene, slik at det kan kjøres som en jobb. Istedenfor å legge oppslagsfilene våre på TK-filer bruker vi midlertidige filer, ellers blir det å slå sammen de tre jobb-oppsettene. Når programmene nå skal kjøres, vil vi også sørge for å legge den ferdige tabellen ut på en fil. Slik blir JCL-en når vi slår sammen de tre programmene:

```
//0414KRLØ JOB (8019),'Tabell 1 (hele jobben)',MSGLEVEL=(2,0),
//         CLASS=A,MSGCLASS=X,NOTIFY=0414KRL,TIME=2
//*****************************************************************
//* PROGRAM SOM LAGER TALL PR. 1000 INNBYGGERE VED Å HENTE
//*  INNBYGGERDATAENE FRA TO FYLD-FILER.
//*  FYLD-FILENE LAGES FØRST. EN FORDELT PÅ KJØNN * FYLKE OG
//*  EN FOR BEGGE KJØNN. DISSE VIL TILSAMMEN DANNE FYLDFILEN VI
//*  SKAL BRUKE I DET 3. PROGRAMMET I DENNE JOBBEN.
//*****************************************************************
//TAB1     EXEC TAB1B
//INPUT    DD DSN=P6216.S0137.G185A1A1.G8800.V01,DISP=OLD
//TABELL   DD DSN=&&INNB1,
//            DISP=(NEW,PASS,DELETE),
//            SPACE=(6700,(1,1),RLSE),UNIT=WORK,
//            DCB=(LRECL=67,RECFM=FB,DSORG=PS,BLKSIZE=6700)
:
Her kommer det første programmet
:
:
//TAB1X    EXEC TAB1B
//INPUT    DD DSN=&&INNB1,DISP=(OLD,PASS)
//TABELL   DD DSN=&&INNB2,
//            DISP=(NEW,PASS,DELETE),
//            SPACE=(6700,(1,1),RLSE),UNIT=WORK,
//            DCB=(LRECL=67,RECFM=FB,DSORG=PS,BLKSIZE=6700)
:
Her kommer det andre programmet
:
:
//TAB1XX   EXEC TAB1B,TIME=2
//INPUT    DD DSN=P4216.S0137.G184A3A2.G8800.V00,DISP=SHR
//INNB     DD DSN=&&INNB2,DISP=(OLD,DELETE)
//         DD DSN=&&INNB1,DISP=(OLD,DELETE)
//TABELL   DD DSN=TK414.S8019.KRL.TABELLER(TAB1),DISP=OLD
//KJØNN    DD *
0BEGGE KJØNN
0
0HELE LANDET
1MENN
1
1HELE LANDET
2KVINNER
2
2HELE LANDET
//FYLKTXT  DD DSN=TAB.DIV(FYLKTXTS),DISP=SHR
//SYSIN    DD *
:
Her kommer det siste programmet
:
:
```

VEDLEGG V: SKJEBNESVANGER BRUK AV ARBEIDSFELT

Vi skal her vise hvor galt det kan gå hvis arbeidsfelt (WORK) blir brukt uten omtanke for hvordan de fungerer. La oss tenke oss at vi er interessert i å lage en tabell over kommuner etter befolkningsstørrelse for kommuner med 10000 innbyggere eller mer. Til dette trenger vi en fil som har en record pr. kommune og som inneholder et felt med antall innbyggere i kommunen. Når vi har ordnet oss en slik fil, kan vi begynne programmeringen.

Vi må da lage et program hvor vi bruker et arbeidsfelt til å kode om antall innbyggere til befolkningsstørrelse. Vi kaller feltet ANT og lager en CASE-test i programmet vårt. Programmet vi lager ser da slik ut:

```
//O414KRLØ JOB (8019),'Feil bruk av WORK!',MSGLEVEL=(2,0),
//           CLASS=A,MSGCLASS=X,NOTIFY=O414KRL
//WORK2    EXEC TAB1A
//INPUT    DD DSN=PP414.S8019.TABKURS.DATAFIL2,DISP=OLD
//ANTTXT   DD *
1 10000-19999
2 20000-29999
3 30000-49999
4 50000-99999
5100000 OG OVER
//TABELL   DD SYSOUT=*
         START TYPE=P
BEFOLKN  FIELD (9,6,X)
ANT      WORK  (1,X),SPACES
         CASE  (BEFOLKN = 10000-19999)
         SET   ANT=1
         CASE  (BEFOLKN = 20000-29999)
         SET   ANT=2
         CASE  (BEFOLKN = 30000-49999)
         SET   ANT=3
         CASE  (BEFOLKN = 50000-99999)
         SET   ANT=4
         CASE  (BEFOLKN = 100000-999999)
         SET   ANT=5
         ENDCASE
         SØJLE S1
         FORSP ANT,TXTFIL=ANTTXT,TXTKEY=(1,1,X),TXT=(2,15),PRINT=1
TABELL   FILE  PRINT,MAXLIN=46,OVERFLOW
         HDR   (1-4),(1-55)
ANTALL KOMMUNER ETTER BEFOLKNINGSSTØRRELSE.

BEFOLKNINGS-            ANTALL
STØRRELSE              KOMMUNER
         MOVE  (23,7,Z-),S1
```

Dette programmet vil, for 1985, gi oss denne tabellen!

ANTALL KOMMUNER ETTER BEFOLKNINGSSTØRRELSE.

BEFOLKNINGS-STØRRELSE	ANTALL KOMMUNER
10000-19999	334
20000-29999	46
30000-49999	45
50000-99999	4
100000 OG OVER	25
TOTAL	454

Vi ser at vi har fått 25 kommuner med mer enn 100 000 innbyggere. Her må det være noe feil. Programmet ser jo riktig ut, vi har kodet om og fått tall i alle befolkningsstørrelser, men tallene ser helt gale ut! Hva har vi gjort galt?

For å finne feilen må vi vite hvordan arbeidsfelt brukes. Et arbeidsfelt beholder sin verdi helt til den får en ny. Dette høres jo logisk ut, og det stemmer overens med tankegangen vår. Men hvis vi nå prøver å tenke oss hvordan programmet arbeider, record for record, vil vi se hvordan vi fikk feil tall.

Filen vår starter med kommune 0101 (Halden) med innbyggertall 25876. Den gir i vår omkoding feltet ANT = 2. Denne telles så opp. Deretter leses neste record, 0102 (Sarpsborg). Befolkningen i Sarpsborg er 12069, hvilket gir ANT = 1. Så telles denne opp. Vi går videre med 0111 (Hvaler) med 2889 innbyggere. Vår CASE-test slår ikke til idet hele tatt denne gangen. Det betyr at ANT = 1 fortsatt. Så telles det opp. Hvaler blir nå telt opp som kommune med mellom 10000 og 20000 innbyggere. Her ser vi altså hvordan feilen oppsto. Når en test som gir et arbeidsfelt verdi ikke slår til, vil arbeidsfeltet beholde den verdi den hadde fra før. Når arbeidsfeltet brukes på denne måten, vil tabellen vår bli helt gal. De kommunene som ikke har innbyggertall som er med i CASE-testen vår, vil plasseres i samme befolkningsstørrelse som kommunen i recorden foran!

Dette må du være klar over når du bruker arbeidsfelt. Når du er klar over hva som skjer, er det veldig enkelt å ta forholdsregler, slik at tabellene dine blir riktige, også når du bruker arbeidsfelt.

Når du koder om data ved hjelp av arbeidsfelt, må du sørge for at omkodingen skjer for hver eneste record, uansett hvilken verdi det aktuelle feltet måtte ha. I CASE-tester gjøres det enkelt ved å legge inn en siste CASE-instruksjon i testen. Denne gir du ingen betingelse, det betyr at hvis ingen av de foranstående CASE-instruksjoner har slått til, vil denne slå til. Vi lager altså en CASE-instruksjon som lager en eventuell restgruppe. Dette gjør vi for vårt programeksempel, og programmet viser vi på neste side.

```
//O414KRLØ JOB (8019),'Riktig bruk av WORK!',MSGLEVEL=(2,0),
//           CLASS=A,MSGCLASS=X,NOTIFY=O414KRL
//WORK2    EXEC TAB1A
//INPUT    DD DSN=PP414.S8019.TABKURS.DATAFIL2,DISP=OLD
//ANTTXT   DD *
1 10000-19999
2 20000-29999
3 30000-49999
4 50000-99999
5100000 OG OVER
//TABELL   DD SYSOUT=*
         START TYPE=P
BEFOLKN  FIELD (9,6,X)
ANT      WORK  (1,X),SPACES
         CASE  (BEFOLKN = 10000-19999)
         SET   ANT=1
         CASE  (BEFOLKN = 20000-29999)
         SET   ANT=2
         CASE  (BEFOLKN = 30000-49999)
         SET   ANT=3
         CASE  (BEFOLKN = 50000-99999)
         SET   ANT=4
         CASE  (BEFOLKN = 100000-999999)
         SET   ANT=5
         CASE                              <=== Restgruppe
         SET   ANT=0
         ENDCASE
         SØJLE S1
         FORSP ANT,TXTFIL=ANTTXT,TXTKEY=(1,1,X),TXT=(2,15),PRINT=1
TABELL   FILE  PRINT,MAXLIN=46,OVERFLOW
         HDR   (1-4),(1-55)
ANTALL KOMMUNER ETTER BEFOLKNINGSSTØRRELSE.

BEFOLKNINGS-             ANTALL
STØRRELSE               KOMMUNER
         MOVE  (23,7,Z-),S1
```

Når vi nå kjører vårt lille program ser vi at tallene er blitt riktige (neste side). Vi ser også at vi har fått endel kommuner i restgruppen vår. Denne består teoretisk av de kommuner som har mindre enn 10000 innbyggere og de som har flere enn 999999. Siden vi i Norges land ikke har noen kommuner med flere enn 999999 innbyggere, skal restgruppen denne gang være de kommuner som har mindre enn 10000 innbyggere. Når det gjelder CASE-testen for 100000-999999 innbyggere ser vi at den ikke stemmer helt overens med teksten til denne gruppen. Det bør den gjøre, og testen vil da se slik ut:

```
         CASE  (BEFOLKN >= 100000)
         SET   ANT=5
```

Restgruppen vår har ikke noen tekst i tekstkatalogen, derfor skrives det ut stjerner. Vi får denne tabellen:

ANTALL KOMMUNER ETTER BEFOLKNINGSSTØRRELSE

BEFOLKNINGS-STØRRELSE	ANTALL KOMMUNER
0**********	356
10000-19999	65
20000-29999	15
30000-49999	11
50000-99999	4
100000 OG OVER	3
TOTAL	454

Dette er et tenkt eksempel, lagd for å illustrere et problem. Skulle vi laget denne tabellen i virkeligheten, ville vi nok hatt med kommuner med mindre enn 10000 innbyggere som en egen gruppe (og hvis vi ikke var interessert i disse kommunene, kunne vi ha fjernet dem med en SELECT-instruksjon). Men det ville allikevel være lurt å ta med en restgruppe for å være på den sikre siden. Det er tross alt viktig at tallene våre blir riktige.

Tips: Husk **ALLTID** å ha med en restgruppe når du koder om dataene dine.

Når du bruker IF, husk også å ha med en ELSE.

Bruker du CASE, husk en siste CASE-instruksjon uten betingelser (lager restgruppe)

VEDLEGG VI: SPØRSMÅL DU BØR BESVARE FØR DU LAGER ET TAB-PROGRAM

- Hvordan skal tabellen se ut?
- Hvordan ser innfilen ut?
- Hvor mange (cirka) records er det innfilen?
- Brukes alle records på innfilen til å lage tabellen?
- Finnes alle opplysninger vi trenger for å lage tabellen på innfilen? og hvis ikke, kan de avledes fra opplysninger i innfilen?
- Kan jeg lage tabellen i TAB1A eller TAB1B?
- Hvilke forspaltekriterier har jeg?
- Hvor ofte skal tabellen kjøres?
- Skal tabellen publiseres?
- Har jeg et TAB-program fra før som lager en forholdsvis lik tabell?
- Er eventuelle tekstkataloger laget tidligere?
- Har jeg en fasit for tallene i tabellen?
  og hvis ikke, kan jeg få kontrollert tallene allikevel?

Ut fra de svarene du har gitt, bør du greie å danne deg et bilde av hvordan programmet skal se ut, og hvilken framgangsmåte du skal bruke. Du vil også få en viss formening om hvor eventuelle problemer kan oppstå.

VEDLEGG VII: TEGNENES SORTERINGSREKKEFØLGE

Når en tabell sorteres etter forspaltekriteriene, vil TAB bruke SYNCSORT til dette. Tegnene er plassert i en innbyrdes rekkefølge. Det er denne rekkefølgen SYNCSORT bruker når den skal sortere filer. Tabellen under (delt i to spalter) viser rekkefølgen (det laveste tegn står først). Som vi ser er æ, ø, å, Æ, Ø og Å plassert på feil sted for oss. Vi ser også at blank er lavest, små bokstaver lavere enn store og tall er høyere enn bokstaver. Når vi lager forspaltetekstfiler er det viktig at kodene er sortert stigende etter tabellen under her.

	(blank)	v	
#		w	
.		x	
<		y	
(		z	
+		æ	
!		A	
&		B	
¤	(soltegn)	C	
Å		D	
*		E	
)		F	
;		G	
^		H	
-	(strek)	I	
/		å	
ø		J	
,	(komma)	K	
%		L	
_	(understrek)	M	
>		N	
?		O	
`		P	
:		Q	
Æ		R	
Ø		\	
'	(fnutt)	S	
=		T	
"		U	
a		V	
b		W	
c		X	
d		Y	
e		Z	
f		0	
g		1	
h		2	
i		3	
j		4	
k		5	
l		6	
m		7	
n		8	
o		9	
p			
q			
r			
ü			
s			
t			
u			

VEDLEGG VIII: IBM'S OVERPUNCH FOR LAGRING AV NEGATIVE TALL

For å slippe å bruke én posisjon av et felt til oppbevaring av fortegn, bruker IBM i stormaskinverdenen noe som kalles overpunch. Istedenfor å ha med fortegnet, endrer de heller det siste tallet i feltet. Dette gjøres om til en bokstav. Denne bokstaven forteller både hvilket tall det bokstaven står for, og hvilket fortegn tallet skal ha. For eksempel vil tallet 1162 bli lagret som 116K, der K betyr 2 og at hele tallet er negativt.

Her følger en oversikt over hvilke bokstaver som kan brukes og hvilket tall og fortegn de representerer.

ü	+1
s	+2
t	+3
u	+4
v	+5
w	+6
x	+7
y	+8
z	+9
æ	+0
A	+1
B	+2
C	+3
D	+4
E	+5
F	+6
G	+7
H	+8
I	+9
å	-0
J	-1
K	-2
L	-3
M	-4
N	-5
O	-6
P	-7
Q	-8
R	-9
\	+0
S	+2
T	+3
U	+4
V	+5
W	+6
X	+7
Y	+8
Z	+9

VEDLEGG IX: PROGRAM MED AUTOMATISK BEREGNEDE RESTGRUPPER

Når vi lager tabeller med Tab1, får vi som kjent én linje for hver verdi feltene i forspalten har. Dette gir oss ofte ganske store tabeller, ofte så store at de ikke egner seg for publisering. Vi kan isteden slå sammen linjene som har minst tall til en rest-gruppe. Men som oftest er det vanskelig å si hvem som bør tilhøre restgruppen, den vil også kunne endre seg når en tabell skal kjøres på en ny datafil. Ved å automatisere utvelgelsen til rest-gruppene, sparer vi en del forarbeid med å finne restgruppene hver gang programmet skal kjøres. Anta at du skal lage denne tabellen:

Tabell 2. Utenlandsfødte etter alder. 1. januar 1991

		Alder						
Fødeland	I alt	0-9	10-19	20-29	30-39	40-49	50-59	60-69
I alt	143304	19862	17628	35714	33815	19466	8904	4667
Europa i alt	72858	7471	8713	14911	17542	13206	6437	3167
Danmark	17198	1228	2038	3959	3560	2763	2071	1189
Finland	3051	267	247	430	890	792	292	105
Island	2202	372	308	490	589	289	105	40
Sverige	11672	909	1179	2104	2845	2337	1055	750
Belgia	355	37	34	58	113	71	22	12
Bulgaria	298	17	24	71	134	40	8	3
Frankrike	1768	189	197	258	537	399	127	42
Irland	379	29	27	81	134	79	23	5
Italia	745	37	65	138	205	145	89	51
Jugoslavia	4242	1065	504	1223	881	361	137	53
Nederland	2552	247	314	424	679	549	200	91
Polen	2854	442	353	653	868	365	86	57
Portugal	436	34	59	118	104	65	44	9
Sovjetunionen	373	30	57	75	112	61	27	7
Spania	880	34	105	189	243	207	78	19
Storbritannia	11766	904	1465	1789	3061	2825	1157	394
Sveits	767	38	80	138	177	165	105	59
Tyrkia	5523	1264	1026	1587	909	483	183	51
Tyskland	4270	208	505	764	1012	955	524	180
Østerrike	526	28	47	101	131	120	67	27
Europa ellers	1001	92	79	261	358	135	37	23
Afrika i alt	9400	1412	1008	3715	2528	523	151	43
Algerie	356	31	11	161	116	24	10	2
Etiopia	1447	177	200	719	279	58	9	3
Gambia	593	55	24	261	219	34	0	0
Ghana	857	100	39	323	359	32	3	1
Marokko	2163	382	333	759	434	153	83	17
Nigeria	363	67	14	116	127	36	2	1
Somalia	1706	376	242	649	356	64	11	7
Tunisia	328	37	10	176	88	16	1	0
Afrika ellers	1587	187	135	551	550	106	32	12
Asia i alt	42092	8407	5792	13045	9398	3534	1192	496
Afghanistan	322	53	57	119	59	20	12	2
Bangladesh	462	73	29	181	154	18	6	1
Filippinene	2304	305	194	818	722	170	43	34
India	3459	646	380	955	959	328	102	58
Irak	898	207	101	231	263	70	16	8
Iran	5942	1143	682	2154	1424	357	126	45
Israel	277	20	12	114	105	14	4	3
Japan	433	53	43	47	106	143	33	7
Kina	1469	155	92	428	509	204	41	23
Libanon	705	149	42	386	103	15	3	5
Pakistan	11442	3088	1976	2419	2022	1334	434	116
Sør-Korea	290	118	38	36	72	23	1	0
Sri Lanka	5247	714	361	2677	1165	256	46	16
Thailand	1127	113	177	378	363	79	14	3
Vietnam	6898	1402	1530	1919	1117	408	287	164
Asia ellers	817	168	78	183	255	95	24	11

Tabellen viser netto innvandring til Norge. Som du ser er det en restgruppe for hver verdensdel. Innvandringen til Norge vil variere over tid, det betyr at restgruppenes innhold også vil endre seg. For å slippe å dele inn restgruppene hver gang, vil vi nå lage et program som gjør det for oss.

Det første vi må gjøre er å bestemme oss for hvilket kriterium som skal gjelde for at et land skal havne i restgruppen. La oss anta at alle land som har netto innvandring mindre enn 277 personer, skal havne i restgruppen. Denne grenseverdien skal vi ha som en parameter til programmet, slik at den kan endres uten at vi behøver å forandre programmet.

Vi må først lage et program som kjører gjennom datafilen og summerer innvandring fordelt på land. I dette programmet unnlater vi å skrive ut de land som har en innvandring under 277 personer (grenseverdien). Dette gjør vi med instruksjonene IF og CANCEL.

Denne filen som lages i det første programmet, skal brukes som en oppslagsfil i det andre. Programmet som lager oppslagsfilen ser slik ut:

```
         START TYPE=P
ANTALL   FIELD (15,6,X)
STAT     FIELD (21,3,X)
GRENSE   WORK  (4,X),PARM(1,4)
SELECT   INPUT (STAT NE 000)
         SØJLE S1,,ANTALL
         FORSP STAT,NOSUM
         IF    (S1 < GRENSE)
         CANCEL
         ENDIF
MATRISE  FILE  LRECL=9,BLKSIZE=23472
         MOVE  (1,3,Z),STAT
         MOVE  (4,6,Z),S1
```

Dette ble jo et veldig lite program. Fra innfilen fjernes alle norske statsborgere (STAT NE 000). Det lages en søjle, S1, som inneholder netto innvandring, og vi har land (STAT) som forspaltekriterium. Utfilen blir en datafil som inneholder landkode og antall personer innvandret. Denne filen skal altså brukes som en oppslagsfil i programmet som skal lage den trykkeklare tabellen.

Foruten problematikken med restgruppene, har vi noen andre utfordringer før tabellen blir perfekt. Det gjelder restgruppenes plassering sist under hver verdensdel, de nordiske land først i Europa-delen og til slutt å sortere landene etter tekst istedenfor etter nummer.

Men la oss først se hvordan riktige land havner i riktig restgruppe. Vi må gi hver restgruppe en unik landkode. Hver restgruppes landkode må begynne med det tallet verdensdelen har som sin kode. Dette ordner vi med en CASE-test (den andre i programmet, den første brukes for å gi landene i Afrika som begynner med 3, verdensdelkode 2, samt å gi de som har 5 i første siffer i landkoden verdensdel 4 (Asia)). Men før det må vi ha sjekket om det aktuelle landet skal være i restgruppen eller ikke. Det gjør vi ved å slå opp i oppslagsfilen (den første FIND-instruksjonen). Hvis landet finnes i oppslagsfilen skal det **IKKE** være i restgruppen, er landet ikke funnet (FUNNET = 0) skal det være i restgruppen. Nå skulle restgruppene være laget.

For å få sortert tabellen etter landtekst istedenfor etter landkode, må vi bruke landkatalogen som en oppslagsfil. Fra oppslagsfilen henter vi landteksten til et arbeidsfelt (den andre FIND-instruksjonen). Dette arbeidsfeltet (NAVN) bruker vi som felt i forspalten. Dette vil sørge for at tabellen sorteres etter landteksten.

Når vi nå skal plassere restgruppene nederst innenfor hver verdensdel, gjør vi det ved først å gi arbeidsfeltet NAVN en verdi som garantert kommer sist i sorteringsrekkefølgen (f.eks. "ZZropa ellers" for Europas restgruppe). Vi gjør dette med en CASE-test (den tredje). De landkodene som ikke er i landkatalogen, får tekster som begynner med to Z'er. Det vil sørge for at de havner nederst innen hver verdensdel. Problemet med dette er at det ikke ser så pent ut i tabellen (ZZropa i stedenfor Europa). Dette ordner vi i en CASE-test ved utflytting (MOVE) av data til tabellen.

Hvis de nordiske land skal stå først innenfor Europa, gjør vi det på samme måte som for restgruppene, bortsett fra at disse landene må få tekster (NAVN) som starter med tegn som er lavere en de store bokstavene, f. eks. små bokstaver (se CASE-test 4 og 5). Programmet vil bli slik:

```
          START TYPE=P
ALDER     FIELD (7,3,X)
ANTALL    FIELD (15,6,X)
STAT      FIELD (21,3,X)
STAT1     FIELD (21,1,X)
AAR       WORK  (4,X),PARM(1,4)
VDEL      WORK  (1,X)
IALT      WORK  (6,X),0
NAVN      WORK  (22,X)
FUNNET    WORK  (1,X)
FUNNET2   WORK  (1,X)
GRENSFIL  FYLD  KEY=(1,3,X),DATA=(4,6,X),MAX=60
LANDTXT   FYLD  KEY=(1,3,X),DATA=(6,22,X),MAX=250
SELECT    INPUT (STAT NE 000)
          FIND  ARG=STAT,DATA=IALT,DDNAVN=GRENSFIL,SW=FUNNET
          CASE  (STAT1 = 1,2,4,6,7,8,9)
          SET   VDEL=STAT1
          CASE  (STAT1 = 3)
          SET   VDEL=2
          CASE  (STAT1 = 5)
          SET   VDEL=4
          ENDCASE
          IF    (FUNNET = 0)
          CASE  (VDEL = 1)     * Restgruppe Europa
          SET   STAT=194
          CASE  (VDEL = 2)     * Restgruppe Afrika
          SET   STAT=394
          CASE  (VDEL = 4)     * Restgruppe Asia
          SET   STAT=594
          CASE  (VDEL = 6)     * Restgruppe Nord-Amerika
          SET   STAT=694
          CASE  (VDEL = 7)     * Restgruppe Sør-Amerika
          SET   STAT=794
          CASE  (VDEL = 8)     * Restgruppe Oceania
          SET   STAT=894
          ENDCASE
          ENDIF
```

```
         FIND   ARG=STAT,DATA=NAVN,DDNAVN=LANDTXT,SW=FUNNET2
         IF     (FUNNET2 = 0)
         CASE   (VDEL = 1)
         SET    NAVN='ZZropa ellers ........'
         CASE   (VDEL = 2)
         SET    NAVN='ZZrika ellers ........'
         CASE   (VDEL = 4)
         SET    NAVN='ZZia ellers ..........'
         CASE   (VDEL = 6)
         SET    NAVN='ZZllom-Amerika ellers '
         CASE   (VDEL = 7)
         SET    NAVN='ZZr-Amerika ellers ...'
         CASE   (VDEL = 8)
         SET    NAVN='ZZeania ellers .......'
         ENDCASE
         ENDIF
         CASE   (STAT = 101)
         SET    NAVN='aanmark ..............'
         CASE   (STAT = 103)
         SET    NAVN='abnland ..............'
         CASE   (STAT = 105)
         SET    NAVN='acland ...............'
         CASE   (STAT = 106)
         SET    NAVN='aderige ..............'
         CASE   (STAT = 153)
         SET    NAVN='ZXterrike ............'
         ENDCASE
         SØJLE S1,,ANTALL
         SØJLE S2:S10,(ALDER = 0-9:10-19:20-29:30-39:40-49:
                50-59:60-69:70-79:),ANTALL
         TOTAL TXT='I alt ..............'
         FORSP VDEL,PRINT=1,TXTFIL=VDELTXT,TXTKEY=(1,1,X),
                TXT=(6,22),                NEWLINE
         FORSP NAVN,PRINT=1
TABELL   FILE   PRINT,OVERFLOW,MAXLIN=72
         HDR    (1-7),(1-63)
```

Tabell 2. Utenlandsfødte etter alder. 1. januar

```
-------------------------------------------------------------

Alder

----------------------------------------
Fødeland                    I alt     0-9    10-19  20-29   30-39

-------------------------------------------------------------
         HDR    1,50,AAR
         HDR    (1-7),(64-107)

----------------------------------------

----------------------------------------
   40-49   50-59   60-69   70-79   80-

----------------------------------------
```

```
          CASE   (NAVN = 'aanmark ..............' AND TALLY0 = 0)
          MOVE   (1,2,X),'Da'
          CASE   (NAVN = 'abnland ..............' AND TALLY0 = 0)
          MOVE   (1,2,X),'Fi'
          CASE   (NAVN = 'acland ...............' AND TALLY0 = 0)
          MOVE   (1,2,X),'Is'
          CASE   (NAVN = 'aderige ..............' AND TALLY0 = 0)
          MOVE   (1,2,X),'Sv'
          CASE   (NAVN = 'ZXterrike ............' AND TALLY0 = 0)
          MOVE   (1,2,X),'Øs'
          CASE   (NAVN = 'ZZropa ellers ........' AND TALLY0 = 0)
          MOVE   (1,2,X),'Eu'
          CASE   (NAVN = 'ZZrika ellers ........' AND TALLY0 = 0)
          MOVE   (1,2,X),'Af'
          CASE   (NAVN = 'ZZia ellers ..........' AND TALLY0 = 0)
          MOVE   (1,2,X),'As'
          CASE   (NAVN = 'ZZllom-Amerika ellers ' AND TALLY0 = 0)
          MOVE   (1,2,X),'Me'
          CASE   (NAVN = 'ZZr-Amerika ellers ...' AND TALLY0 = 0)
          MOVE   (1,2,X),'Sø'
          CASE   (NAVN = 'ZZeania ellers .......' AND TALLY0 = 0)
          MOVE   (1,2,X),'Oc'
          ENDCASE
          MOVE   (22,8,Z-)*10,S1
```

Tilslutt må vi slå sammen de to programmene til et kjøreoppsett.
Det gjør vi ved å lage JCL'en slik:

```
//0303KRLH JOB (8019),'P3',MSGLEVEL=(2,0),
//         CLASS=A,MSGCLASS=X,NOTIFY=0303KRL,REGION=4096K
//KATT     EXEC TAB1A,REGION=4096K,PARM='TXT=0277'
//INPUT    DD DSN=PX214.S0108.I459A5A1.G90MC.V00,DISP=OLD
//MATRISE  DD   DSN=&&MATRISE,
//         UNIT=WORK,
//         DISP=(NEW,PASS,DELETE),
//         DCB=(BLKSIZE=23472,LRECL=9,RECFM=FB),
//         SPACE=(23472,(1,1),RLSE)
//*
//SYSIN    DD *
:
Første program (lager liste over de som er over grenseverdien)
:
//PROGG    EXEC TAB1B,REGION=4096K,PARM='TXT=1991'
//INPUT    DD DSN=PX214.S0108.I459A5A1.G90MC.V00,DISP=OLD
//LANDTXT  DD DSN=PX214.S0130.LANDKAT,DISP=OLD
//GRENSFIL DD DSN=&&MATRISE,DISP=OLD
//VDELTXT  DD *
1    Europa i alt ................
2    Afrika i alt ................
4    Asia i alt ..................
6    Nord- og Mellom-Amerika i alt
7    Sør-Amerika i alt ...........
8    Oceania i alt ...............
9    Uoppgitt/statsløse ..........
//TABELL   DD SYSOUT=*
//SYSIN    DD *
:
Andre program (lager den trykkeklare tabellen)
```

STIKKORDSREGISTER

Kort oversikt over instruksjonene i TAB i alfabetisk rekkefølge.

Ord med store bokstaver er instruksjoner, parametre og faste verdier. Der det er ord med små bokstaver, skal du selv velge en passende verdi eller et ord. Bokstavene a og b kan erstattes med både bokstaver og tall, mens bokstavene m , n, x og y skal erstattes med tall. De instruksjoner som har flere syntakser har fått disse nummerert med en forklaring på når de skal brukes under. Listen er sortert alfabetisk etter instruksjonsnavn.

Posisjon 1- 8 Feltnavn og labels. Posisjon 10-14 Instruksjoner. Posisjon 16-72 Parametre. Posisjon 73-75 SELKRIT.

TAB1 krever at innfilen er sortert på forspaltekriteriene.
TAB1A gir summer nederst. TAB1B gir summer øverst.
TAB2 krever at du definerer hver eneste linje selv.

Max 1024 søyler når TYPE=B. Max 512 søyler når TYPE=P. Max 9999 linjer i TAB2.

Største tall vi kan ha ved binære regneoperasjoner: 2**31 - 1 (dvs. ca. 2,1 * 10**9)
Største tall vi kan ha ved regning på pakkede desimal: 10**15 - 1

****

```
        ABEND

Label   CALL  'Pgmnavn',Felt1,Felt2,.....

        CANCEL

        CASE  (Betingelse)
        :
        CASE  (Betingelse)
        :
        ENDCASE

        DOPAGE

Feltnavn FIELD (Start,Lengde,Type)*Antall

DDNAVN  FILE  PRINT,MAXLIN=nn,OVERFLOW,LRECL=nnn,BLKSIZE=nnnn,SKIP1=n      (1)

DDNAVN  FILE  LRECL=nnn,BLKSIZE=nnnn,SUM                                  (2)
```

(1) Ferdig tabell med overskrifter og printkarakterer (tabellfil)
(2) De aggregerte dataene skrives som en sekvensiell fil (sekvensiell fil)

```
        FIND  ARG=Feltnavn1,DATA=Feltnavn2,SW=Feltnavn3,DDNAVN=Filnavn

(1)     FORSP Feltnavn,PRINT=Start,PAGE/CPAGE,NOSUM/CSUM,SKIP=(m,n),PRINTGRP=n,NEWLINE,SEPCHAR='Karakter',
              CONTXT='Tekststreng 1',SELECT=(Intervall,Intervall...)/SUPRESS=(Intervall,Intervall...),FPAGE,NOLEAD,
              SUMTXT='Tekststreng 2'

(2)     FORSP Feltnavn1,TXTFIL=Filnavn/MEMBER=Membernavn,TXTKEY=(Start,Lengde,Type),TXT=(Start,Lengde),
              PRINT=Start/WORK=Feltnavn2,PAGE/CPAGE,NOSUM/CSUM,SKIP=(m,n),PRINTGRP=n,NEWLINE,SEPCHAR='Karakter',
              CONTXT='Tekststreng 1',SUMTXT='Tekststreng 2',SELECT=(Intervall,Intervall...)/
              SUPRESS=(Intervall,Intervall...),FPAGE,NOLEAD,TYPE=ALL

(3)     FORSP Feltnavn,NOSUM/CSUM,SELECT=(Intervall,Intervall...)/SUPRESS=(Intervall,Intervall...)

(4)     FORSP Rx:Ry

(5)     FORSP TXTFIL=Filnavn,TXT=(Start,Lengde),ASA=Start,RAKNR=Start,PRINT=Start
```

(1) TAB1, TAB1A og TAB1B. Forspalte uten tekstdatasett.
(2) TAB1, TAB1A og TAB1B. Forspalte med tekstdatasett.
(3) TAB1, TAB1A og TAB1B. Forspalte for sekvensiel utfil.
(4) TAB2. Forspalte uten tekstdatasett.
(5) TAB2. Forspalte med tekstdatasett.

```
DDNAVN  FYLD  KEY=(Start,Lengde,Type),DATA=(Start,Lengde,Type),MAX=n

        GETPST

(1)     HDR   (Startlinjenr-Sluttlinjenr),(Startposisjon-Sluttposisjon)

(2)     HDR   LINJENR,Start/(Start,Lengde),Konstant/Feltnavn/SIDENR/DATO
```

(1) Brukes til overskriftstekst
(2) Brukes for å få verdier fra variabler ut i overskriften

```
Label1  HOP   Label2                 (1)

        HOP   NF,Label2              (2)

        HOP   (Betingelse),Label2    (3)
```

(1) Det hoppes alltid
(2) Det hoppes hver gang, unntatt første gang
(3) Det hoppes hvis betingelsen er oppfylt

```
        IF    (Betingelse)
        :
        ELSE
        :
        ENDIF

        LIN

(1)     LOOP  (Betingelse)
        :
        ENDLOOP

(2)     LOOP  TALLYn,Startverdi,Sluttverdi,Skrittlengde
        :
        ENDLOOP
```

(1) Instruksjonen i loop-en utføres så lenge betingelsen er sann
(2) Instruksjonen i loop-en utføres fra feltet TALLYn har startverdi til det får sluttverdi.

```
        MOVE  (Start,Lengde,Type)*Antall,Søylenr/Feltnavn/Konstant/SPACES/ZEROES/LOW-VALUES/HIGH-VALUES/p10,Maske
```

```
(1)      PRCT  Sx:Sy/Sz,DEC=N

(2)      PRCT  Sx:Sy/Sz,DEC=n,TOTRAK=nnn

(1) TAB1, TAB1A, TAB1B.
(2) TAB2.
(1)      RGRP  (Betingelse),ALL

(2)      RGRP  Rekkeantall,(Betingelse),ALL

(1) Avgrenser en gruppe med rekker og gir eventuelt en felles betingelse for gruppen.
(2) Brukes til å velge opptelling i en eller flere rekkegrupper.


(1)      RREGN Rx=Parameter Operator Parameter....

(2)      RREGN Rx:Ry=Parameter Operator Parameter....

(1) Beregner en rekke
(2) Beregner en gruppe av rekker


(1)      RÆKKE Rn,(Betingelse),Verdi

(2)      RÆKKE Rn1:Rn2,(Betingelse),Verdi

(3)      RÆKKE Rn1:Rn2,TALLYn,Verdi

(1) Opptelling i en rekke
(2) Opptelling av flere rekker der verdien av et felt avgjør hvilken
(3) Opptelling av flere rekker der verdien av et TALLYn-felt avgjør hvilken


SELECT   INPUT NOT(Betingelse)

Label    SET   Feltnavn(ROUNDED/R)=Parameter Operator Parameter ...,SIGN/ABS


(1)      SGRP  (Betingelse),ALL

(2)      SGRP  Søyleantall,(Betingelse),ALL

(1) Avgrenser en gruppe med søyler og gir eventuelt en felles betingelse for gruppen.
(2) Brukes til å velge opptelling i en eller flere søylegrupper.


         SOEG  Tilfelt=Frafelt,(Tilkode1,(Intervall,...Intervall),
                                Tilkode2,(Intervall,...Intervall),
                                :
                                Tilkoden,(Intervall,...Intervall),
                                Restkode)


(1)      SREGN Sx=Parameter Operator Parameter....

(2)      SREGN Sx:Sy=Parameter Operator Parameter....

(1) Beregner en søyle
(2) Beregner en gruppe av søyler


         START TYPE=B/P,NULDIV=(a,b),TXTFEJL=n


         STOP


(1)      SØJLE Sn,(Betingelse),Verdi/Feltnavn

(2)      SØJLE Sn1:Sn2,(Betingelse),Verdi/Feltnavn

(3)      SØJLE Sn1:Sn2,,Feltnavn(Indeks)

(4)      SØJLE Sn1:Sn2,,TALLYn,Verdi/Feltnavn

(1) Opptelling i en søyle.
(2) Opptelling av flere søyler der verdien av et felt avgjør hvilken.
(3) Opptelling av en serie felter til en serie søyler.
(4) Opptelling av flere søyler der verdien av et TALLYn-felt avgjør hvilken.


(1)      TABEL Feltnavn,PAGE/PAGE1/CPAGE,SKIP=n,PRINT=(Start,Lengde)/Start,TOTKEY=aa,SELECT=(Intervall,Intervall...)/
         SUPRESS=(Intervall,Intervall...)

(2)      TABEL Feltnavn1,TXTFIL=Filnavn/MEMBER=membernavn,TXTKEY=(Start,Lengde,Type),TXT=(Start,Lengde),
         PRINT=Start/WORK=Feltnavn2,PAGE/PAGE1/CPAGE,NOSUM/CSUM,SKIP=n,PRINTGRP=n,TOTKEY=aa,
         SEPCHAR='Karakter',CONTXT='Tekststreng',SELECT=(Intervall,Intervall...)/
         SUPRESS=(Intervall,Intervall...),FPAGE,NOLEAD,TYPE=ALL

(3)      TABEL Feltnavn,TOTKEY=aa

(4)      TABEL Feltnavn1,TXTFIL=Filnavn/MEMBER=Membernavn,TXTKEY=(Start,Lengde,Type),TXT=(Start,Lengde),
         TOTKEY=aa/TOTTXT='Totaltekst',WORK=Feltnavn2

(1) TAB1, TAB1A og TAB1B. Tabellnivå uten tekstdatasett
(2) TAB1, TAB1A og TAB1B. Tabellnivå med tekstdatasett
(3) TAB2. Tabellnivå uten tekstdatasett
(4) TAB2. Tabellnivå med tekstdatasett


         TAEL


         TEST  n


         TOTAL TXT='Tekststreng',SKIP=n,PRINT=Start,FPAGE


Feltnavn WORK  (Lengde,Type),Verdi                (1)

Feltnavn WORK  (Lengde,Type),KORT(Start,Lengde)   (2)

Feltnavn WORK  (Lengde,Type),PARM(Start,Lengde)   (3)

(1) Arbeidsfeltet får den startverdi du eventuelt måtte gi
(2) Arbeidsfeltet får startverdi fra en kortfil
(3) Arbeidsfeltet får startverdi fra en parameter i EXEC-kortet i JCL
```