T. Vejl 4

LÆREBOG I KODNING

FOR

DASK

VED

CHR. ANDERSEN, NIELS IVAR BECH, OLE MØLLER

F O R O R D .

Denne lærebog i kodning er udarbejdet i tilknytning til de kodningskurser Regnecentralen har afholdt på Danmarks Tekniske Højskole gennem de sidste par år.

Ved tilrettelægningen og udarbejdelsen af de enkelte kapitler har vi været afhængige af hjælp fra mange sider, idet vort arbejde har bestået i at beskrive forhold, hvis tilblivelse skyldes dels Regnecentralens øvrige medarbejdere og konsulenter dels medlemmerne af en studiekreds bestående af deltagere fra de første kurser.

Her skal særlig fremhæves, at kapitlerne, der angår indlæsning og kapitlet om udlæsning, er skrevet i nært samarbejde med henholdsvis Per Mondrup og Helge Brodersen, som har udarbejdet de programmer, der danner grundlaget for disse afsnit, og at kapitlet om DASK-biblioteket af lignende grunde i hovedsagen er skrevet af Willy Heise. En særstilling indtager kapitel 16, der omhandler programfejl, idet dette kapitel er skrevet dels af Peter Naur, der på grundlag af sine praktiske erfaringer her har givet en vejledning til alle, der skal kode, dels af Jørn Jensen, hvis kontrolprogrammer afslutter det nævnte kapitel. Endelig må det nævnes, at Bent Scharøe Petersen, der har bygget maskinen, har været inddraget i diskussioner på omtrent alle felter.

Kontakt med de svenske regnemaskinecentrer har gjort det muligt at øse af disses rige erfaringer.

Det er vort håb, at vi i dette samarbejde har fået skrevet en lærebog i kodning for DASK, der vil lette adgangen til brugen af elektronregnemaskiner indenfor såvel forskning, industri som vore øvrige erhvervsgrene.

Tekniske udvidelser af DASK, samt udvikling og forfinelse af kodetekniken må forventes at nødvendiggøre supplerende kapitler til lærebogen f.eks. om hulkortbehandling, magnetbånd samt autokodning. Sådanne supplerende kapitler vil efterhånden som de fremkommer kunne rekvireres på Regnecentralen i lighed med specifikationer af biblioteksprogrammer og -sekvenser, som omtalt i kapitel 12.

REGNECENTRALEN 31. august 1958

CHR. ANDERSEN NIELS IVAR BECH OLE MØLLER

I N D H O L D S F O R T E G N E L S E .

KAPITEL 1.

1.1. Indledning.

De moderne automatiske regnemaskiner arbejder på grundlag af instruktionsprogrammer, der udarbejdes i en sådan form, at de kan "opfattes" af maskinen, som derefter udfører programmets instruktioner.

Formålet med denne bog er at sætte læseren ind i kodeteknikken for DASK, d.v.s. sætte læseren i stand til at udfærdige de omtalte instruktionsprogrammer for den danske elektronregnemaskine. Da dette kan gøres uden speciel forståelse af maskinens tekniske funktioner, har vi udeladt en dyberegående beskrivelse af disse. Bogen er i det hele skrevet ud fra følgende synspunkt: Alt, hvad vi har ment, koderen skulle vide besked om, er medtaget i passende omfang, hvorimod alle andre spørgsmål vedrørende DASK er udeladt.

1.2. Tal og DASK.

For alle maskiner, der arbejder med tal, gælder det, at der til et bestemt tal må være en bestemt fysisk tilstand i maskinens indre og omvendt: til en bestemt fysisk tilstand svarer der et tal. Da vi udtrykker vore tal i decimalsystemet, ligger det - ved planlægningen af en regnemaskine - nær at prøve at bygge den ved hjælp af fysiske elementer, der kan antage en af ti mulige stillinger, idet hver stilling så kan svare til et af tallene 0,1,...,9. Princippet er kendt fra f.eks. vandmålere eller kilometertællere, hvor henholdsvis tandhjul med ti tænder og tidelte tromler er de omtalte fysiske elementer.

I hurtige regnemaskiner som DASK benyttes imidlertid fysiske elementer, der kun har to mulige stillinger. Eksempelvis kan nævnes: kontakter, der enten er åbne eller lukkede; hulkort, der enten har eller ikke har et hul på et bestemt sted; elektronrør, der enten tillader strømmen at passere eller spærrer for den; ferritringe, der enten har eller ikke har en bestemt magnetisk tilstand.

De nævnte fysiske elementer svarer således ikke til decimalsystemet, hvor hvert ciffer jo kan antage ti forskellige værdier.

Skriver man derimod tal i 2-talsystemet - det binære talsystem -, hvor hvert ciffer kun kan antage en af de to værdier 0 eller 1, vil de fysiske elementer i DASK være velegnede. I overensstemmelse hermed vil vi i det følgende konsekvent benytte den skrivemåde, at indholdet af et bestemt af maskinens elementer er 0 eller 1. DASK kaldes af disse grunde en binær ciffermaskine. Det binære talsystem og repræsentation af tal i DASK behandles i kapitel 2.

1.3. Hoveddelene i DASK.

DASK består af fire hoveddele samt et kontrolbord. Figur 1 tjener til at belyse sammenhængen mellem delene, idet en pil angiver, at information kan overføres mellem de hoveddele, pilen forbinder. Kontrolbordet står i forbindelse med alle fire hoveddele.

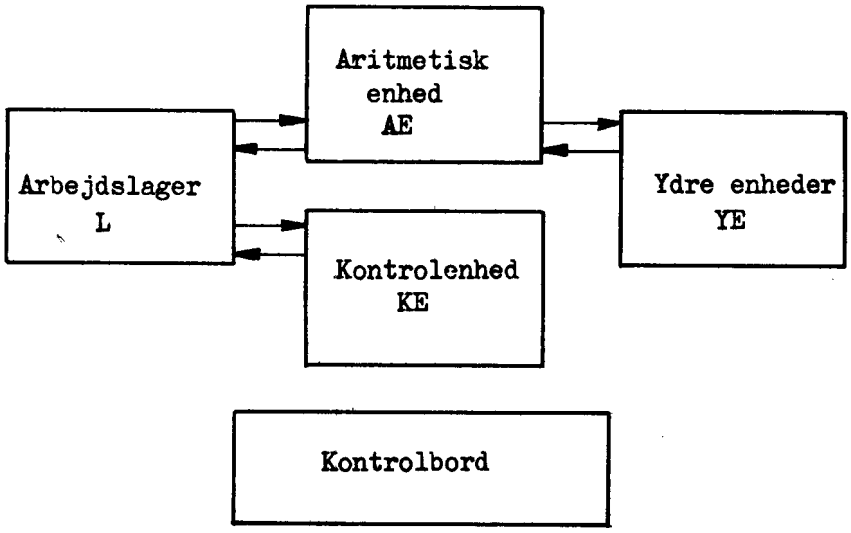


fig. 1.

1.4. Arbejdslageret

Figur 2 viser princippet i arbejdslageret. Som man ser, er det opdelt i 1024 helceller (hec) nummereret med de lige tal fra 0 til 2046. Hver helcelle rummer et helord, hvorved vi forstår en gruppe af 40 binære cifre. (Fysisk realiseres dette ved, at hver helcelle består af 40 af de under 1.2 omtalte ferritringe)

Hver helcelle kan deles i to halvceller: en venstrehalvcelle (vhac) og en højrehalvcelle (hhac). De 1024 vhac. tildeles samme numre som helcellerne, medens de 1024 hhac nummereres med de ulige tal fra 1 til 2047. Disse numre kaldes hel- eller halvcellernes adresser. Hver halvcelle rummer et halvord d.v.s. en gruppe af 20 binære cifre. Undertiden benyttes betegnelsen venstre halvord (vhao) for indholdet af en vhac. Tilsvarende benyttes højrehalvord (hhao) som betegnelse for indholdet af en hhac.

Helceller			
Nr.	Venstrehalvceller	Højrehalvceller	Nr
0			1
2			3
4			5
6			7
⋮			⋮
2046			2047

fig. 2.

Under 1.1 blev det nævnt, at DASK arbejder på grundlag af instruktionsprogrammer. De instruktioner, et sådant program indeholder, kaldes også ordrer. Den væsentligste og mest karakteristiske egenskab ved de moderne automatiske regnemaskiner er, at samtlige ordrer, der indgår i et instruktionsprogram, skal stå i maskinens lager, inden regningerne påbegyndes. Ligeledes skal det talmateriale, der skal anvendes under regningerne, være anbragt i lageret.

I DASK kan enhver ordre udtrykkes ved hjælp af 20 binære cifre, d.v.s. kan rummes i en halvcelle.

Hvordan man nedskriver ordren behandles senere; her skal det kun fastslås, at to ordrer, der i koderens program følger umiddelbart efter hinanden, i maskinens lager skal placeres i halvceller, hvis adresser er fortløbende numre.

Koderens arbejde med instruktionsprogrammet består altså i at give en række fortløbende ordrer, der skal placeres i halvcelle efter halvcelle, samt i at specificere i hvilke celler talmaterialet skal lagres.

Inden vi slutter afsnittet om arbejdslageret, vil vi kort omtale forbindelsen mellem dette og de øvrige dele af maskinen.

Kontrolenheden, KE, styrer regningerne, og disse startes ved at man sætter den i forbindelse med den halvcelle i lageret, hvor ordre nr. 1 står. Indholdet af denne halvcelle (d.v.s. ordre nr. 1) giver KE præcis besked om det, der skal ske. KE får at vide, i hvilken celle det tal står, der skal regnes med (ordrens adressedel) og den får at vide, hvori regningen består (ordrens operationsdel).

Når ordren er udført, går KE af sig selv videre til halvcellen umiddelbart efter den, der indeholder ordre nr. 1. Her har koderen skrevet ordre nr. 2, der derefter udføres o.s.v.

I tilknytning til dette billede af maskinens virkemåde fremsætter vi nogle vigtige bemærkninger:

1. DASK er i stand til at ændre programmet under regningernes forløb. Dette sker ganske simpelt ved at adressen i en ordre viser hen til en celle, hvor vi i overensstemmelse med det kodede program har placeret en ordre og ikke et tal, således at den aritmetiske funktion udføres på ordren. (Dette vil man f. eks. kunne ønske i forbindelse med visse operationer, der bevirker, at maskinen hopper tilbage i programmet og derved gennemløber en del af dette flere gange).

Herved skabes en ny ordre, d.v.s. programmet er ændret på dette sted.

2. Man vil kunne forstå, at kodningsarbejdet kræver den største nøjagtighed af koderen. En fejl i en ordres adresse kan ikke alene medføre, at man får forkerte resultater, men fejlen kan eventuelt medføre, at KE sættes i forbindelse med en hac, hvor koderen har lagret en konstant. Denne vil nu blive opfat-

tet som en ordre, hvorefter maskinen vil udføre regninger der utvivlsomt svarer yderst dårligt til koderens forventninger.

Vi afslutter afsnittet om lageret med at vise, hvordan ordren er opbygget.

Fig. 3 viser en vilkårlig af lagerets helceller. De 40 binære cifre, der kan rummes i helcellen, har numrene 0 til 39.

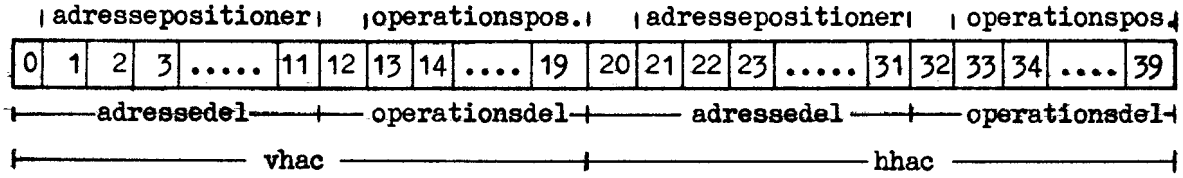


fig. 3.

Ordren står i en halvcelle. Dennes 20 pladser (binære cifre) har numrene 0 til 19 (vhac) eller 20 til 39 (hhac).

Ordrens operationsdel defineres som de otte sidste pladser, nemlig pladserne 12 til 19 i en vhac eller pladserne 32 til 39 i en hhac. Ved ordrens operationspositioner forstår vi pladserne 13 til 19 (33 til 39). Plads nr. 12 indtager altså en særstilling.

Ordrens adressedel optager de tolv første pladser. Imidlertid er elleve pladser nok til at angive adressen på en vilkårlig halvcelle i lageret (Der er netop 2048 halvceller og $2048 = 2^{11}$) og adressepositionerne 1 til 11 anvendes til dette formål. Indholdet af disse positioner kalder vi for ordrens pseudo-adresse. Tilbage er der plads nr. 0, der ligesom plads nr. 12 indtager en særstilling.

Det skal senere vise sig at indholdet i disse to pladser, indekspositionerne, kan give maskinen information om eventuelt at ændre pseudo-adressen, et forhold der på mange måder gør kodearbejdet lettere.

1.5. Den aritmetiske enhed

I denne enhed udføres alle regneoperationer (additioner, multiplikationer m.v.).

Aritmetisk enhed indeholder tre registre:

Multiplikandregistret MD;

Akkumulatorregistret AR;

Multiplikatorregistret MR samt

en adder (additionskreds), som kan danne summen af indholdet i MD og indholdet i AR og derefter indsætte summen i AR.

Figur 4 viser, hvorledes koderen kan opfatte AE.

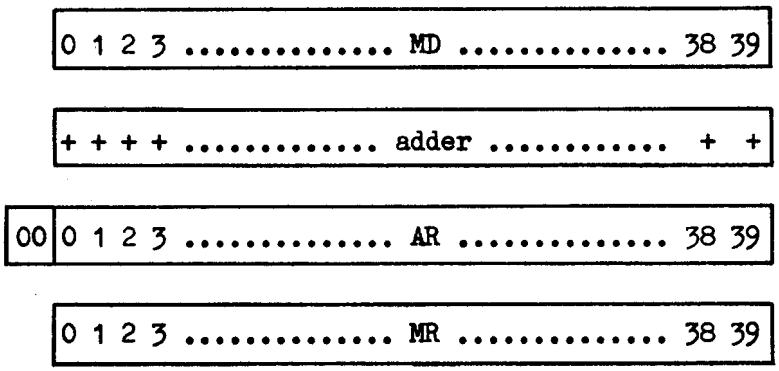


fig. 4.

Som det fremgår af figuren kan alle de tre registre rumme et helord. AR indeholder desuden en ekstra position. Denne betegnes AR_{00}. AR's øvrige positioner betegnes med AR_0, AR_1,...,AR_{39}. Tilsvarende betegnelser anvendes i det følgende for MR's og MD's enkelte positioner (MR_0,..,MD_{39}).

Da vi ofte skal tale om indholdet i en bestemt lagercelle, indholdet af et bestemt register eller indholdet af en enkelt position i et sådant, indfører vi følgende betegnelser: C(AR), C(MD), $C(MR_{17})$, C(1246), C(17) o.s.v.

Dette behøver ingen kommentarer udover den ene, at C(1246) kan betyde enten indholdet af hec 1246 eller indholdet af vhac 1246. Betegnelsen bør derfor ikke anvendes for lige tal, hvis der kan være tvivl om, hvilken af de to muligheder der

foreligger. F. eks. kan man da skrive C(1246v).

I hvilke funktioner de enkelte registre medvirker vil fremgå af de følgende kapitler.

1.6. Kontrolenheden

Denne enhed omfatter de dele af maskinen, ved hvis hjælp forløbet af de forskellige operationer styres.

KE indeholder følgende registre:

Kontrolregistret		KR	11 binære positioner
Adresseregistret		AS	11 binære positioner
Operationsregistret		OP	9 binære positioner
3 Indeksregistre	*)	IRB	11 binære positioner
		IRC	
		IRD	
Binærtæller		BR	7 binære positioner.

Endelig omfatter KE en afkodningsmatrix, hvis funktion er at vælge den celle, hvis adresse står i AS. Da det er indholdet af AS, der har betydning for koderen, vil vi ikke yderligere beskæftige os med afkodningsmatricen.

Mellem indeksregistrene og AS er der anbragt en adder, der arbejder på tilsvarende måde som adderen i AE.

Samtidig med gennemgangen af de enkelte registres funktioner uddyber vi den tidligere givne forklaring af maskinens behandling af de enkelte ordrer.

Ved beregningens begyndelse vil maskinen opfatte KR's indhold som adressen på en halvcelle i lageret, og den pågældende halvcelle vil derefter blive sat i forbindelse med KE.

---

*) Koderen arbejder yderligere med IRA, et fiktivt indeksregister. Herom nærmere på de følgende sider.

Vi starter da regningerne med i KR at anbringe adressen på den halvcelle, hvori vi har placeret den 1. ordre. Dernæst gennemløbes følgende fire operationsafsnit.

Afsnit 1. Indholdet af den pågældende halvcelles sidste syv positioner (operationspositionerne) samt indholdet af de to "særlige" indekspositioner 0 og 12 overføres til OP, der straks, styret af indholdet i indekspositionerne, vælger et indeksregister (IRB, IRC, IRD eller det fiktive indeksregister IRA)

Afsnit 2. Indholdet af adressepositionerne 1 til 11 (ordrens pseudoadresse) overføres til AS. Derefter adderes indholdet af det under 1) udvalgte indeksregister til indholdet af AS.

Afsnit 3. Additionsresultatet går til AS. C(AS) er nu adressen på operanden.

Afsnit 4a. En af de i maskinen indbyggede operationer udføres på operanden, nemlig den operation, der bestemmes af indholdet i OP. Samtidig med, at operationen udføres, forberedes start af næste ordre.

Dette sker normalt ved, at indholdet af KR øges med 1 således, at KR nu indeholder adressen på den halvcelle, hvori næste ordre er lagret.

Adskillige af de indbyggede operationer, nemlig alle hopoperationer, kan imidlertid ændre det under 4a nævnte totalt. Disse operationer bevirker nemlig, at 4a erstattes med

4b. Den - under 3 - i AS indsatte adresse, føres direkte videre til KR. Da maskinen henter sin næste ordre fra den halvcelle, hvis adresse står i KR, er vi hermed hoppet hen til et helt andet sted i regneprogrammet.

Bemærkninger. Enhver ordre skal indeksmærkes, d.v.s. ordren skal indeholde en angivelse af hvilket indeksregister, man vil benytte. Dette kan være indeksregistret IRA. Dette findes ikke! hvoraf navnet det fiktive indeksregister. Benyttes IRA i en ordre, betyder det, at der intet adderes til pseudoadressen, der altså bliver den endelige adresse på operanden. Benyttes et vil-

kårligt af de tre indeksregistre IRB, IRC eller IRD, vil den endelige adresse kun blive lig pseudoadressen, såfremt indholdet af det pågældende indeksregister er nul.

Ved denne gennemgang af maskinens virkemåde har vi omtalt samtlige registre i KE med undtagelse af binærtælleren BR.

Vi afslutter dette afsnit med et par eksempler, der illustrerer de to forskellige behandlinger af en ordre.

Eks. 1. I vhac 218 placerer vi en ordre, hvis udførelse vil bevirke at tallet x, der står i vhac 1624, adderes til C(AR). C(IRB) = 1600.
Hvordan dette kodes, og hvad der egentlig står i de enkelte dele af maskinen er helt uden betydning i denne sammenhæng. Vi skriver derfor blot de nødvendige informationer på de rigtige steder. Se figur 5, der viser situationen inden starten.

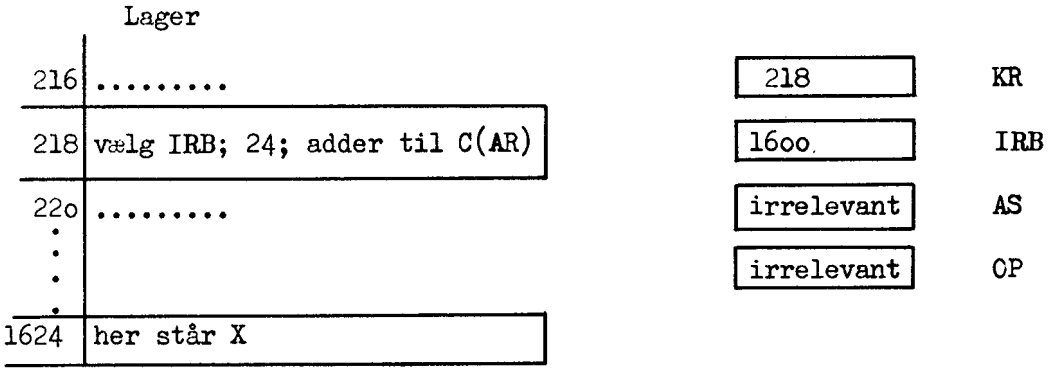


fig. 5.

Efter udførelsen af afsnit 1, er L, KR og IRB uændrede. OP ser ud, som fig. 5a viser, og IRB er valgt.

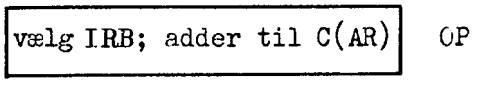


fig. 5a.

Så udføres afsnit 2. L, KR, OP og IRB er uændrede.
KE iøvrigt ser ud, som fig. 5b viser.

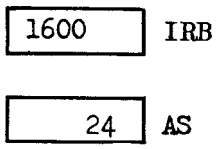


fig. 5b.

Efter afsnit 3 har AS fået det på fig. 5c viste indhold.
L, KR, OP og IRB er stadig uændrede

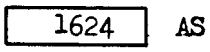


fig. 5c.

Under afsnit 4a sker den ønskede operation: C(AS) er adressen på tallet x og C(OP) bevirker additionen af x til C(AR). Samtidig øges C(KR) med 1 til 219.
I eksemplet har vi ønsket, at vise hvorledes indeksregistrene arbejder. I praksis ville man ikke have brugt IRB, men anvendt det fiktive indeksregister IRA. Ordren skulle da have haft pseudoadressen 1624 (i stedet for 24) og "vælg IRA" (i stedet for "vælg IRB"), og resultatet var blevet det samme.

Eks. 2. I vhac 218 ønsker vi at placere en hopordre, der skal bevirke, at maskinen henter sin næste ordre fra vhac 1624. Figuren og tekst fra eks. 1 kan også anvendes her med den ændring, at operationsdelen "adder til C(AR)" naturligvis skal erstattes med "hop".
Og 4a skal erstattes med 4b, d.v.s. at C(AS) efter udførelsen af 3. overføres til KR, d.v.s. C(KR)=C(AS)=1624, hvormed vi har opnået det ønskede resultat.

1.7. Ydre enheder

Disse enheders betydning er foreløbig af mindre interesse for koderen. Vi vil derfor udskyde en nærmere omtale af de enkelte enheder indtil videre, og indskrænke os til at nævne de dele af maskinen, der hører ind under betegnelsen YE.

Disse er: Foto) Elektrisk strimmellæser til indlæsning af instruktioner og tal kodet i 5-hulsstrimmel.

Skrivemaskine og strimmelhuller til udlæsning af resultater.

Tromlelager og magnetisk bånd som reservelager for arbejdslageret.

Endelig hulkortenheder til hurtig ind- og udlæsning af mange data.

1.8. Afsluttende bemærkninger til 1. kapitel.

Meningen med de foregående sider har været at give læseren et indtryk af maskinens virkemåde. For at kunne opnå dette har vi set helt bort fra mange detaljer, som udarbejdelsen af det endelige instruktionsprogram kræver, at koderen er helt fortrolig med.

Mange af de allerede behandlede spørgsmål må følgelig tages op igen og underkastes en nøjagtigere undersøgelse. Det vil herved ikke kunne undgås, at de følgende kapitler indeholder visse gentagelser af det her skrevne, men i det store og hele vil den følgende fremstilling bygge videre på kapitel 1.

Vi opfordrer derfor læseren til at gøre sig fuldt fortrolig med indholdet af bogens første sider.

Spørgsmålet om maskinens regnehastighed skal sluttelig berøres.

Mange ordrer udføres på 56 μ sek ($\frac{56}{1000000}$ sek) d.v.s. at maskinen kan udføre ca. 1 million simple operationer i minuttet.

KAPITEL 2.

2.1 Det binære talsystem

I vort sædvanlige, decimale talsystem er grundtallet 10, hvilket betyder, at f.eks. heltallet 3567 skal forstås som

$$3 \cdot 10^3 + 5 \cdot 10^2 + 6 \cdot 10^1 + 7 \cdot 10^0,$$

og brøken 0.3975 skal forstås som

$$0 \cdot 10^0 + 3 \cdot 10^{-1} + 9 \cdot 10^{-2} + 7 \cdot 10^{-3} + 5 \cdot 10^{-4},$$

således at et tal (heltal, brøk eller blandet) i virkeligheden skal læses som en produktsum af de angivne cifre og passende potenser af 10, hvis eksponenters størrelse afhænger af det tilsvarende ciffers placering i afstand fra kommaet.

En talværdi kan naturligvis udtrykkes i et talsystem med vilkårligt grundtal - specielt 2, d.v.s. i det binære talsystem, hvor de enkelte cifre kaldes binære cifre og cifrene efter kommaet specielt binaler.

Det decimale tal 47.625 skrives således i binær form

$$101111.101$$

som skal forstås

$$1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3}$$

hvilket netop er lig

$$32 + 0 + 8 + 4 + 2 + 1 + 0.5 + 0 + 0.125 = 47.625$$

2.2 Transformation af decimalbrøk til binærbrøk - og omvendt

Af speciel interesse er binærbrøken, idet cifrene i et ord i lageret eller et register i AE normalt tillægges en talværdi, der numerisk er $\left(\leq\right)$ 1. (Se afsnit 2.4).

Omformningen fra decimalbrøk til binærbrøk sker ved successiv multiplikation med 2:

Decimalbrøken a tænkes skrevet på formen:

$$a = b_1 \cdot 2^{-1} + b_2 \cdot 2^{-2} + \ldots\ldots = 0.b_1b_2b_3 \ldots\ldots b_v \leq 1.$$

Ved multiplikation af denne ligning med 2 fås

$$2a = b_1 + b_2 \cdot 2^{-1} + \ldots\ldots = b_1.b_2b_3 \ldots\ldots b_v$$

Heraf aflæses, at b_1 = heldelen af 2a, som er enten 0 eller 1.
Sættes $2a - b_1 = a_1 = 0.b_2b_3 \ldots\ldots b_v$
og gentages processen, findes b_2 o.s.fr.

$$\begin{aligned}
&0.5625 \cdot 2\\
&1.1250 \cdot 2\\
&0.2500 \cdot 2\\
&0.5000 \cdot 2\\
&1.0000 \cdot 2\\
&0.0000 \qquad\qquad \text{d.v.s. } 0.5625 \sim 0.1001.
\end{aligned}$$

Omdannelse fra binærbrøk til decimalbrøk kan ske på den tilsvarende måde, nemlig ved at multiplicere med basis i 10-talsystemet blot noteret i 2-talsystemet, altså 1010. Måske foregår omformninger denne vej dog lettere ved en anden metode, som vi derfor vil vise ved et eksempel; til gengæld er den ikke særlig egnet ved overgangen fra decimal- til binærbrøk.

$$\begin{aligned}
0.1001 &= \left(\left(1 \cdot \tfrac{1}{2} + 0\right) \cdot \tfrac{1}{2} + 0\right) \cdot \tfrac{1}{2} + 1) \cdot \tfrac{1}{2} \quad \text{, ren omskrivning}\\
&= \left(\left(0.5 \cdot \tfrac{1}{2} + 0\right) \cdot \tfrac{1}{2} + 1\right) \cdot \tfrac{1}{2} \quad \text{, inderste parentes udregnes,}\\
&= \left(0.25 \cdot \tfrac{1}{2} + 1\right) \cdot \tfrac{1}{2} \quad \text{, næste parentes udregnes}\\
&= (0.125 + 1) \cdot \tfrac{1}{2}\\
&= 1.125 \cdot \tfrac{1}{2}\\
&= 0.5625
\end{aligned}$$

Ved omsætningerne, der er vist her, er de givne brøker opfattet som eksakte, uafkortede tal, og det er derfor fyldestgørende at skrive

$$\begin{aligned}
0.5625 &\longrightarrow 0.1001\\
0.1001 &\longrightarrow 0.5625 \ .
\end{aligned}$$

Tænker man sig derimod, at de givne tal er almindelige, afrundede værdier, bør man skrive:

$$\begin{aligned}
0.5625 &\longrightarrow 0.1001000000000 \pm 2^{-14}\\
0.1001 &\longrightarrow 0.56 \pm 0.03 \ .
\end{aligned}$$

Bemærk iøvrigt, at en vilkårlig endelig binalbrøk altid omsættes til en endelig decimalbrøk, mens det omvendte ikke gælder.

2.3 Udover det binære (og decimale) system får vi også brug for det sedecimale system, d.v.s. 16-talsystemet. Dette fremkommer ved, at 4 binaler slås sammen til eet ciffer og derved giver det tilsvarende sedecimale ciffer. De ciffersymboler, man kommer til at mangle, er kaldt A, B, C, D, E og F.

10-talsystem	2-talsystem	16-talsystem
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

2.4 Repræsentation af tal i DASK

De 40 binære positioner i en helcelle eller i et register er, som vi ved, nummereret fra 0 til 39. Placerer vi kommaet mellem pos. 0 og 1, og betegnes indholdet i pos. j med p_j, vil et vilkårligt indhold i de 40 positioner umiddelbart have værdien

$$p = \sum_{j=0}^{39} p_j \cdot 2^{-j}$$

Dette giver tal i intervallet $0 \leqq p \leqq 2 - 2^{-39}$. Vi går imidlertid frem på en lidt anden måde, således at vi, uden anden information end den de 40 pos. kan give, også får negative tal med, mod til gengæld at indskrænke værdiområdet numerisk, nemlig til $-1 \leqq p \leqq 1 - 2^{-39}$. Dette kalder vi kort DASK-intervallet, og tilsvarende kaldes tal i dette område for DASK-tal.

Tænker vi os et DASK-tal, p, skrevet som binærbrøk med sædvanligt fortegn, + eller -, repræsenteres det på følgende måde i maskinen:

i celler eller registre med 40 positioner:

når $p \geq 0$ ved p's binære cifre

når $p < 0$ ved cifrene i $2^1 + p$ $(= 10 + p)$

Eks: $p = +\ 0.1100 \ldots\ldots 0$ $(= +\ 0.75)$

vil stå som

$0.1100 \ldots\ldots 0$

$p = -\ 0.0110 \ldots\ldots 0$ $(= -0.375)$

$= \ 1.1010 \ldots\ldots 0 - 2^1$

vil stå som

$1.1010 \ldots\ldots 0.$

Vi ser, at der for DASK-tal gælder, at cifret foran kommaet altid er 0, når $p \geq 0$ og altid 1, når $-1 \leq p < 0$. Dette ciffer kaldes derfor fortegncifret og vil sammen med de øvrige cifre i registret entydigt bestemme p's størrelse og fortegn. Når vi altså har et indhold i en celle: $p_0 p_1 p_2 \ldots p_{39}$ er værdien:

når $p_0 = 0$ simpelthen $\sum_0^{39} p_j \cdot 2^{-j}$,

når $p_0 = 1$ derimod $\sum_0^{39} p_j \cdot 2^{-j} - 2^1$,

hvilket samlet kan skrives

$$\sum_{j=0}^{39} p_j \cdot 2^{-j} - 2p_0 \ .$$

Ønsker man at skifte fortegn på et tal, som står i DASK, gøres det simpelthen ved at udskifte alle 1-taller med nul og alle nuller med et 1-tal og derefter addere 1 i sidste position - eller anderledes sagt: at udskifte alle cifre med deres komplementære, begyndende fra venstre og til, men ikke med, den position der indeholder sidste betydende ciffer (1-tal). Dette kaldes at danne tallets komplement i det pågældende register.

Eks. Idet vi nøjes med 10 positioner i registret, er:

$p = 0.101101100$

$-p \sim 1.010010011 + 0.000000001$

$= 1.010010100.$

Vedrørende tegnskifte af DASKtallet -1 henvises til afsnittet 2.8.

Alt i det foregående svarer nøje til forholdene i almindelige bord-regnemaskiner, hvor negative tal i et register (med fuld menteoverføring) repræsenteres ved 10-talskomplementet regnet i forhold til et 1-tal i en tænkt position, umiddelbart tilvenstre for registrets yderste position. Iøvrigt svarer det også til fremstillingsmåden ved negative logaritmer.

Vi gør tilsidst opmærksom på, at vor fremstilling af tal, når der er brug for det, selvfølgelig kan udvides til registre med flere positioner. Er der f.eks. to positioner foran kommaet, ialt 41 positioner, repræsenteres et DASK-tal, p, ved

$$\text{når } p \geq 0 \quad : \quad p\text{'s binære cifre,}$$

$$\text{når } p < 0 \quad : \quad \text{cifrene i } 2^2 + p \ (= 100+p).$$

I eksemplerne ovenfor bliver det henholdsvis

$$00.1100 \ldots\ldots 0$$

$$\text{og} \quad 11.1010 \ldots\ldots 0 \ .$$

2.5 Vi skal her og i de følgende afsnit gennemgå de fire regningsarter.*) Der vil indledningsvis hver gang blive givet en kort beskrivelse af, hvad udgangssituationen og resultatet af processen er. Det gælder i store træk, at man kan kode alene ud fra disse korte beskrivelser, og de efterfølgende detaljerede behandlinger vil derfor mere eller mindre kunne overspringes.

---

*) Ved hjælp af følgende enkeltresultater, gældende for addition med encifrede tal i 2-talssystemet:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$\text{og} \quad 1 + 1 = 10 \ ,$$

overbeviser læseren sig let om, hvordan de sædvanlige fremgangsmåder fra regning i 10-talssystemet med papir og blyant, direkte kan overføres til regning i 2-talssystemet.

Addition og Subtraktion

Disse operationer foregår med det ene led stående i AR og det andet i en celle i lageret. Resultatet kommer i AR, men vil i almindelighed kun kunne anvendes, hvis det ligger i DASK-intervallet. Ligger det udenfor dette interval, er der indtruffet spild.

Udførlig beskrivelse

Vi bemærker først, at subtraktion foregår ved, at subtrahenden (ved transporten fra lageret gennem registret MD til AR) byttes ud med sit komplement, og at processen derefter foregår som en addition. Det er derfor nok kun at behandle addition.

Additioner er knyttet til AR, og af grunde, vi uddyber nedenfor, udvides AR ved denne proces med en hjælpeposition, AR_{00}, foran AR_0 (jvf. figur 4, side 1.6). Det er derfor hensigtsmæssigt og i overensstemmelse med denne positions virkemåde at behandle tal, der overføres fra lageret til AR, som om de blev suppleret op til 41 bits, så deres fortegnsciffer gentages i AR_{00} (jvf. de sidste linier i afsnit 2.4).

Vi gennemgår alle muligheder.

Tilfælde 1. To pos. DASK-tal:

$$\begin{array}{rl} & 00.\ a_1a_2 \ldots\ldots a_{39} \\ + & \underline{00.\ b_1b_2 \ldots\ldots b_{39}} \\ & 0c_0.c_1c_2 \ldots\ldots c_{39} \end{array}$$

hvor c_0 kan være 0 eller 1.

Tilfælde 2. Et pos. og et neg. DASK-tal:

$$\begin{array}{rl} & 00.\ a_1a_2 \ldots\ldots a_{39} \\ + & \underline{11.\ b_1b_2 \ldots\ldots b_{39}} - 2^2 \\ & 11.\ c_1c_2 \ldots\ldots c_{39} - 2^2 \end{array}$$

$$\text{eller } 100.\ c_1c_2 \ldots\ldots c_{39} - 2^2 = 00.c_1c_2 \ldots\ldots c_{39}$$

Tilfælde 3. To neg. DASK-tal:

$$\begin{array}{l} 11.\ a_1a_2\ \ldots\ldots\ a_{39} - 2^2 \\ \underline{11.\ b_1b_2\ \ldots\ldots\ b_{39} - 2^2} \\ 11c_0.c_1c_2\ \ldots\ldots\ c_{39} - 2^3 = 1c_0.c_1c_2\ \ldots\ldots\ c_{39} - 2^2 \end{array}$$

hvor c_0 kan være 0 eller 1.

I de to tilfælde : 1. med $c_0 = 1$ og 3. med $c_0 = 0$, har, som man ser, resultatet overskredet DASK-intervallet, og tænkte vi på uden videre at overføre det til en celle, ville det få galt fortegnsciffer. Vi siger, at der er indtruffet spild. Det er klart, at den omstændighed, at addition af to DASK-tal ikke behøver pany at give et DASK-tal, er så alvorlig, at det ved regninger, hvis numeriske forløb man ikke på forhånd helt behersker, kan blive nødvendigt at undersøge om spild er indtruffet. Maskinen er derfor indrettet til på forlangende at slå alarm i disse tilfælde, og man ser at kendetegnet på spild er, at $C(AR_{00}) \neq C(AR_0)$, (spildindikation). Man ser også, at selvom spild er indtruffet, er sagen stadig under kontrol, thi takket være AR's 41 positioner kan addition af to DASK-tal aldrig føre til overskridelse af det udvidede AR's kapacitet (deres sum vil jo altid være numerisk ≤ 2). Lader man derfor maskinen skifte indholdet i alle AR's positioner en plads tilhøjre, d.v.s. man ganger med $\frac{1}{2}$, en proces der foregår sådan, at der i AR_0 indføres det, der stod i AR_{00}, mens der i AR_{00} indføres et 0, har man den halve sum stående med rigtigt fortegnsciffer i AR_0 (c_{39} er selvfølgelig forsvundet). Altså har den simple ting at skifte een pos. til højre gjort det muligt, trods spild, at få fat i den halve sum, som man så f.eks. kan lagre.

Vi fremhæver udtrykkelig, at udvidelsen af AR til 41 positioner kun foregår i hver enkel addition taget for sig og ikke føres ubrudt videre fra addition til addition, idet additionen med AR's indhold rent teknisk foregår ved, at de 40 bits, $C(AR_{0-39})$, tages ud og indføres pany. En kæde af additioner forløber således efter skemaet:

$$\begin{array}{rl} a & 11.0011 \\ + b & \underline{11.0101} \\ a + b & 10.1000 \quad \text{(spild), der ved næste addition} \end{array}$$

tages ud og føres ind som

$$\begin{array}{rl} & 00.1000 \quad \text{(spild forsvundet!)} \\ + c & \underline{00.0010} \\ & 00.1010 \end{array}$$

, og dette er ikke a + b + c. D.v.s., reagerer man ikke efter hver enkel addition overfor et spild, men mener, at det kan være nok at prøve tilsidst , har man mistet kontrollen over, hvad der sker.

Tilsidst nævner vi, at tilfældene 1., 2. og 3. viser, hvordan sædvanlig sammenlægning af cifrene i de 41 positioner: 00,0,1,2,....39 giver de cifre, der skal stå i DASK, altså at korrektionsleddene 2^2, som kun tjener til i overvejelserne skridt for skridt at sikre den talmæssige identitet, kan udelades. Sammenlægningen af de 41 bits er derfor et fuldtud dækkende billede af, hvad der foregår i maskinen.

Eks.: $(-0.1875) - (+ 0.25) = (-0.0011) - (+0.0100)$

$= (11.1101 - 2^2) - (00.0100)$

$= (11.1101 - 2^2) + (11.1100 - 2^2)$

$\sim 11.1101 + 11.1100$

$$\begin{array}{rl} \text{d.v.s.} & 11.1101 \\ + & \underline{11.1100} \\ & 11.1001 \end{array}$$

2.6 Multiplikation

Multiplikation udføres med multiplikator b i MR og multiplikand a i en celle m.

Der er to former:

lang multiplikation, hvor produktet a · b står uafkortet i det, der kaldes lang akkumulator, og hvor multiplikator går tabt,

kort multiplikation, hvor det til 39 binaler afrundede produkt står i AR, og multiplikator er bevaret i MR.

Lang akkumulator er positionerne AR_{0-39} og MR_{1-39} kædet sammen til eet langt register (MR_0 overspringes altså).

Udførlig beskrivelse

Som nævnt står multiplikator i MR, og den kan skrives

$$b = \sum_{j=0}^{39} b_j \cdot 2^{-j} - \begin{cases} 0 & , \text{hvis } b_0 = 0. \\ 2^1 & , \text{hvis } b_0 = 1. \end{cases} \qquad (1)$$

For at forklare fremgangsmåden, der benyttes i maskinen, skal vi imidlertid bruge, at (1) kan samles i det ene udtryk

$$b = \sum_{j=1}^{39} b_j \ 2^{-j} - b_0. \qquad (2)$$

Kaldes multiplikanden a, kan a ∘ b nu udtrykkes ved

$$a \cdot b = \sum_{1}^{39} a \circ b_j \circ 2^{-j} - a \cdot b_0$$

$$= (...(((a \circ b_{39})\tfrac{1}{2} + a \circ b_{38})\tfrac{1}{2} + a \circ b_{37})\tfrac{1}{2} + ... + ab_1)\tfrac{1}{2} - a \circ b_0, \qquad (3)$$

og efter dette udtryk arbejder maskinen.
Multiplikationen begynder med, at AR renses for et eventuelt indhold. Derefter adderes $a \cdot C(MR_{39}) = a \circ b_{39}$ til C(AR) (=0), d.v.s. hvis $b_{39} = 1$ lægges a til C(AR), hvis $b_{39} = 0$ lægges 0 til C(AR). Additionen foregår, som vi ved, i AR udvidet til 41 positioner. Dernæst halveres det nye C(AR), ved at der skiftes en position til højre, hvorved $C(AR_0)$ bliver det ciffer, der stod i AR_{00} før skiftet (AR_{00} selv får 0), og $C(AR_{39})$ føres ind i MR_0, idet samtidig $C(MR_{0-38})$ rykkes over i MR_{1-39}, og b_{39}, der stod i MR_{39}, forsvinder, kort sagt: et højreskift i AR og i MR. I AR, MR_0 står altså efter dette første skridt: $a \cdot b_{39} \circ \frac{1}{2}$. Næste skridt kan nu begynde, og det er denne gang $a \circ C(MR_{39}) = a \circ b_{38}$, der adderes til $C(AR) = a \cdot b_{39} \circ \frac{1}{2}$, og efter højreskiftet vil der i AR, MR_{0-1} stå: $(a \circ b_{39} \circ \frac{1}{2} + a \cdot b_{38})\ \frac{1}{2}$. På denne måde fortsættes, og først når b_0 står i MR_{39}, ændrer maskinen fremgangsmåde, idet nu $-a \circ b_0$

bliver adderet til C(AR), og der foretages ikke noget højreskift i AR, men kun i MR, hvor et nul føres ind i MR_0. Resultatet er, at hele det uafkortede produkt a · b, der består af fortegnscifret og 78 binaler, står i AR_{0-39}, MR_{1-39}.

I hvert skridt af processen udføres en addition. Hvordan forholder det sig her med spild? Da addenderne i den første addition begge er DASK-tal, skal den påfølgende halvering påny give et DASK-tal, og vi ved fra afsnit 2.5, at dette tal ikke er blevet beskadiget af, at additionen eventuelt har givet spild. Addenderne i de følgende additioner vil derfor vedblivende begge være DASK-tal. Vi ser endvidere, i overensstemmelse med hvad vi ved, at resultatet vil være et DASK-tal. For tilfældet a = b = -1 se dog afsnittet 2.8.

Den beskrevne proces giver altså det uafkortede produkt stående i den lange akkumulator.

Vi vil nøjes med at forklare kort multiplikation gennem et af de to eksempler, der vises nedenfor, men nævner blot, at afrundingen kommer i stand ved før multiplikationen at indføre $\frac{1}{2} \sim 0.10 \ldots 0$ i AR.

I eksemplerne har vi for at spare plads ladet 5 positioner gælde for et helord.

Eks. Lang multiplikation. $C(n) = 1.0011 = -\frac{13}{16}$ (= a)

$C(MR) = 1.1011 = -\frac{5}{16}$ (= b).

	AR	MR	
C(AR)	00.0000	1.1011	
+ a · b_4	11.0011		
C(AR) + a · b_4	11.0011	1.1011	
skift	01.1001	1.1101	
C(AR)	11.1001	1.1101	
+ a · b_3	11.0011		
$C(AR)^3$ + a · b_3	10.1100	1.1101	spild !
skift	01.0110	0.1110	
C(AR)	11.0110	0.1110	
+ a · b_2	00.0000		
C(AR) + a · b_2	11.0110	0.1110	
skift	01.1011	0.0111	

	AR	MR	
C(AR)	11.1011	0.0111	
+ a·b_1	11.0011		
C(AR) + a·b_1	10.1110	0.0111	spild !
skift	01.0111	0.0011	
C(AR)	11.0111	0.0011	
-a·b_0	00.1101		
C(AR) - a·b_0	00.0100	0.0011	
skift i MR	00.0100	0.0001	
resultat	0.0100	0001	= + $\frac{65}{256}$.

Eks. Kort multiplikation: C(n) = 0.1101 = $\frac{13}{16}$ (= a)

C(MR)= 1.1001 = - $\frac{7}{16}$ (= b)

	AR	MR	
C(AR)	00.1000	1.1001	
+ a·b_4	00.1101		
C(AR) + a·b_4	01.0101	1.1001	) ved højreskiftet føres her
skift	00.1010	1.1100	) indholdet i MR_{39} ind i MR_0.
C(AR)	00.1010	1.1100	
+ a·b_3	00.0000		
C(AR) + a·b_3	00.1010	1.1100	
skift	00.0101	0.1110	
C(AR)	00.0101	0.1110	
+ a·b_2	00.0000		
C(AR) + a·b_2	00.0101	0.1110	
skift	00.0010	0.0111	
C(AR)	00.0010	0.0111	
a·b_1	00.1101		
C(AR) + a b_1	00.1111	0.0111	
skift	00.0111	1.0011	
C(AR)	00.0111	1.0011	
- a b_0	11.0011		
C(AR)-a b_0	11.1010	1.0011	
skift i MR	11.1010	1.1001	
resultat	1.1010	= - $\frac{6}{16}$	.

2.7 Division.

Division udføres med divisor stående i en celle i lageret og dividenden stående

enten i lang akkumulator: lang division
eller i AR: kort division.

Kvotienten kommer i begge tilfælde i MR_{0-39} og divisionsresten med faktoren 2^{39} i AR.

For at tallet, der fremkommer i MR, virkelig skal være kvotienten, kræves, at $|\text{dividend}| \leq |\text{divisor}|$, en betingelse koderen må sørge for er opfyldt.

Udførlig beskrivelse.

I maskinen gennemføres divisionsprocessen efter følgende algoritme, kaldet "non-restoring division".

Vi sætter divisor $C(m) = N$

og dividend $C(AR_{0-39}, MR_{1-39}) = T$

Første skridt.

Der skiftes 1 position til venstre i den lange akkumulator. Var C(m) og C(AR)'s fortegn det samme før skiftet, indføres under dette et 1-tal i MR_{39}, og derpå følger en subtraktion af C(m) fra nyt C(AR). Var C(m) og C(AR)'s tegn modsatte, indføres et 0 i MR_{39} og skiftet efterfølges af en addition af C(m) til C(AR).

Hvis man tænker sig, at man i stedet for at skifte T til venstre i den lange akkumulator skiftede C(m) tilhøjre, ville det lige beskrevne have vist en vis analogi til det, der sker på det enkelte trin af en sædvanlig division. Vi vil derfor opfatte C(AR) efter det første skridts udførelse som en divisionsrest, r_1, forskudt 1 position tilvenstre: $C(AR) = 2r_1$.

De næste 38 skridt udføres efter nøjagtig samme forskrift; det p'te skridt forløber således:

1 venstreskift i den lange akkumulator,
samme tegn hos C(m) og $C(AR) = 2^{p-1}r_{p-1}$ *) før skiftet bevirker, at et 1-tal føres ind i MR_{39}, og C(m) subtraheres,
modsatte tegn bevirker, at et 0 føres ind i MR_{39}, og at C(m) adderes.

---

*) faktoren 2^{p-1} da der før skridt nr. p er udført ialt p-1 venstreskift.

Når der på denne måde er udført ialt 39 skridt, vil $C(AR)=2^{39}\cdot r_{39}$, og vi vil til angivelse af kvotienten have fået 39 binære cifre i MR_{1-39}.

Derefter udføres et venstreskift alene i MR, hvorunder der altid indføres 1 i MR_{39}, og til sidst inverteres cifret i MR_0. (Invertering af et ciffer q betegnes med $\overline{q}$ og består i, at cifret byttes ud med sit modsatte: $\overline{0} = 1$, $\overline{1} = 0$.)

Kaldes cifrene i MR_{0-39}, før cifret i position 1 bliver inverteret, før $q_0\ q_1.....q_{38}1$, og vedtager vi at betegne DASK-værdien af en sådan cifferrække med $D(q_0,q_1....q_{38}1)$, skal det nu vises, at $C(MR) = D(\overline{q}_0,\ q_1....q_{38}1)$ virkelig er kvotienten.

Vi har efter første skridt, at $C(AR) = 2r_1 = 2T + (1 - 2q_0)\cdot N$ og almindeligt efter det p'te skridt, at $C(AR) = 2^p\cdot r_p = 2^p\cdot r_{p-1}+(1-2q_{p-1})\cdot N$

Multipliceres med passende potenser af 2, får vi for de 39 skridt

$$\begin{aligned}
2^{39}r_1 &= 2^{39}T + 2^{38}(1-2q_0)N\\
2^{39}r_2 &= 2^{39}r_1 + 2^{37}(1-2q_1)N\\
&\vdots\\
2^{39}r_p &= 2^{39}r_{p-1} + 2^{39-p}(1-2q_{p-1})N\\
&\vdots\\
2^{39}r_{39} &= 2^{39}r_{38} + 2^{0}(1-2q_{38})\ N
\end{aligned}$$

udtryk, der ved summation giver

$$2^{39}\sum_{i=1}^{39} r_i = 2^{39}\left[T + \sum_{i=1}^{38} r_i\right] + \left[\sum_{i=0}^{38} 2^i - \sum_{i=1}^{39} 2^i q_{39-i}\right]\cdot N$$

$$2^{39}\ r_{39} = 2^{39}T + \left[2^{39} - 1 - \sum_{j=0}^{38} q_j\ 2^{39-j}\right]\cdot N$$

$$r_{39} = T + \left[1-2^{-39} - \sum_{j=0}^{38} q_j\ 2^{-j}\right]\cdot N$$

hvoraf

$$T = r_{39} + \left[\sum_{j=0}^{38} q_j \, 2^{-j} + 1 \cdot 2^{-39} - 1 \right] \cdot N$$

$$= r_{39} + \left[\sum_{j=0}^{39} q_j \, 2^{-j} - 1 \right] \cdot N, \qquad (\underline{q_{39} = 1})$$

$$= 2^{-39} C(AR) + \left[\sum_{j=0}^{39} q_j 2^{-j} - 1 \right] N$$

eller

$$Q = \frac{T}{N} = \sum_{j=0}^{39} q_j \, 2^{-j} - 1 + \frac{r_{39}}{N}$$

Idet $$D(q_0, q_1 \ldots . q_{39}) = \sum_{j=0}^{39} q_j \, 2^{-j} - 2q_0,$$

bliver

$$Q = D(q_0, q_1 \ldots . q_{39}) + 2q_0 - 1 + \frac{r_{39}}{N} \, ,$$

og her er

$$D(q_0, q_1 \ldots . q_{39}) + 2q_0 - 1 = \begin{cases} D(1, q_1 \ldots . q_{39}) & \text{når } q_0 = 0 \\ D(0, q_1 \ldots . q_{39}) & \text{når } q_0 = 1 \end{cases}$$

kort sagt: $D(q_0, q_1 \ldots . q_{39}) + 2q_0 - 1 = D(\bar{q}_0, q_1 \ldots q_{39})$.

Altså får vi $Q = \frac{T}{N} = D(\bar{q}_0, q_1 \ldots . q_{39}) + \frac{r_{39}}{N} = C(MR) + \frac{r_{39}}{N}$.

Man lægger mærke til, at algoritmen har givet det ønskede resultat ganske uafhængig af T og N's fortegn.

Det er selvfølgelig afgørende for metoden, at der ved sammenligningen af C(AR) og C(m)'s fortegn ikke er spild i AR, således at AR_0 på dette tidspunkt virkelig viser r_{p-1}'s sande fortegn.

Man ser imidlertid, at selvom venstreskiftet har bevirket spild, vil den påfølgende addition eller subtraktion altid bringe resultatet tilbage i DASK-intervallet, thi taget numerisk gælder

at $\quad |C(m)| \geq |T|$

medfører $-|C(m)| \leq 2\,|T| - |C(m)| \leq |T|$,

således, at $|2\,|T| - |C(m)|| = |2r_1|$*) $= |C(AR)|$ igen er et DASK-tal numerisk $\leq |C(m)|$, hvilket vil forplante sig videre til at gælde alle $2^{p-1} \cdot r_{p-1}$

Vi vil ved et eksempel vise, hvordan processen forløber, og samtidig gennem en særlig opstilling illustrere, hvordan regningerne kan omplantes til 10-talsystemet på en måde, der svarer temmelig nøje til sædvanlig division med papir og blyant. Læseren får herved at indtryk af, hvor nær maskinens metode i virkeligheden er ved en sådan sædvanlig division.

$C(AR_{0-39},\ MR_{1-39}) = T = 0.1011\ 0000 \sim +\ 0.6875$

$C(m) = N = 0.1110 \sim +\ 0.875$

	AR	MR
	0.1011	·0000
Venstreskift	1.0110	·0001
C(AR)-C(m)	1.0110	·0001
	1.0010	
$2 \cdot r_1$	0.1000	·0001
Venstreskift	1.0000	·0011
C(AR)-C(m)	1.0000	·0011
	1.0010	
$2^2 \cdot r_2$	0.0010	·0011
Venstreskift	0.0100	·0111
C(AR)-C(m)	0.0100	·0111
	1.0010	
$2^3 \cdot r_3$	1.0110	·0111
Venstreskift	0.1100	·1110
C(AR)+C(m)	0.1100	·1110
	0.1110	
$2^4 \cdot r_4$	1.1010	·1110

*) Man forstår lettest, at venstre side er lig $|2r_1|$, når man gør sig klart, at der efter venstreskiftet i AR altid opereres, så det, der tilføres C(AR), har modsat tegn af C(AR) selv.

	AR	MR	
Skift i MR	1.1010	1.1101	
$C(MR_0)$ inverteres	1.1010	0.1101	resultat: 0.1101

$$\underbrace{+0.875}\quad +0.6875\quad \underbrace{(1\cdot 1^0)+1\cdot\tfrac{1}{2}+1\cdot\tfrac{1}{4}+0\cdot\tfrac{1}{8}+1\cdot\tfrac{1}{16}} \sim (1).1101$$

$(1-2q_0)N\frac{1}{2}$	$\underline{-0.4375}$
r_1	+0.25000
$(1-2q_1)N\frac{1}{4}$	$\underline{-0.21875}$
r_2	+0.031250
$(1-2q_2)N\frac{1}{8}$	$\underline{-0.109375}$
r_3	-0.0781250
$(1-2q_3)N\frac{1}{16}$	$\underline{+0.0546875}$
r_4	-0.0234375

Position AR_{00} er ikke taget med i eksemplet, da den ikke deltager i divisionsprocessen og her hele tiden holdes nulstillet. For som vist ovenfor kan spild ikke fremkomme i det samlede resultat af et skridt, og man ville derfor aldrig få brug for denne positions indhold.

Ved at prøve med eksempler, hvor T og N har modsatte tegn, vil man se, hvordan metoden lader den negative kvotient's cifre komme i komplementær form. Dette skyldes, at noteringen retter sig efter, om restens fortegn på hvert trin er det "rigtige", og det rigtige fortegn er ved maskinens metode nævner's fortegn, noget der karakteriserer denne proces fremfor den sædvanlige division, hvor det er tællers fortegn, der er det "rigtige".

Ved fremstillingen her er der tænkt på lang division, men da man kan opfatte kort division som specialtilfælde heraf, nemlig de tilfælde, hvor dividendens sidste 39 binaler alle er 0, omfatter beskrivelsen i virkeligheden også kort division (jvfr. det viste eksempel).

2.8 Specielle forhold ved DASK-aritmetikken.

Dask-tallet -1 indtager på mange måder en særstilling, og giver i regneoperationerne ofte uventede resultater. Grunden hertil vil man som regel finde i det forhold, at det modsatte tal, +1, ikke findes i DASK-intervallet.

Fortegnsskifte udført på -1 ~ 1.00.....0 giver

00.11.....1

\+ 1

01.0.....0, altså -(-1)= -1, men samtidig er der spildindikation, hvilket ikke fremkommer ved fortegnsskifte på andre DASK-tal.

Tilsvarende fås at |-1| = -1 med spildindikation. -1 er det eneste tal i DASK-intervallet, hvis numeriske værdi er negativ.

Multiplikation af (-1) med (-1) giver, som man kan efterprøve, 01.0.....0, altså: (-1)·(-1)= -1 og samtidig spildindikation, hvilket ellers ikke fremkommer ved multiplikation.

Ved division bevirker blandt andet den omstændighed, at kvotientens sidste binal altid sættes til 1, at i tilfælde, hvor kvotienten kunne stå uafkortet i MR (når divisionen går op indenfor 39 binaler), vil den kun undtagelsesvis gøre det. Forholdene er forskellige ved kort og lang division, og vi går ikke ind på at udrede de ikke helt simple regler, der her gælder. Kun skal vi nævne, at

$$\frac{a}{-a} = \frac{-a}{a} = 1.0....01 \quad \text{og} \quad \frac{a}{a} = \frac{-a}{-a} = 0.11....11,$$

mens iøvrigt $\frac{b}{a}$ og $\frac{-b}{-a}$, stadig forudsat at divisionen går op indenfor registrets kapacitet, ikke giver eksakt samme resultat i MR. Denne sidste ejendommenlighed skyldes ikke, at q_{39} sættes til 1, men har sin grund i forhold svarende til at f.eks. decimalbrøken 0.3200... også kan skrives 0.3199..., idet maskinen, når divisor er positiv, vælger den første fremstillingsmåde, og når divisor er negativ den sidste.

3.1. Ordrens opbygning

Det blev i kap. 1 i korte træk forklaret, hvad vi forstår ved en ordre, og hvordan den er placeret i lageret. Vi kan resumere ved at sige, at en ordre er et mønster opbygget af to forskellige fysiske tilstande, fordelt udover en halvcelles 20 positioner, og mønstret virker ved, når det opfattes af KE, at give signalerne til en bestemt ordres udførelse. Da vi ved ordrer såvel som ved tal har valgt at betegne de to tilstande med 0 og 1, kunne vi simpelthen angive ordren ved en talværdi. Dette gør man dog ikke, da det ville give en dårlig adskillelse og notering af ordrens tre bestanddele: adresse, indeksmærke og operation, og vi skal nu se, hvordan man i stedet går frem.

Først må det nævnes, at selve den binære form - den, der kan siges at være direkte givet - selvfølgelig ikke er egnet; dertil er så lange og ens udseende rækker af 0 og 1 altfor uhåndterlige. Man opdeler i stedet mønstret i grupper, svarende til ordrens bestanddele, og hver gruppe noteres i et passende talsystem, nemlig således

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

1. tallet i position 1 - 11 (opfattet som binært heltal) noteres decimalt og nedskrives forrest.

2. position 0 og 12 slås sammen og noteres efter skemaet:

pos. 0	pos. 12	symbol
0	0	A
0	1	B
1	0	C
1	1	D

3. tallet i positionerne 13 - 19 (opfattet som binært heltal) noteres sedecimalt og nedskrives sidst.

Eks.: 1 00001011000 0 1010100 bliver til 88 C 54
0 01100100101 1 0011101 " " 805 B 1D

Dette kan se noget spredt og tilfældigt ud, og som ved alt, der har en udvikling bag sig, er det svært at give en god begrundelse for, at det er blevet netop, som vi her har vist. Vi skal dog

give en forklaring, i det omfang det er rimeligt.

C(pos. 1 - 11) giver, opfattet som binært heltal, simpelthen nummeret på den celle, det drejer sig om, dog når der ses bort fra indeksregistrets bidrag. Da der under kodearbejdet i høj grad forekommer optællinger etc. i forbindelse med adresserne, er det en fordel, at vi her noterer dem i det sædvanlige 10-talsystem (fremfor i et uvant system).

C(pos. 0 og 12), som angiver indeksregistret, skal i noteringen af rent praktiske grunde kunne skelnes fra den decimale adresse, der står lige foran; derfor kan de sædvanlige taltegn ikke bruges, og man har valgt A,B,C og D. Den spredte placering af de to cifre har en speciel og delvis historisk forklaring.

Pos. 13 - 19 er operationspositionerne, og deres indhold kunne ligeså godt være betegnet ved f.eks. korte bogstavnavne. Noteringen er derfor egentlig ligegyldig. Det sedecimale system har dog her en vis fordel fremfor f.eks. det decimale.

Inden vi i kapitel 4 går over til det vigtige punkt: at beskrive hvilket udvalg af operationer, DASK kan udføre, og hvilke betegnelser de har fået (de to sedecimale cifre i ordren), skal vi omtale nogle principielle og generelle forhold.

3.2. Betegnelser i tilknytning til adresser

Den skrevne adresse øges altid med indholdet i det benyttede indeksregister, og man må derfor i omtale skelne mellem:

den skrevne adresse, kaldet pseudoadressen

og den egentlige adresse, kaldet effektiv adresse.

Som det vil fremgå af operationslisten, er der operationer, der behandler indholdet i positionerne 0 - 11 som et hele, mens andre behandler indholdet i blot positionerne 1 - 11 som et hele, Derfor må man have navn for hver af disse positionsgrupper, sådan som det er vist side 1.5.

Ialt har vi følgende betegnelser:

pseudoadresse: indholdet i positionerne 1 - 11

effektiv adresse: pseudoadresse + indholdet i det aktuelle indeksregister

adressedel: indholdet i positionerne 0 - 11

adressepositioner: positionerne 1 - 11.

I ordren 88 C 54 er

pseudoadressen : 88

effektiv adresse: 88 + C(IRC)

som adressedel står: 100001011000

i adressepositionerne står: 00001011000

3.3. Transporter til og fra lagerceller

Et flertal af maskinens egentlige regneoperationer består i, eller har som bestanddel, at en celles indhold, x, sendes på forskellig måde til AR: for eksempel som x, |x|, -x etc. Her gælder altid, at den afsendende celle i lageret vil have samme indhold før som efter afsendelsen; den er altså uforandret.

I operationer, hvor en lagercelle er det modtagende register, gælder, at dens hidtige indhold slettes og erstattes med det tilsendte. *)

3.4. Varianter indenfor operationerne

AR vil være impliceret i de allerfleste operationer, og i mange af disse har man mulighed for at variere det, der sker, ved at nulstille eller ikke nulstille AR før operationen udføres.

Er AR modtagende register, vil dette kunne føre til, at der, når AR slettes, kommer en ren transport (af x, |x|, -x etc.) til AR, mens der, når AR ikke slettes, sker en addition af C(AR) og det tilsendte. Er AR det afsendende register, opnår man ved forudgående sletning noget ret specielt, nemlig at afsende 0, hvilket dog langtfra er så betydningsløst, som det måske straks kunne se ud til.

---

*) Ved operationer, der synes i modstrid med dette, er der i virkeligheden tale om en sammenkædning af flere enkelt-operationer.

En anden måde, operationer kan varieres på, har til formål at kunne udføre regninger med tal (eller ordrer), som kun behøver at fylde 20 positioner. Ved mange opgaver er det tilstrækkeligt at operere med tal af denne længde, og herved bliver det muligt at nøjes med halvceller ved tallenes lagring, og således få dobbelt så stor plads til talmaterialet. Men for at regningerne skal gå glat, må f.eks. tal i lagerets højrehalvceller*) kunne transporteres til ARv, sådan at deres komma, som jo står mellem positionerne 20 og 21, kommer mellem positionerne 0 og 1. Foruden at indholdet i en helcelle på sædvanlig måde føres over i hele AR, har man derfor operationer, hvor indholdet i

en hhac overføres i ARh (AR_{20-39}) og omvendt
en vhac " " ARv (AR_{0-19}) " "
en $\underline{\text{h}}$hac " " AR$\underline{\text{v}}$ (AR_{0-19}) " " ,

og tilsvarende variationer ved ordrer, som division o. lign., der ikke direkte "overfører".

Det er her vigtigt at slå fast, at operationer, som benytter korrespondancen mellem en højrehalvcelle og ARh, normalt ikke bruges, da det, bedømt ud fra talværdien C(hhac m), kan føre til uventede, og i denne forbindelse, gale resultater. Kalder vi cifrene i højrehalvcelle m,

$$s_0, s_1, s_2, \ldots, s_{19} ,$$

således at $C(\text{hhac m}) = \sum_{j=0}^{19} s_j \cdot 2^{-j} - 2 \cdot s_0$, vil maskinen nemlig operere med følgende tal af den sædvanlige længde (40 bits):

ved additioner med $0.00\ldots0s_0s_1\ldots s_{19}$

ved subtraktioner med $1.11\ldots1\bar{s}_0\bar{s}_1\ldots\bar{s}_{19} + 2^{-39}$.

Dette er tal, som henholdsvis har værdierne $+2^{-20}(C(\text{hhac m})+2s_0)$ og $-2^{-20}(C(\text{hhac m}) + 2s_0)$ og derfor kun når $s_0 = 0$ ($C(\text{hhac m})$ positivt) svarer til det, man skulle forvente: $2^{-20}(C(\text{hhac m}))$. Ved operationer som overførsel af $\pm|x|$ fra højrehalvceller til ARh er det på tilsvarende måde tallet $\pm\,|0.00\ldots0s_0s_1\ldots s_{19}|$ $= \pm\, 0.00\ldots0s_0s_1\ldots s_{19}$ der overføres.

---

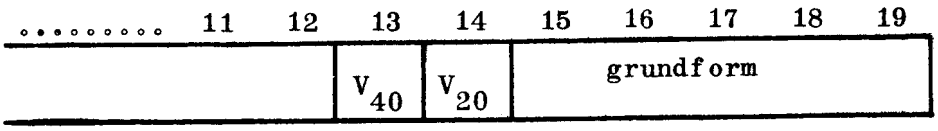
*) Talværdien af disse er selvfølgelig $\sum_{j=20}^{39} p_j \cdot 2^{20-j} - 2 \cdot p_{20}$.

Før vi gennem eksempler gør nøjere rede for, hvad der sker, skal vi se, hvordan varianterne viser sig i operationsdelens 7 binære cifre.

3.5. Operationers grundform og deres varianter

En operations grundform er bestemt af de 5 sidste positioner, 15 - 19 (35 - 39), og har 0 i positionerne 13 og 14 (der er således mulighed for 32 grundformer).

Vi får brug for en kort betegnelse for indholdet i position 13 og 14 og kalder disse cifre henholdsvis V_{40} og V_{20}.



Udfra grundformen fås nu varianterne ved at sætte V_{20}, V_{40} eller begge til 1. Vi ser, at varianterne, som har $V_{20} = 1$, fås af grundformen ved at addere 20 (sedecimalt!), varianter, der har $V_{40} = 1$, ved at addere 40, og de der har $V_{40} = V_{20} = 1$ ved at addere 60.*)

Funktionen af V_{40} er:

$V_{40} = 1$ sletter AR før operationen, mens

$V_{40} = 0$ lader oprindeligt C(AR) tage del i processen.

Funktionen af V_{20} er:

$V_{20} = 0$ og adressen lige lader helcelle korrespondere med hele AR

$V_{20} = 0$ og adressen ulige lader den højrehalvcelle, adressen angiver, korrespondere med ARh

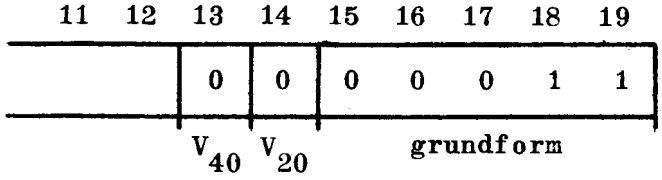
$V_{20} = 1$ lader altid en halvcelle korrespondere med ARv, og naturligvis den halvcelle, adressen angiver, d.v.s.: ved lige adresse venstrehalvcelle, ved ulige adresse højrehalvcelle.

---

*) Havde man valgt at notere ordrens operationscifre i 10-talsystemet, skulle der have været adderet henholdsvis 32, 64, og 96, hvilket er knap så simpelt.

3.6. Eksempler med en operation og dens varianter

Vi vælger den operation, der hedder "subtraher numerisk". Den bevirker, at indholdet, x, i en celle overføres som $-|x|$ til AR. Grundformen er



, og noteres altså 03.

V_{40}-varianterne.

Bruges grundformen, får vi som resultat, at der i AR kommer:

$$C(AR) - |x| .$$

Sættes $V_{40} = 1$, hvorved operationen noteres 43, får vi som resultat

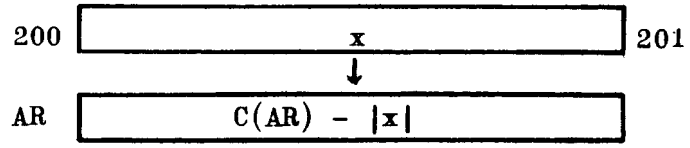
$-|x|$ i AR, altså en ren transport af $-|x|$ til AR.

V_{20}-varianterne vil vi illustrere ved at tage følgende 4 ordrer igennem:

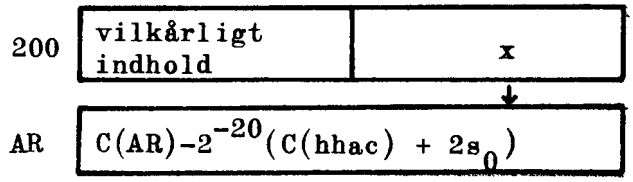
1. 200 A 03, adressen er lige, $V_{20} = 0 (= V_{40})$. Vi får

$$C(AR) - |C(200)| \text{ i AR,}$$

hvilket vi anskueliggør således:

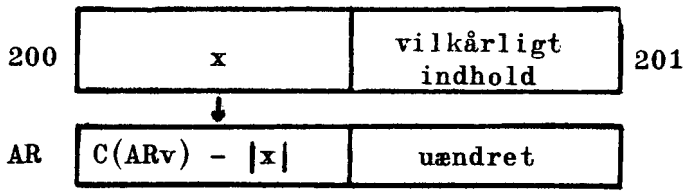


2. 201 A 03, her er adressen ulige, $V_{20} = 0 (= V_{40})$. Vi får:

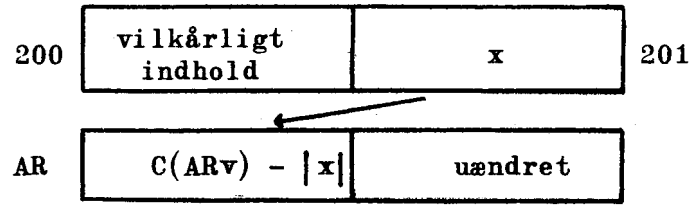


Obs! $C(AR_{0-19})$ i almindelighed ikke uændret: "mente-overføring".

3. 200 A 23, adressen er lige, $V_{20} = 1$, $(V_{40} = 0)$. Vi får:



4. 201 A 23, adressen ulige, $V_{20} = 1$, ($V_{40} = 0$)



Det vil være let at udlede, hvordan det går, når man i disse fire ordrer sætter $V_{40} = 1$, altså når ordrerne hedder: 200 A 43, 201 A 43, 200 A 63 og 201 A 63. Det vil også uden vanskelighed forstås, hvad der sker i de operationer, hvor transporten går fra AR til lageret (en menteoverføring i den "urørte" halvcelle, som under 2, kan naturligvis aldrig forekomme her).

Vi understreger, at vi i eksemplet af let forståelige grunde har valgt indeksregister A; havde vi valgt f.eks. B, ville det have drejet sig om, hvorvidt

$\left.\begin{matrix}200\\201\end{matrix}\right\}$ + C(IRB) - nemlig den <u>effektive adresse</u> - var lige eller ulige.

3.7. Afvigelser

Fra de regler, der er opstillet i det foregående, er der nogle undtagelser. Der er tilfælde, hvor den beskrevne virkning af V_{20}- og V_{40}-værdierne ingen mening har, og man har så i maskinen enten ladet en helt ny operation komme i stand i stedet for varianten, eller man har indrettet det således, at spørgsmålet om disse cifre kan negligeres : der sker det samme, hvilken værdi de end har.

Af afvigelser skal vi kun nævne to grupper.

<u>$V_{20} = 1$ i hopordrer</u> bevirker, at maskinen <u>stopper før</u> hoppet, <u>som først vil blive</u> udført, når der igen trykkes på startknappen.

<u>$V_{40} = 1$</u> vil i en del operationer bevirke, at de forløber i tilknytning til den lange akkumulator (AR_{0-39}, MR_{1-39}).

KAPITEL 4.

Operationsliste I.

På de følgende sider omtales samtlige operationer, der ikke vedrører de ydre enheder. På grund af den udførlige beskrivelse, vi har givet af hver enkelt operation, består denne operationsliste af ikke mindre end 19 sider, hvoraf følger, at det vil være besværligt for koderen at benytte dette kapitel i det praktiske arbejde med nedskrivningen af koden. Til det formål er der derfor udarbejdet et særligt skema over operationerne. Det er imidlertid nødvendigt, at koderen ved besked om alt, hvad der sker under de enkelte operationer, og vi må derfor råde læseren til at arbejde de følgende sider grundigt igennem.

Bemærkninger.

1. Som det vil fremgå af listen, beskrives forskellige varianter af samme grundform hver for sig. For at afgøre, hvilken variant der skal benyttes, må man læse beskrivelsen af operationens virkning.

2. For hver operation nævnes de registre i AE, de indeksregistre samt de celler, hvis indhold eventuelt ændres under operationens udførelse. Det er herved underforstået, at indholdet i de øvrige celler, registre i AE og indeksregistre ikke ændres.

3. Operationstiden er angivet i AT. 1 AT = 56 μs = den tid, der medgår til en addition i AE.

Addér til AR: +; AR	Virkning: Er m lige vil C(hec m) adderes til C(AR). Resultatet i AR.
Operationsbetegnelse: 00	
Kode: n,I,00	Er m ulige anvendes 00 normalt ikke; iøvrigt vil $C(AR) + 2^{-20} \cdot \left[C(hhac\ m) + 2 \cdot p_{20} \right]$ fremkomme i AR.
	Spild kan forekomme.
	Indholdet ændres i: AR.
	Tid: 1 AT.

Addér til ARv: +;ARv	Virkning: Er m lige vil C(vhac m) adderes til C(AR). Resultatet i AR.
Operationsbetegnelse: 20	
Kode: n,I,20	Er m ulige vil C(hhac m) adderes til C(AR). Resultatet i AR. C(ARh) uændret i begge tilfælde.
	Spild kan forekomme.
	Indholdet ændres i: AR.
	Tid: 1 AT.

Addér til tom AR: 0;+; AR	Virkning: Som ved 00, idet AR først nulstilles.
Operationsbetegnelse: 40	Aldrig spild.
Kode: n,I,40	Indholdet ændres i: AR.
	Tid: 1 AT.

Addér til tom AR: 0;+;ARv	Virkning: Som ved 20, idet AR først nulstilles.
Operationsbetegnelse: 60	Aldrig spild.
Kode: n,I,60	Indholdet ændres i: AR
	Tid: 1 AT.

NB. m betegner den effektive adresse; m = n+C(IR)

Subtrahér fra AR: -;AR	Virkning: Er m lige vil -C(hec m) adderes til C(AR). Resultatet i AR. Er m ulige anvendes 01 normalt ikke; iøvrigt vil $C(AR)-2^{-20}$. $\left[C(hhac\ m)+2\cdot p_{20}\right]$ fremkomme i AR. Spild kan forekomme. Indhold kan ændres i: AR. Tid: 1 AT.
Operationsbetegnelse: 01	
Kode: n,I,01	

Subtrahér fra ARv: -;ARv	Virkning: Er m lige vil -C(vhac m) adderes til C(AR). Resultatet i AR. Er m ulige vil -C(hhac m) adderes til C(AR). Resultatet i AR. C(ARh) uændret i begge tilfælde. Spild kan forekomme. Indholdet ændres i: AR. Tid: 1 AT.
Operationsbetegnelse: 21	
Kode: n,I,21	

Subtrahér fra tom AR: 0;-;AR	Virkning: Som ved 01, idet AR først nulstilles. Kun spild for -(-1). Indholdet ændres i: AR. Tid: 1 AT.
Operationsbetegnelse: 41	
Kode: n,I,41	

Subtrahér fra tom AR: 0;-;ARv	Virkning: Som ved 21, idet AR først nulstilles. Kun spild for -(-1). Indholdet ændres i: AR. Tid: 1 AT.
Operationsbetegnelse: 61	
Kode: n,I,61	

NB. m betegner den effektive adresse; m = n+C(IR)

Addér numerisk til AR: +;||;AR	Virkning: Er m lige vil |C(hec m)| adderes til C(AR). Resultatet i AR. Er m ulige anvendes 02 normalt ikke; iøvrigt vil $C(AR)+2^{-20}\lceil C(hhac\ m) + 2\cdot p_{20}\rceil$ fremkomme i AR. Spild kan forekomme. Indholdet ændres i : AR. Tid: 1 AT.

Operationsbetegnelse: 02
Kode: n,I,02

Addér numerisk til ARv: +;||;ARv	Virkning: Er m lige vil |C(vhac m)| adderes til C(AR). Resultatet i AR. Er m ulige vil |C(hhac m)| adderes til C(AR). Resultatet i AR. C(ARh) uændret i begge tilfælde. Spild kan forekomme. Indholdet ændres i : AR. Tid: 1 AT.

Operationsbetegnelse: 22
Kode: n, I, 22

Addér numerisk til tom AR: 0;+;||;AR	Virkning: Som ved 02, idet AR først nulstilles. Kun spild for |-1| Indholdet ændres i : AR. Tid: 1 AT.

Operationsbetegnelse: 42
Kode: n, I, 42

Addér numerisk til tom AR: 0;+;||;ARv	Virkning: Som ved 22, idet AR først nulstilles. Kun spild for |-1| Indholdet ændres i: AR. Tid: 1 AT.

Operationsbetegnelse: 62
Kode: n,I,62

NB. m betegner den effektive adresse; m = n+C(IR).

NB. |-1| = -1, altså negativ i DASK

Subtrahér numerisk fra AR: -;||;AR

Operationsbetegnelse: 03

Kode: n,I,03

Virkning: Er m lige vil $-|C(\text{hec } m)|$ adderes til C(AR). Resultatet i AR. Er m ulige anvendes 03 normalt ikke. Iøvrigt vil C(AR) $-2^{-20}\left[C(\text{hhac } m)+2p_{20}\right]$ fremkomme i AR.

Spild kan forekomme.

Indholdet ændres i : AR.

Tid: 1 AT.

Subtrahér numerisk fra ARv: -;||;ARv

Operationsbetegnelse: 23

Kode: n,I,23

Virkning: Er m lige vil $-|C(\text{vhac } m)|$ adderes til C(AR). Resultatet i AR. Er m ulige vil $-|C(\text{hhac } m)|$ adderes til C(AR). Resultatet i AR. C(ARh) uændret i begge tilfælde.

Spild kan forekomme.

Indholdet ændres i: AR.

Tid: 1 AT.

Subtrahér numerisk fra tom AR: 0;-;||;AR

Operationsbetegnelse: 43

Kode: n,I,43

Virkning: Som ved 03, idet AR først nulstilles.

Aldrig spild.

Indholdet ændres i : AR.

Tid: 1 AT.

Subtrahér numerisk fra tom AR: 0;-;||;ARv

Operationsbetegnelse: 63

Kode: n,I,63

Virkning: som ved 23, idet AR først nulstilles.

Aldrig spild.

Indholdet ændres i: AR.

Tid: 1 AT.

NB. m betegner den effektive adresse; m = n+C(IR).

Addér til AR og sæt i MR: +;AR;MR	Virkning: Er m lige vil C(hec m) adderes til C(AR). Summen i AR. Helordet i AR derefter til MR. Er m ulige anvendes 04 normalt ikke. Iøvrigt vil $C(AR)+2^{-20}$. $\left[C(hhac\ m)+2p_{20}\right]$ fremkomme i AR og i MR.
Operationsbetegnelse: 04	Spild kan forekomme.
Kode: n,I,04	Indholdet ændres i: MR og AR.
	Tid: 1 AT.

Addér til ARv og sæt i MR: +;ARv;MR	Virkning: Er m lige vil C(vhac m) adderes til C(AR). Summen i AR. Helordet i AR derefter til MR. Er m ulige vil C(hhac m) adderes til C(AR). Summen i AR. Helordet i AR derefter til MR. C(ARh) uændret i dette tilfælde.
Operationsbetegnelse: 24	Spild kan forekomme.
Kode: n,I,24	Indholdet ændres i: MR og AR.
	Tid: 1 AT.

Addér til tom AR og sæt i MR: 0;+;AR;MR	Virkning: Som ved 04, idet AR først nulstilles. Aldrig spild.
Operationsbetegnelse: 44	Indholdet ændres i: MR og AR.
Kode: n,I,44	Tid: 1 AT.

Addér til tom AR og sæt i MR: 0;+;ARv;MR	Virkning: Som ved 24, idet AR først nulstilles. Aldrig spild.
Operationsbetegnelse: 64	Indholdet ændres i: MR og AR.
Kode: n,I,64	Tid: 1 AT.

NB. m betegner den effektive adresse; m = n+C(IR)

Subtrahér fra AR og sæt i MR: -;AR;MR

Operationsbetegnelse: 05

Kode: n,I,05

Virkning: Er m lige vil -C(hec m) adderes til C(AR). Resultatet i AR. Helordet i AR derefter til MR.
Er m ulige anvendes 05 normalt ikke. Iøvrigt vil $C(AR)-2^{-20}\left[C(hhac\ m)+2p_{20}\right]$ fremkomme i AR og i MR.

Spild kan forekomme.

Indholdet ændres i: MR og AR.

Tid: 1 AT.

Subtraher fra ARv og sæt i MR: -;ARv;MR

Operationsbetegnelse: 25

Kode: n,I,25

Virkning: Er m lige vil -C(vhac m) adderes til C(AR). Resultatet i AR. Helordet i AR derefter til MR.
Er m ulige vil -C(hhac m) adderes til C(AR). Summen i AR. Helordet i AR derefter til MR.
C(ARh) uændret i begge tilfælde.

Spild kan forekomme.

Indholdet ændres i: MR og AR.

Tid: 1 AT.

Subtrahér fra tom AR og sæt i MR: 0;-;AR;MR

Operationsbetegnelse: 45

Kode: n,I,45

Virkning: Som ved 05, idet AR først nulstilles.

Kun spild for -(-1)

Indholdet ændres i: MR og AR.

Tid: 1 AT.

Subtrahér fra tom AR og sæt i MR: 0,-,ARv;MR

Operationsbetegnelse: 65

Kode: n,I,65

Virkning: Som ved 25, idet AR først nulstilles.

Kun spild for -(-1)

Indholdet ændres i: MR og AR.

Tid: 1 AT.

NB. m betegner den effektive adresse; m = n+C(IR)

Adder til AR og læs ud: +;AR;L

Operationsbetegnelse: 06

Kode: n,I,06

Virkning: Er m lige vil
C(hec m) adderes til C(AR).
Summen i AR. Helordet i AR
derefter til hec m i L.
Er m ulige anvendes 06 normalt ikke.
Iøvrigt vil $C(AR)+2^{-20}\left[C(hhac\ m)+2p_{20}\right]$
fremkomme i AR.
C(ARh) derefter til hhac m i L.

Spild kan forekomme.

Indholdet ændres i: AR, samt for
m lige: hec m i L,
m ulige: hhac m i L.

Tid: $1\frac{1}{2}$ AT.

Adder til ARv og læs ud:+;ARv;L

Operationsbetegnelse: 26

Kode: n,I,26

Virkning: Er m lige vil
C(vhac m) adderes til C(AR).
Summen i AR.C(ARv) derefter
til vhac m i L.
Er m ulige vil
C(hhac m) adderes til C(AR).
Summen i AR. C(ARv) derefter
til hhac m i L.
C(ARh) uændret i begge tilfælde.

Spild kan forekomme.

Indholdet ændres i: hac m i L og i AR.

Tid: $1\frac{1}{2}$ AT.

NB. m betegner den effektive adresse; m = n+C(IR)

Øg adressedel med 2: Øg adr.

Operationsbetegnelse: 46

Kode: n,I,46

Virkning: Er m lige vil
C(hec m)+ 2^{-10} + 2^{-30} fremkomme i AR.
Helordet i AR derefter til hec m i L.
Pseudoadresserne i begge halvord i
hec m er da øget med 2.*)
Er m ulige vil
C(hhac m)+ 2^{-10} fremkomme i ARh (ARv tom).
C(ARh) derefter til hhac m i L.
Pseudoadressen i halvordet i hhac m
er da øget med 2.*)
*) Er pseudoadressen 2046 eller 2047
vil forøgelser ændre denne til 0
eller 1; men samtidig vil
indeksmærkningen ændres: IRA bliver
til IRC, IRB til IRD, IRC til IRA
og IRD til IRB. De to sidste indeks-
ændringer bevirker menteoverføring
til AR_{00} og/eller AR_{19}.

Indholdet ændres i: AR, samt for m lige: hec m i L,
m ulige: hhac m i L.

Tid: $1\frac{1}{2}$ AT.

Øg adressedel med 2 via ARv
Øg adr: ARv

Operationsbetegnelse: 66

Kode: n,I,66

Virkning: Er m lige vil
C(vhac m) + 2^{-10} fremkomme i ARv (ARh tom).
C(ARv) derefter til vhac m i L.
Pseudoadressen i halvordet i vhac m
er da øget med 2.*)
Er m ulige vil
C(hhac m) + 2^{-10} fremkomme i ARv (ARh tom).
C(ARv) derefter til hhac m i L.
Pseudoadressen i halvordet i hhac m er da
øget med 2.*)
*) Er pseudoadressen 2046 eller 2047
vil forøgelsen ændre denne til 0 eller 1;
men samtidig vil indeksmærkningen
ændres: IRA bliver til IRC, IRB til IRD,
IRC til IRA og IRD til IRB. De to
sidste indeksmærkninger bevirker mente-
overføring til AR_{00}.

Indholdet ændres i: AR, samt for m lige: vhac m i L,
m ulige: hhac m i L.

Tid: $1\frac{1}{2}$ AT.

C(MR) til AR: MR;AR	Virkning: C(MR) fremkommer i AR. C(AR_{00}) sættes = 0.
Operationsbetegnelse: 07 el. 27	Indholdet ændres i: AR.
Kode: x,x,07 eller x,x,27	Tid: 6 AT.

C(MR) logisk produkt med C(AR) MR∘AR	Virkning: Er C(AR_j) = C(MR_j) = 1 kommer der 1 i AR_j; j = 0,1;...;39. I de øvrige positioner sættes 0. C(AR_{00}) sættes = 0.
Operationsbetegnelse: 47 el. 67	Indholdet ændres i: AR.
Kode: x,x,47 eller x,x,67	Tid: 6 AT.

Læs ud til celle: AR; L	Virkning: Er m lige vil C(AR) fremkomme i hec m. Er m ulige vil C(ARh) fremkomme i hhac m.
Operationsbetegnelse: 08	Indholdet ændres i: m lige: hec m, m ulige: hhac m.
Kode: n,I,08	Tid: 1 AT.

Læs ud til celle fra ARv: ARv;L	Virkning: Er m lige vil C(ARv) fremkomme i vhac m. Er m ulige vil C(ARv) fremkomme i hhac m.
Operationsbetegnelse: 28	Indholdet ændres i: m lige: vhac m, m ulige: hhac m.
Kode: n,I,28	Tid: 1 AT.

Nul til celle: 0;AR;L	Virkning: AR nulstilles; derefter som 08.
Operationsbetegnelse: 48	Indholdet ændres i: m lige: hec m, m ulige: hhac m, samt i AR.
Kode: n,I,48	Tid: 1 AT.

Nul til celle fra ARv: 0;ARv;L	Virkning: AR nulstilles; derefter som 28
Operationsbetegnelse: 68	Indholdet ændres i: m lige: vhac m, m ulige: hhac m, samt i AR.
Kode: n,I,68	Tid: 1 AT.

NB. I ordrer af formen x,x,0P angiver x,x, at adressen er irrelevant. Koderen skal dog altid udfylde disse pladser (f.eks. med 0,A).

NB. m betegner den effektive adresse; m = n + C(IR)

Læs adressedel ud til celle: ARadr;L

Operationsbetegnelse: 09

Kode: n,I,09

Virkning: Er m lige
vil $C(AR_{0-11})$ og $C(AR_{20-31})$ fremkomme i pos. 0-11 og pos. 20-31 i hec m.
Er m ulige
vil $C(AR_{20-31})$ fremkomme i pos. 20-31 i hhac m.
Indholdet ændres i:
m lige: pos. 0-11 og pos. 20-31 i hec m,
m ulige: pos. 20-31 i hhac m.
Tid: 1 AT.

Læs adressedel ud til celle via ARv: ARvadr;L

Operationsbetegnelse: 29

Kode: n,I,29

Virkning: Er m lige
vil $C(AR_{0-11})$ fremkomme i pos. 0-11 i vhac m.
Er m ulige
vil $C(AR_{0-11})$ fremkomme i pos. 20-31 i hhac m.
Indholdet ændres i:
m lige: pos. 0-11 i vhac m,
m ulige: pos.20-31 i hhac m.
Tid: 1 AT.

Nul til adressedel i celle: 0;ARadr;L

Operationsbetegnelse: 49

Kode: n,I,49

Virkning: Som ved 09, idet AR først nulstilles.
Indholdet ændres i: AR,samt for
m lige: pos. 0-11 og pos. 20-31 i hec m,
m ulige: pos. 20-31 i hhac m,
Tid: 1 AT.

Nul til adressedel i celle via ARv: 0;ARvadr;L

Operationsbetegnelse: 69

Kode: n,I,69

Virkning: som ved 29, idet AR først nulstilles.
Indholdet ændres i: AR,samt for
m lige: pos. 0-11 i vhac m,
m ulige: pos.20-31 i hhac m.
Tid: 1 AT.

NB. m betegner den effektive adresse, m = n+C(IR)

Afrundet multiplikation: · afr

Operationsbetegnelse: 0A

Kode: n,I,0A

Virkning: Er m lige vil $C(\text{hec m}) \cdot C(MR) + 2^{-40}$ fremkomme afkortet i AR.
Er m ulige anvendes 0A normalt ikke. Iøvrigt vil $[C(\text{hhac m}) + 2\ p_{20}] \cdot 2^{-20} \cdot C(MR) + 2^{-40}$ fremkomme afkortet i AR.
Kun spild for (-1)(-1) AR.
Indholdet ændres i: $6\frac{1}{2}$ AT.
Tid:

Afrundet multiplikation: · afr

Operationsbetegnelse: 2A

Kode: n,I,2A

Virkning: Er m lige vil $C(\text{vhac m}) \cdot C(MR) + 2^{-40}$ fremkomme afkortet i AR.
Er m ulige vil $C(\text{hhac m}) \cdot C(MR) + 2^{-40}$ fremkomme afkortet i AR.
Kun spild for (-1)(-1) AR.
Indholdet ændres i: $6\frac{1}{2}$ AT.
Tid:

Uafkortet multiplikation: · uafk

Operationsbetegnelse: 4A

Kode: n,I,4A

Virkning: Er m lige vil $C(\text{hec m}) \cdot C(MR)$ fremkomme i AR_{0-39}, MR_{1-39}; $C(MR_0) = 0$.
Er m ulige anvendes 4A normalt ikke. Iøvrigt vil $[C(\text{hhac m}) + 2p_{20}] \cdot 2^{-20} \cdot C(MR)$ fremkomme i AR_{0-39}, MR_{1-39}; $C(MR_0) = 0$.
Kun spild for (-1)(-1) AR og MR.
Indholdet ændres i: $6\frac{1}{2}$ AT.
Tid:

Uafkortet multiplikation: · uafk

Operationsbetegnelse: 6A

Kode: n,I,6A

Virkning: Er m lige vil $C(\text{vhac m}) \cdot C(MR)$ fremkomme i AR_{0-39}, MR_{1-39}; $C(MR_0) = 0$.
Er m ulige vil $C(\text{hhac m}) \cdot C(MR)$ fremkomme i AR_{0-39}, MR_{1-39}; $C(MR_0) = 0$.
Kun spild for (-1)(-1) AR og MR.
Indholdet ændres i: $6\frac{1}{2}$ AT.
Tid

NB. m betegner den effektive adresse; m = n+C(IR)

NB. I DASK bliver (-1) · (-1) = (-1)

Kort division: :kort	KRAV: $\lvert C(AR)\rvert \leq \lvert divisor\rvert$; kun når dette er opfyldt, får vi følgende
Operationsbetegnelse: 0B	Virkning: Er m lige vil
Kode: n,I,0B	C(AR): C(hec m) fremkomme i MR.
	Divisionsresten$\cdot 2^{39}$ står i AR.
	Er m ulige anvendes 0B normalt ikke;
	iøvrigt udføres divisionsprocessen
	med divisor lig $[C(hhac\ m)+2p_0]\cdot 2^{-20}$.
	Indholdet ændres i: MR og AR.
	Tid: $6\frac{1}{2}$ AT.

Kort division: :kort	KRAV: $\lvert C(AR)\rvert \leq \lvert divisor\rvert$; kun når dette er opfyldt, får vi følgende
Operationsbetegnelse: 2B	Virkning: Er m lige vil
Kode: n,I,2B	C(AR):C(vhac m) fremkomme i MR.
	Divisionsresten$\cdot 2^{39}$ står i AR.
	Er m ulige vil
	C(AR): C(hhac m) fremkomme i MR.
	Divisionsrest$\cdot 2^{39}$ står i AR.
	Indholdet ændres i: MR og AR.
	Tid: $6\frac{1}{2}$ AT.

NB. m betegner den effektive adresse; m = n + C(IR)

Lang division: :lang

Operationsbetegnelse: 4B

Kode: n,I,4B

KRAV: $|C(AR_{0-39},MR_{1-39})| \leq |\text{divisor}|$; kun når dette er opfyldt, får vi følgende

Virkning: Er m lige vil $C(AR_{0-39},MR_{1-39})$: C(hec m) fremkomme i MR. Divisionsresten$\cdot 2^{39}$ står i AR.

Er m ulige anvendes 4B normalt ikke; iøvrigt udføres divisionsprocessen med divisor lig $[C(\text{hhac m})+2p_0] \cdot 2^{-20}$.

Indholdet ændres i: AR og MR.

Tid: $6\frac{1}{2}$ AT.

Lang division: :lang

Operationsbetegnelse: 6B

Kode: n,I,6B

KRAV: $|C(AR_{0-39},MR_{1-39})| \leq |\text{divisor}|$; kun når dette er opfyldt, får vi følgende

Virkning: Er m lige vil $C(AR_{0-39},MR_{1-39})$: C(vhac m) fremkomme i MR. Divisionsresten$\cdot 2^{39}$ står i AR.

Er m ulige vil $C(AR_{0-39},MR_{1-39})$: C(hhac m) fremkomme i MR. Divisionsresten$\cdot 2^{39}$ står i AR.

Indholdet ændres i: AR og MR.

Tid: $6\frac{1}{2}$ AT.

NB. m betegner den effektive adresse; m = n + C(IR)

Venstre skift, kort: vsk

Operationsbetegnelse: 0C

Kode: n,I,0C

Virkning: C(AR) skiftes m positioner til venstre. Dette sker mod.128(eks. er m=1291=128·10+11 skiftes 11 positioner til venstre). I AR_{39} indskiftes nuller. Fra AR_0 skiftes cifrene til AR_{00}.

Indholdet ændres i: AR.

NB. Eventuelt spildindikation: $C(AR_{00}) \neq C(AR_0)$. *)

Tid: $(1 + \frac{1}{2} \cdot$ antal påbegyndte grupper á 4 skift)AT.

Venstre skift, langt: langt vsk

Operationsbetegnelse: 4C

Kode: n,I,4C

Virkning: $C(AR_{0-39}, MR_{1-39})$ skiftes m positioner til venstre (mod.128). I MR_{39} indskiftes nuller. Fra AR_0 skiftes cifrene til AR_{00}. $C(MR_0)=0$.

Indholdet ændres i: AR og MR_{1-39}.

NB. Eventuelt spildindikation: $C(AR_{00}) \neq C(AR_0)$.

Tid: $(1\frac{1}{2} + \frac{1}{2} \cdot$ antal påbegyndte grupper á 4 skift) *) AT.

Højreskift m.tegn, kort: hsk. m. tegn

Operationsbetegnelse: 0D

Kode: n,I,0D

Virkning: C(AR) skiftes m positioner til højre(mod.128). I AR_0 indskiftes oprindeligt fortegn; $C(AR_{00}) = 0$.

Indholdet ændres i: AR.

NB. Eventuelt spildindikation: $C(AR_{00}) \neq C(AR_0)$. *)

Tid: $(1 + \frac{1}{2} \cdot$ antal påbegyndte grupper á 4 skift) AT.

Højreskift m.tegn, langt: langt hsk m. tegn

Operationsbetegnelse: 4D

Kode: n,I,4D

Virkning: $C(AR_{0-39}, MR_{1-39})$ skiftes m positioner til højre (mod. 128). I AR_0 indskiftes oprindeligt fortegn; $C(MR_0)=0$, og $C(AR_{00})=0$.

Indholdet ændres i: AR og MR.

NB. Eventuelt spildindikation: $C(AR_{00}) \neq C(AR_0)$. *)

Tid: $(1\frac{1}{2} + \frac{1}{2} \cdot$ antal påbegyndte grupper á 4 skift)AT.

NB: Uanset om m opfattes som adresse eller ej, er m = n + C(IR).

---

*) Den første gruppe består af 0,1,2 eller 3 skift, og derefter fortsættes med 4 skift i hver gruppe.

Normalisering, kort: norm.

Operationsbetegnelse: 0E

Kode: n,I,0E

Virkning: C(AR) skiftes til venstre indtil

enten $-1 \leq$ nyt C(AR) $< -\frac{1}{2}$

eller nyt C(AR) = 0

eller $\frac{1}{2} \leq$ nyt C(AR) < 1.

Der udføres højst 40 skift. I AR_{39} indskiftes nuller.

$C(AR_0)$ skiftes ind i AR_{00}.

Antallet af skift placeres:

For m lige i vhac m i pos. 5-11;0 til pos. 0-4.

For m ulige i hhac m i pos.25-31;0 til pos.20-24.

De øvrige positioner i m urørte.

Indholdet ændres i: AR, samt for

m lige:pos 0-11 i vhac m,

m ulige:pos 20-31 i hhac m.

Tid: $(1+\frac{1}{2}$ · antal påbegyndte grupper á 4 skift)*) AT.

Normalisering, lang: lang norm.

Operationsbetegnelse: 4E

Kode: n,I,4E

Virkning: $C(AR_{0-39}, MR_{1-39})$ skiftes til venstre. Iøvrigt helt som ved 0E med de ændringer, at der her højst udføres 80 skift, og at nullerne her indskiftes i MR_{39}. 0 til MR_0.

Indholdet ændres i: AR og MR, samt for

m lige:pos. 0-11 i vhac m,

m ulige:pos.20-31 i hhac m.

Tid: $(1\frac{1}{2}+\frac{1}{2}$· antal påbegyndte grupper á 4 skift)*) AT.

Højreskift u.tegn,kort: hsk.u.tegn

Operationsbetegnelse: 0F

Kode: n,I,0F

Virkning: C(AR) skiftes m pos. til højre(mod 128). I AR_0 indskiftes nuller; $C(AR_{00})=0$.

Indholdet ændres i: AR.

Tid: $(1+\frac{1}{2}$·antal påbegyndte grupper á 4 skift)*) AT.

Højreskift u.tegn, langt: langt hsk.u.tegn

Operationsbetegnelse: 4F

Kode: n,I,4F

Virkning: $C(AR_{0-39}, MR_{1-39})$ skiftes m pos.til højre (mod 128).

I AR_0 indskiftes nuller. $C(AR_{00})=C(MR_0)=0$.

Indholdet ændres i: MR og AR.

Tid: $(1\frac{1}{2}+\frac{1}{2}$· antal påbegyndte grupper á 4 skift)*) AT.

NB. m betegner den effektive adresse; m = n + C(IR).

*) Den første gruppe består af 0,1,2 eller 3 skift, og derefter fortsættes med 4 skift i hver gruppe.

Hop: Hop	Virkning: Næste ordre i hac m.
Operationsbetegnelse: 10	Tid: 1 AT.
Kode: n,I,10	

Stop,hop: Stop:hop	Virkning: Maskinen stopper. Ved tryk på startknap begynder maskinen med ordren i hac m.
Operationsbetegnelse: 30	Tid: 1 AT.
Kode: n,I,30	

Tøm AR og hop: 0;hop	Virkning: AR nulstilles. Næste ordre i hac m.
	Indholdet ændres i: AR.
Operationsbetegnelse: 50	Tid: 1 AT.
Kode: n,I,50	

Tøm AR.Stop,hop: 0;stop;hop	Virkning: AR nulstilles. Maskinen stopper derefter. Ved tryk på startknap begynder maskinen med ordren i hac m.
Operationsbetegnelse: 70	
Kode: n,I,70	Indholdet ændres i: AR.
	Tid: 1 AT.

Hop på nul og plus: + og 0 hop	Virkning: Er C(AR) $\geq$ 0 hentes næste ordre i hac m. Er C(AR) < 0, fortsættes normalt.
Operationsbetegnelse: 11	Tid: 1 AT.
Kode: n,I,11	

Stop,hop på nul og plus: Stop; + og 0 hop	Virkning: Maskinen stopper. Ved tryk på startknap som ved 11.
Operationsbetegnelse: 31	Tid: 1 AT
Kode: n,I,31	

Hop på minus: - hop	Virkning: Er C(AR) < 0 hentes næste ordre i hac m. Er C(AR) $\geq$ 0 fortsættes normalt.
Operationsbetegnelse: 51	
Kode: n,I,51	Tid: 1 AT.

Stop,hop på minus: Stop;-hop	Virkning: Maskinen stopper. Ved tryk på startknap som ved 51.
Operationsbetegnelse: 71	
Kode: n,I,71	Tid: 1 AT.

NB. m betegner den effektive adresse; m = n + C(IR).

Hop på spild: Spildhop	Virkning:	Er der spild, dvs $C(AR_{00}) \neq C(AR_0)$, hentes næste ordre i hac m. Ellers fortsættes normalt.
Operationsbetegnelse: 12		
Kode: n,I,12	Tid:	1 AT.

Stop; hop på spild: Stop, spildhop	Virkning:	Maskinen stopper. Ved tryk på startknap som ved 12.
Operationsbetegnelse: 32		
Kode: n,I,32	Tid:	1 AT.

Skift eller hop på ikke spild: Sk.; Ikke spildhop	Virkning:	1) Er der spild, dvs $C(AR_{00}) \neq C(AR_0)$, skiftes 1 position til højre i AR. $C(AR_{00})$ skiftes ind i AR_0, hvorefter AR_{00} nulstilles. Derefter fortsættes normalt med den korrekte halve sum i AR_{0-39}.
Operationsbetegnelse: 52		2) Er der ikke spild, $C(AR_{00}) = C(AR_0)$ hentes næste ordre i hac m.
Kode: n,I,52		Indholdet ændres eventuelt i: AR.
	Tid:	1 AT.

Stop,skift eller hop på ikke spild: Stop;SK;Ikke spildhop	Virkning:	Maskinen stopper. Ved tryk på startknap som i 52.
Operationsbetegnelse: 72		Indholdet ændres eventuelt i: AR.
Kode: n,I,72	Tid:	1 AT.

Hop på indeks B: IRB hop	Virkning:	Er C(IRB) ≠ 0 hentes næste ordre i hac m. Ellers fortsættes normalt.
Operationsbetegnelse: 33		
Kode: n,I,33	Tid:	1 AT.

Hop på indeks C: IRC hop	Virkning:	Er C(IRC) ≠ 0 hentes næste ordre i hac m. Ellers fortsættes normalt.
Operationsbetegnelse: 53		
Kode: n,I,53	Tid:	1 AT.

Hop på indeks D: IRD hop	Virkning:	Er C(IRD) ≠ 0 hentes næste ordre i hac m. Ellers fortsættes normalt.
Operationsbetegnelse: 73		
Kode: n,I,73	Tid:	1 AT.

NB. m betegner den effektive adresse; m = n + C(IR).

Læs indeks B til celle: C(IRB); L	Virkning: C(IRB) går til adressepositionerne i hac m. (1-11 for m lige, 21-31 for m ulige). Halvordet i hac m får derved ny pseudo-adresse. De øvrige positioner i hac m er uændrede.
Operationsbetegnelse: 34	Indholdet ændres i: hac m.
Kode: n,I,34	Tid: 1 AT.

Læs indeks C til celle: C(IRC); L	Virkning: Som ved 34, idet IRC træder i stedet for IRB.
Operationsbetegnelse: 54	Indholdet ændres i: hac m.
Kode: n,I,54	Tid: 1 AT.

Læs indeks D til celle: C(IRD); L	Virkning: Som ved 34, idet IRD træder i stedet for IRB.
Operationsbetegnelse: 74	Indholdet ændres i: hac m.
Kode: n,I,74	Tid: 1 AT.

Læs til indeksregister B: m → IRB	Virkning: m fremkommer i IRB.
	Indholdet ændres i: IRB.
Operationsbetegnelse: 35	Tid: 1 AT.
Kode: n,I,35	Bemærkning: m = n + indholdet af det indeksregister hvormed ordren er indeksmærket.

Læs til indeksregister C: m → IRC	Virkning: m fremkommer i IRC.
	Indholdet ændres i: IRC.
Operationsbetegnelse: 55	Tid: 1 AT.
Kode: n,I,55	Bemærkning: m = n + indholdet af det indeksregister hvormed ordren er indeksmærket.

Læs til indeksregister D: m → IRD	Virkning: m fremkommer i IRD.
	Indholdet ændres i: IRD.
Operationsbetegnelse: 75	Tid: 1 AT.
Kode: n,I,75	Bemærkning: m = n + indholdet af det indeksregister hvormed ordren er indeksmærket.

NB. Uanset om m opfattes som effektiv adresse eller ej,
er m = n + C(IR).

Indekshop: 16-hop	Virkning: C(KR) overføres til IRD. Næste ordre hentes i hac m.
Operationsbetegnelse: 16	Indholdet ændres i: IRD.
Kode: n,I,16	Tid: 1 AT.
	Bemærkning: Denne vigtige operation anvendes ved hop til specielle sekvenser (undersekvenser), hvorfra man senere ønsker at hoppe tilbage til hovedsekvensen. C(IRD)+1 bliver jo adressen på den ordre, man skal fortsætte med.
Stop; Indekshop: Stop; 16-hop	Virkning: IRD nulstilles, hvorefter maskinen stopper. Ved tryk på startknap føres C(KR) til IRD, hvorefter næste ordre hentes i hac m.
Operationsbetegnelse: 36	Indholdet ændres i: IRD.
Kode: n,I,36	Tid: 1 AT.
	Se bemærkningen ved 16.
Betinget indekshop: 56-hop	Virkning: IRD nulstilles. Hvis afbryderen på kontrolbordet er i aktiv-stilling udføres operation 16. I passiv stilling ingen yderligere funktion.*)
Operationsbetegnelse: 56	
Kode: n,I,56	Se iøvrigt 16.
Stop, betinget indekshop: Stop; 56-hop	Virkning: IRD nulstilles og maskinen stopper. Ved tryk på startknap udføres operation 16, hvis afbryderen på kontrolbordet er i aktiv-stilling. I passivstilling ingen yderligere funktion.*)
Operationsbetegnelse: 76	
Kode: n,I,76	
	Se iøvrigt 16.

NB. m betegner den effektive adresse; m = n + C(IR).

*) NB. C(IRD) = 0.

KAPITEL 5.

Øvelser og eksempler I.

I dette kapitel vil vi gennem en række eksempler vise, hvordan man kan kode forskellige meget simple problemer. De operationer, der anvendes på de følgende sider, er alle nævnt i kapitel 4.

For simpelheds skyld er det i eksemplerne forudsat, at spild ikke kan finde sted, altså at ingen af regningerne fører os uden for intervallet $-1 \leq x < 1$.

Eksempel 5.1 Stop af maskinen sker ved anvendelse af en vilkårlig af følgende otte operationer:
30, 70, 31, 71, 32, 72, 36, og 76.
Maskinen stopper uafhængig af hvilken pseudoadresse (n) og hvilket indeksregister (I), der er benyttet i ordren. Vi kan altså få maskinen til at stoppe ved at anvende følgende ordrer:

n I 30	n I 32
n I 70	n I 72
n I 31	n I 36
n I 71	n I 76

Har man stoppet maskinen ved hjælp af en af operationerne 30, 70 eller 36, vil maskinen - ved tryk på startknap - begynde med ordren i halvcelle m = n + C(I). Dette gælder også for de betingede stop- og hopoperationer 31, 71, 32, 72 og 76 såfremt hopbetingelsen er opfyldt. Er dette ikke tilfældet fortsættes - ved fornyet start - med ordren i næste halvcelle.

Anvendelse af 00, 20, 40 og 60.

Eksempel 5.2. Idet x,y,z og v er lagrede i henholdsvis vhac 100, hhac 101, hec 102 og hec 104, skal x+y+z+v dannes i AR.

hæa. nr.	ordre	
0	100 A 60	C(100v) = x til ARv. (hele AR tom)
1	101 A 20	C(101) = y adderes til C(ARv) d.v.s. x+y i ARv.
2	102 A 00	C(102) = z adderes til C(AR) d.v.s. x+y+z i AR.
3	104 A 00	C(104) = v adderes til C(AR) d.v.s. x+y+z+v i AR.
4	5 A 30	Stop.- Ved ny start går maskinen videre til ordren i hac 5.

1) Tegn et skitse af en del af lageret og "indsæt" x,y,z og v i de rigtige hel- eller halvceller. "Indsæt" ligeledes ordrerne i lageret.

2) Hvordan kunne vi kode, hvis operation 40 skal anvendes?

Anvendelse af 01, 21, 41, og 61.

Øvelse 5.3 x,y,z og v er lagrede som i eks. 5.2. Dan -x-y-z-v i AR.

Anvendelse af 02, 22, 42, 62, 03, 23, 43 og 63.

Øvelse 5.4 x, y, z og v er lagrede som i eks. 5.2.
Dan $|x| - |y| - |z| - |v|$ i AR.

Problemet bør kodes flere gange (indtil man har prøvet alle otte operationer).

Anvendelse af 08, 28, 48 og 68.

Eksempel 5.5 x, y, z og v er lagrede som i eks. 5.2.
1) |z| + v skal lagres i hec 106.

```
0    102 A  42    |z| i AR
1    104 A  00    |z| + v i AR
2    106 A  08    |z| + v i 106
3      4 A  30    Stop.
```

2) x - |y| skal lagres i vhac 106 og i hhac 107.
hhac 109 skal nulstilles.

```
0    100 A  60    x i ARv  (ARh tom)
1    101 A  23    x - |y| i ARv
2    106 A  28    x - |y| i vhac 106
3    107 A  28    x - |y| i hhac 107
4    109 A  68    0 i hhac 109
5      6 A  30    Stop
```

Øvelse 5.6 x,y,z og v er lagrede som i eks. 5.2
Anbring -y + |x| i hhac 107 og
z - v i hec 108.
Nulstil derefter hec 110.

Anvendelse af 04, 24, 44, 64, 0A, 2A, 4A og 6A

Eksempel 5.7 Idet x, y og z er lagrede i henholdsvis
hec 230, hec 232 og hec 234 skal
(x+y)z udregnes og lagres i hec 236.

```
0    230 A  40    x i AR
1    232 A  04    x+y i AR og i MR
2    234 A  0A    (x+y)z afrundet i AR ; x+y stadig i MR
3    236 A  08    (x+y)z i hec 236
4      5 A  30    Stop
```

1) Kunne 44 erstatte 40 i ordre 0?
2) Kunne 44 erstatte 04 i ordre 1?

Øvelse 5.8 x i vhac 230, y i hhac 231,
z i vhac 232 og v i hhac 233.
1) (x+y)z skal lagres i hec 234.
2) (y+v)x og (y+v)y skal lagres
i henholdsvis hec 236 og hec 238.
3) (x+v) · (y+z) skal lagres i hec 240.
Uafrundet multiplikation benyttes.

Øvelse 5.9 Hvad ville benyttelse af afrundet
multiplikation bevirke i øvelse 5.8?

Anvendelse af 05, 25, 45 og 65

Øvelse 5.10 Samme lagring som i øvelse 5.8.
(x+y) · (-z) skal lagres i hec 234.

Øvelse 5.11 a i hec 1000, b i hec 1002, c i hec 1004.
(-|a|-b) · c skal lagres i hec 998.
Hvilken multiplikation bør her benyttes?

Anvendelse af 06 og 26

Eksempel 5.12 x i hec 100, y i hec 102.
x og y skal byttes om.

```
0    100 A  41    -x i AR
1    102 A  06    y-x i AR og i 102
2    100 A  06    x+(y-x) = y i AR og i 100
3    102 A  01    y-(y-x) = x i AR
4    102 A  08    x i 102
5      6 A  30    Stop
```

Behøver vi at ændre denne kode, hvis spild kan indtræffe

Øvelse 5.13 x i vhac 100; y i hhac 101.
x og y skal byttes om.

Øvelse 5.14 Samme opgave som i eks. 5.12
06 må ikke benyttes. Der må kun
anvendes ialt seks ordrer incl. stopordren.

Eksempel 5.15 x i vhac 100; y i hhac 101.
$\frac{x}{y}$ skal lagres i hec 102.
Det antages, at $|y| \geq |x|$.

0	100 A 60	x i ARv
1	101 A 2B	$\frac{x}{y}$ i MR; divisionsresten • 2^{39} i AR.
2	0 A 07	$\frac{x}{y}$ i AR (og i MR)
3	102 A 08	$\frac{x}{y}$ i 102
4	5 A 30	Stop.

NB. Ved operation 07 er pseudoadressen og indeksmærkningen uden indflydelse på resultatet.

Øvelse 5.16 x i hec 100, y i hec 102 og z i hec 104.
$\frac{-|z|}{|x|+|y|+|z|}$ skal lagres i hec 106.

Eksempel 5.17 a i hec 100, b i vhac 102, c i hhac 103.
Idet det antages, at $c \neq 0$ samt at $|a| \leq |c|$
skal $-\frac{ab}{c}$ lagres i hec 104.
metode: $(-\frac{a}{c}) \cdot b$

0	100 A 41	-a i AR
1	103 A 2B	$-\frac{a}{c}$ i MR
2	102 A 2A	$(-\frac{a}{b})$b afrundet i AR
3	104 A 08	$(-\frac{a}{c})$b i 104
4	5 A 30	Stop.

Øvelse 5.18 Samme opgave som i eks. 5.17,
idet koden nu skal svare til
metoden: $(-a \cdot b) : c$.

Anvend uafrundet multiplikation og
lang division.

Anvendelse af 10, 50, 11 og 51.

Eksempel 5.19 x i vhac 50, og y i hhac 51,
u i vhac 52, og v i hhac 53.

Først skal x-y dannes. Hvis x-y $\geq$ 0, skal (x-y) - u dannes. Hvis derimod x-y < 0, skal (x-y) + v dannes. Resultatet skal i vhac 54.

```
      0     50 A 60    x i ARv
      1     51 A 21    x-y i ARv
5 <-- 2      5 A 11    hop hvis x-y >= 0 ; 1)
      3     53 A 20    x-y+v i ARv
6 <-- 4      6 A 10    hop til udlæsning; 2), 3)
2 --> 5     52 A 21    x-y-u i ARv
4 --> 6     54 A 28    Resultatet i vhac 54
      7      8 A 30    Stop.
```

1) Pseudoadressen indsættes først senere.
2) Hvad sker der, hvis man udlæser direkte til vhac 54 på dette sted?
3) Kan 10 erstattes med 50?

Øvelse 5.20 Samme opgave som i eks. 5.19, idet dog

1) x, y, u og v er lagrede i henholdsvis hec 80, 82, 84 og 86 (Resultat i 88).
2) Minushoppet 51 skal benyttes.

Anvendelse af 12, 52, 0C, 4C, 0D, 4D, 0F og 4F.

Eksempel 5.21 x står i den lange akkumulator AR_{0-39}, MR_{1-39}

Dan 2 x, hvis det er muligt. Hvis der er spild, skal maskinen stoppe, ellers skal 4x dannes.

Hvis der ikke er spild nu, skal 4x lagres i hec 20 og i hec 22. (4x fylder jo to helceller). Er der spild, skal maskinen stoppe.

0	1 A 4C	1 venstre skift i AR_{0-39}, MR_{1-39} d.v.s. 2x i den lange akkumulator.
1	8 A 12	Er der spild hoppes til stop. [1]
2	1 A 4C	1 venstre skift i AR_{0-39}, MR_{1-39} d.v.s. 4x i den lange akkumulator.
3	8 A 12	Er der spild hoppes til stop. [1]
4	20 A 08	Fortegn og 39 første binaler i 4x i hec 20.
5	22 A 48	hec 22 og AR nulstilles for at få korrekt fortegnsciffer på den sidste del af 4x.
6	39 A 4C	39 venstreskift i den lange akkumulator
7	22 A 08	"resten" af 4x lagres.
8	9 A 30	Stop.

1) Pseudoadressen indsættes til slut.

Øvelse 5.22 x står i AR. x:16 skal dannes i AR_{0-39}, MR_{1-39} og derefter lagres i hec 48 og i vhac 50.
(Division med 16 svarer til 4 højre skift).
Kan 4F altid bruges? Kan 4F aldrig bruges?

Øvelse 5.23 x står i AR. 2x skal lagres i hec 78, hvis det er muligt. Indtræffer der spild, skal x lagres i hec 78. Anvend operation 52.

Øvelse 5.24 Tallene a, b, og c er lagrede i hec 1220, således at a indtager positionerne 1-13, b positionerne 14-26 og c positionerne 27-39. a, b, og c er alle DASKtal; fortegnscifrene står i henholdsvis pos. 1, pos. 14 og pos. 27. a og b skal lagres normalt i henholdsvis vhac 1222 og hhac 1223. c skal stå i MR, således at C(MR) = c.

I de følgende opgaver står
a i vhac 200, b i hhac 201,
c i vhac 202, d i hhac 203,
x i hec 204 og y i hec 206.
Når intet andet forlanges, skal resultatet lagres i hec 208.

Øvelse 5.25 $a + 2b + 3c + 4d$ i hhac 209.

5.26 $a \cdot |b| + cd$

5.27 $(a+b) : (c+d)$ i hhac 209.

5.28 $a \cdot b \cdot |c|$

5.29 $(a \cdot b + c \cdot d) : (x \cdot y)$

5.30 $(a-bc) : 4(x+y)$

5.31 $a+bx+cx^2$

5.32 $(ax+b) : (cx^2-dx)$

5.33 Det største af tallene a, b og c skal lagres i vhac 208.

Øvelse 5.34 Koderne til opgaverne i øvelserne 5.25 til 5.32 skal ændres således, at maskinen stopper, hvis der indtræffer spild eller divisionerne ikke kan udføres.

Anvendelse af løkker.

I mange programmer skal en bestemt programdel gentages et stort antal gange. Koderen kan opnå, at maskinen selv sørger for, - ved løkkeregning - at gennemløbe den pågældende programdel det ønskede antal gange. Hertil vil man normalt anvende indeksregistrene, men løkkeregningen kan også kodes ved anvendelse af 46 og 66.

Eksempel 5.35 Tallene $a_o a_1$,..., a_{19} er lagrede i hec 100, 102, ..., 138, og tallene b_o, b_1,...,b_{19} er lagrede i hec 200, 202, ...,238.

$S = a_o b_o + a_1 b_1 + ... + a_{19} b_{19}$ skal lagres i hec 98.

I vhac 0 lagrer vi tællekonstanten 139,A,00

	0	139 A 00	tællekonstant		
Start →	1	98 A 48	lagringscellen og AR tømmes		
			ved 1. gennemløb	ved (i+1)'te gennemløb	
7 →	2	[100] A 44	a_o til AR og MR	a_i til AR og MR	
	3	[200] A 0A	$a_o \cdot b_o$ i AR	$a_i \cdot b_i$ i AR	
	4	98 A 06	$a_o \cdot b_o + C(98) = a_o b_o$ til 98	$a_i \cdot b_i + C(98) = a_o b_o + ... + a_i b_i$ til 98	
	5	2 A 46[1)]	$C(2) + 2^{-10} + 2^{-30}$ til AR og hec 2	$C(2) + 2^{-10} + 2^{-30}$ til AR og hec 2.	
	6	0 A 01[2)]	C(AR) - C(o) til AR.	C(AR)-C(o) til AR	
2 ←	7	2 A 51	hop på -	hop på -	
	8	9 A 30	stop	stop	

1) Ved 1. gennemløb kommer der 102A44202A0A i AR (omsat på rette måde i 2-talsystemet).

2) Ved 1. gennemløb kommer der 102A44202A0A - 139A0098A48 i AR; dette er negativt. Først ved 20. gennemløb kommer der et positivt tal i AR. (140A44240A0A - 139A0098A48 i AR).

yderligere bemærkninger:

3) De indrammede pseudoadresser ændres.

4) Tællekonstanten er lagret i vhac 0, for på en naturlig måde at få de to adresser der skal ændres (modificeres) til at ligge i samme hec.

Øvelse 5.36 Samme opgave som i eks. 5.35, men starten skal foregå med ordren i vhac 0 (tællekonstanten lagres til sidst i programmet; 66 bør benyttes).

Anvendelse af 0E og 4E

Eksempel 5.37 x i hec 100, y i hec 102

$z = \frac{x}{y}$ skal dannes på formen $z = z_N \cdot 2^{-q}$, hvor z_N er normaliseret d.v.s. $-1 \leq z_N < -\frac{1}{2}$ eller $z_N = 0$ eller $+\frac{1}{2} \leq z_N < 1$. z_N i hec 104. Eksponenten med modsat fortegn, d.v.s. + q, i hhac 107 med enhed i pos. 11.
Da x = 0 og y = 0 giver afvigelser, der kan bortlede opmærksomheden fra det principielle i sagen, forudsætter vi $x \neq 0$ og $y \neq 0$. Derimod forudsættes ikke at $|y| \geq |x|$.

$$z = \frac{x}{y} = \frac{x}{y_N \cdot 2^{-q_y}} = \frac{x}{y_N} \cdot 2^{q_y} = \frac{\frac{x}{2}}{y_N} \cdot 2^{q_y + 1} = z' \cdot 2^{q_y + 1}$$

Da y_N er normaliseret er $|z'| = \left|\frac{\frac{x}{2}}{y_N}\right| \leq 1$, og

divisionen $\frac{x}{2} : y_N$ kan altid udføres. Sluttelig normaliseres z'.

$$z = z' \cdot 2^{q_y+1} = z_N \cdot 2^{-q_z} \cdot 2^{q_y+1} = z_N \cdot 2^{-q}$$

hvor $q = q_z - q_y - 1$.

0	106 A 48	0 til 106; 0 til 107
1	102 A 40	y i AR
2	106 A 0E	y_N i AR. Antallet af venstreskift = $+q_y$ i 106 med enhed i pos. 11.
3	108 A 08	y_N lagres i hec 108
4	100 A 40	x i AR
5	1 A 4D	$\frac{x}{2}$ i AR_{0-39}, MR_{1-39}
6	108 A 4B	$\frac{x}{2} : y_N$ i MR
7	0 A 07	$\frac{x}{2} : y_N$ i AR
8	107 A 0E	z_N i AR. Antallet af venstre skift = + q_z i 107
9	104 A 08	z_N lagres i 104
10	106 A 61	$-q_y$ i AR
11	2039 A 21	$-q_y - 1$ i AR (med enhed i pos. 11) *)
12	107 A 26	$q_z - q_y - 1 = q$ i 107
13	15 A 30	stop

*) Det vil normalt gælde, at der i celle 2039 står konstanten 2^{-11} (en såkaldt permanent konstant)

Øvelse 5.38 Samme opgave som i eks. 5.37, men koden skal ændres således, at følgende krav opfyldes:

1) maskinen stopper hvis y = 0.

2) C(107) = 0 og stop hvis x = 0.

Anvendelse af 09 29 49 og 69.

I mange maskiner har disse operationer stor betydning, men i maskiner som DASK, der anvender indeksregistre, varetager disse registre delvis de funktioner operationerne 09, 29, 49 og 69 kan udføre. Vi gør opmærksom på en forskel: de nævnte operationer kan ændre indholdet i positionerne 0 - 11, hvorimod indeksregistrene kun kan ændre indholdet i positionerne 1 - 11.

Hvis IRB, IRC og IRD imidlertid er benyttede i en kode, er det af betydning at kunne anvende de nævnte operationer; specielt Wheelerhoppet, der forklares og anvendes i det følgende eksempel.

Eksempel 5.39 Vi betragter det meget hyppigt forekommende tilfælde, hvor der fra forskellige steder i hovedprogrammet, skal springes til en og samme undersekvens, der står et bestemt sted i lageret. I denne foretages visse regninger, hvorefter der skal fortsættes, hvor man slap i hovedprogrammet.

Lad hovedsekvensen være lagret med 1ste ordre i vhac 50, og lad os antage, at man ved ordren i hhac 55 ønsker at hoppe til en undersekvens, der tænkes lagret i vhac 1028 til hhac 1033. Derefter skal der hoppes tilbage til ordren i vhac 56 i hovedsekvensen.

50	- - - - - - - - - - -	51	
52	- - - - - - - - - - -	53	
54	54 A 60 1028 A 10	55	
56	- - - - - - - - - - -	57	
1028	2039 A 20 2039 A 20	1029*)	
1030	1033 A 29 - - - - - -	1031	
1032	- - - - - - - (0) A 10	1033	

Efter Wheeler koder man da på følgende måde:

I hovedsekvensen:

```
53 -------
54   54 A 60   C(54) = 54 A 60 i ARv, 0 i ARh
55 1028 A 10   hop til 1028
```

I undersekvensen

```
1028   2039 A 20   C(2039)+ C(AR) =
                   55 A 60 i AR
1029   2039 A 20   C(2039)+ C(AR) =
                   56 A 60 i AR.
1030   1033 A 29   til adressepos. i 1033
1031   ---  ----
1032   ---  ----
1033     (0)A 10   Her skriver koderen
                   f.eks. 0. Når ordren
                   gennemløbes står der
                   56 A 10 i hhac 1033, og
                   næste ordre hentes i
                   vhac 56.
```

NB. I hhac 2039 står der altid 1 i pos. 11.

*)Gentagelsen af ordren 2039 A 20 kan undgås, hvis koderen selv lagrer et 2 tal i adressepos., i stedet for at benytte det fast lagrede 1 tal i adressepos. i 2039.

Vi bemærker, at Wheelerhoppet ved hvert nyt indhop skaber netop den rigtige udhopsadresse. Havde man kun skullet bruge undersekvensen fra et sted i hovedsekvensen klares tilbagehoppet ved en fast ordre i undersekvensen (her ved ordren 56 A 10 i hac 1033).
Iøvrigt anvendes i DASK den fremgangsmåde, der er vist på side 5.20.

Anvendelse af 47 med benyttelse af indeksoperationer.

Eksempel 5.40. Ved mange opgaver, bl.a. i forbindelse med anvendelsen af hulkort, skal man undersøge om en bestemt binær størrelse har 1-taller i bestemte positioner. Til denne undersøgelse anvendes den logiske multiplikation.

I helcellerne 1000 til 1398 har vi lagret tallene $a_0, a_1, \ldots, a_{199}$.

Vi vil undersøge hvor mange af disse 200 tal, der har 1 i positionerne 7 og 13.

Konstanten K = 0,000000100000100...0 tænkes lagret i helcelle 998. I helcelle 994 lagres tællekonstanten $1400 \cdot 2^{-11}$. Det søgte antal i hec 996 med enhed i pos. 11.

0	996 A 48	0 til 996
1	998 A 44	K → MR(og AR)
9 → 2	1000 A 40	a_i → AR
3	0 A 47	$a_i \cap K$ → AR
4	998 A 01	$a_i \cap K - K$ → AR
7 ← 5	7 A 51	hop på minus (d.v.s. hvis a_i ikke har 1 i pos. 7 og 1 i pos. 13)
6	996 A 66	tæl
5 → 7	2 A 66	øg adressedel i 2 med 2
8	994 A 21	$C(2) - 1400 \cdot 2^{-11}$ → AR
2 ← 9	2 A 51	hop på ÷
10	996 A 40	
11	1 A 0D	
12	996 A 08	
13	14 A 30	

Må koden ændres, ifald K er negativ.

I de følgende eksempler vil vi vise, hvordan indeksoperationerne anvendes. Da den rette brug af disse operationer dels vil gøre kodearbejdet lettere og dels vil bevirke en forøgelse af maskinens regnehastighed, er det vigtigt, at koderen bliver fortrolig med indholdet af de følgende sider. Ydermere er de følgende eksempler af en mere generel karakter end de foregående, idet et eller flere af de problemer, der behandles i den resterende del af indeværende kapitel, skal løses i forbindelse med alle i praksis forekommende koder.

Vi indleder med at minde læseren om maskinens behandling af en indeksmærket ordre. I en sådan vil den effektive adresse fremkomme som sum af pseudoadressen og indholdet af det i ordren anførte indeksregister.

$$m = n+C(IR)$$

Hertil er dog at bemærke, at denne addition sker modulo 2048, hvad der betyder, at m, såfremt n+C(IR) overstiger 2048, bliver lig

$$n+C(IR) - 2048.$$

Eks. C(IRB) = 1656. Ordren 1714 B 40 bevirker da, at

$$C\left[1714+C(IRB)\right] = C\left[1714+1656-2048\right] = C(1332) \text{ kommer i AR.}$$

Anvendelser af 33, 53, 73, 35, 55 og 75

Eksempel 5.41 Læs 142 til IRB, 2004 til IRC og 1357 til IRD

```
0    142 A 35    142+C(IRA)=142+0 til IRB
1   2004 A 55   2004+C(IRA)=2004+0 til IRC
2   1357 A 75   1357+C(IRA)=1357+0 til IRD
3      4 A 30   Stop
```

Vi bemærker, at indlæsningen til indeksregistrene er foregået, uden at AR er berørt.

Øvelse 5.42 Koden i eks. 5.41 tænkes forøget med følgende ordrer:

```
4   1908 B 35
5      0 B 40
6     44 C 55
7      1 C 64
8    694 D 10
```

Nu begynder vi igen med ordre i hac 0; efter stoppet trykkes på startknap.

1) Hvad sker der i IRB efter udførelsen af ordre 4?
2) - - - - AR - - - - 5?
3) - - - - IRC - - - - 6?
4) - - - - MR - - - - 7?
5) Hvad sker der efter at ordre 8 er udført?

I alle koder, der anvender løkker, skal koderen sørge for, at maskinen gennemløber de i løkken forekommende ordrer et bestemt antal gange. Denne optælling kan klares som vist i det følgende:

Eksempel 5.43 Det antages, at løkken skal gennemløbes 75 gange. Inden vi kommer til løkken anbringer vi 75 i et indeksregister. Dette gøres ofte i den umiddelbart før løkken forekommende ordre.

```
Den                0  ...  .  ..
almin-             1  ...  .  ..
delige             .  ...  .  ..
kode I           m-1  75   A  35     75 til IRB
                 m    2047 B  35     C(IRB) - 1 til IRB, idet 2047, på grund
                 .    ...  .  ..              af additionen modulo 2048, virker
løkken           .    ...  .  ..              som -1.
                 .    m    A  33     Her hoppes til den første ordre
                 ---------------     i løkken, hvis C(IRB) ≠ 0
Den              .    ...  .  ..     Efter 1. gennemløb er C(IRB) = 74;
almin-           .    ...  .  ..              d.v.s. hop til ordre i
delige           .    ...  .  ..              halvcelle m.
kode II          .    ...  .  ..     Efter 2. gennemløb er C(IRB) = 73;
                                              d.v.s. hop til ordre i
                                              halvcelle m,
```

o.s.v. indtil løkken netop er gennemløbet 75 gange; da er C(IRB) = 0, og maskinen fortsætter med den almindelige kode II.

Af hensyn til de følgende anvendelser gør vi læseren opmærksom på, at koden fungerer på helt samme måde, hvis vi f.eks. sætter 2 · 75 (eller n · 75) til IRB og derefter trækker 2 (henholdsvis n) fra i IRB i den 1. ordre i løkken.

Den næste vigtige anvendelse vi vil gøre af indeksregistrene hænger også sammen med løkkeregningen. Som oftest skal adressen ændres (modificeres) i en eller flere af de ordrer, der indgår i løkken. For at dette kan ske på en rimelig måde, kræves det af koderen, at denne placerer sine konstanter i "pæn" orden i lageret, normalt i konsekutive halv- eller helceller. Metoden belyses ved det følgende eksempel.

Eksempel 5.44 a_0; a_1; ...; a_{42} tænkes lagrede i halvcellerne 100, 101,....,142. Summen af tallene skal dannes i hac 144.
Koden indeholder en løkke, der skal gennemløbes 43 gange. Lad IRB sørge for dette. IRC anvendes til adressemodifikationen.

	0	144 A 68	0 til vhac 144 og til AR
	1	0 A 55	0 til IRC, i = 0
	2	43 A 35	43 til IRB
6 →	3	2047 B 35	C(IRB) -1 til IRB
	4	100 C 20	a_i + C(AR) → AR
	5	1 C 55	C(IRC) + 1 → IRC, i øges med 1
3 ←	6	3 A 33	hop, hvis C(IRB) ≠ 0
	7	144 A 28	Resultatet i 144
	8	9 A 30	Stop.

Bemærkninger. Nulstillingen af 144 i ordre 0 er overflødig, men AR kal nulstilles på grund af operation 20 i ordre 4.
Ved 1. gennemløb er den effektive adresse i ordre 4: 100 + C(IRC) = 100 + 0.
D.v.s. ordren bevirker, at C(100)= a_0 adderes til C(AR) = 0.
Ved 2. gennemløb er den effektive adresse 100 + C(IRC) = 100 + 1 = 101; altså vil a_1 adderes til C(AR) o.s.v.

Normalt anvender man kun et indeksregister til samtidig at løse begge de nævnte opgaver. Som eksempel på dette viser vi, hvordan det i eks. 5.35 side 5.9 stillede problem kan løses ved anvendelse af IRD.

Eksempel 5.45 Tallene $a_1, a_2, \ldots, a_{19}$ er lagrede i hec 100, 102,....., 138, og tallene $b_0, b_1, \ldots, b_{19}$ er lagrede i hec 200, 202,....., 238.
$S = a_0b_0 + a_1b_1 + \ldots + a_{19} \cdot b_{19}$ skal lagres i hec 98.
Det er naturligt at anvende løkkeregning med 20 gennemløb.
Da de konstanter, vi anvender, er lagrede i konsekutive helceller, skal den effektive adresse ændres med 2, hver gang løkken gennemløbes. Koden kan da f.eks. se således ud:

0	98 A 48	0 til hec 98
1	40 A 75	40 = 2·(antal gennemløb) til IRD
6 →2	2046 D 75	C(IRD) - 2
		Ved 1. gennmløb
3	100 D 44	$C(100 + 2 \cdot 19) = C(138) = a_{19}$ til MR (og AR)
4	200 D 0A	$C(238) \cdot C(MR) = a_{19} \cdot b_{19}$ til AR
5	98 A 06	$C(98) + a_{19} \cdot b_{19} = a_{19} \cdot b_{19}$ til hec 98 (og til AR)
2←6	2 A 73	hop til 2 da C(IRD) ≠ 0
7	8 A 30	Stop

Vi bemærker, at denne anvendelse af indeksregistret - hvor vi starter med at sætte 2·(antallet af gennemløb) til IRD - medfører, at begyndelsesadresserne 100 og 200 skal anbringes i ordrerne 3 og 4. Regningerne begynder da, som vist, med de tal, der er lagrede i slutadresserne 138 og 238. Ønsker man, at regningerne skal begynde med $a_0 \cdot b_0$ kan dette ske ved følgende ændringer.

1) I IRD anbringes - 2 · 20 = -40, der kodes som 2008
2) I ordre 2 adderes 2 til C(IRD)
3) I ordrerne 3 og 4 skrives slutadresserne 138 og 238.

Anvendelse af 34,54,74,16,36,56 og 76

Eksempel 5.46 Indholdet af hhac 139 ønskes ændret fra 1246 D 42 til 1887 D 42.

```
0    1887 A 55        1887 til IRC
1     139 A 54        C(IRC) = 1887 til adressepos. 1 - 11 i 139.
2       3 A 30        Stop.
```

Det bemærkes, at operationerne 34,54 og 74 kun kan bevirke ændringer i indholdet af adressepos. 1-11.

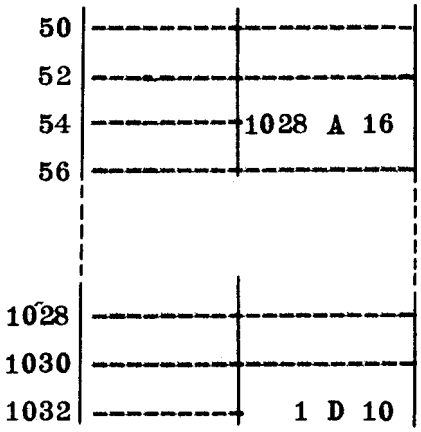
34,54 og 74 anvendes iøvrigt, når antallet af indeksregistre ikke slår til.
Ved hjælp af en af de tre operationer lagrer man indholdet af det pågældende indeksregister, hvorefter dette frit kan anvendes. Efter anvendelsen kan man så, inden man går videre i regningerne, indsætte indeksregistrets oprindelige indhold i dette.

Medens IRB, IRC og IRD fungerer på helt samme måde ved alle de nævnte operationer, indtager IRD en særstilling i de følgende eksempler, idet den vigtige operation 16 med variant-erne 36, 56 og 76 udelukkende benytter dette indeksregister.

16-hoppet benyttes i stedet for Wheelerhoppet, der, som omtalt i eks.5.39 side 5.11 og 5.12 kunne anvendes ved hop fra hovedsekvens til undersekvens og tilbage igen. Disse hop, der forekommer i næsten alle større koder, klares af operation 16, således som de følgende eksempler viser.

Eksempel 5.47 16-hop til undersekvens, der ikke benytter IRD

Lad selve koden (hovedsekvensen) være lagret med første ordre i vhac 50 og lad os antage, at man ved ordren i hhac 55 ønsker at hoppe til en undersekvens, der tænkes lagret i vhac 1028 til hhac 1033. Derefter skal der hoppes tilbage til ordren i vhac 56 i hovedsekvensen.



I hovedsekvensen koder man da:

54 ----------

55 1028 A 16

56 ----------

Herved vil C(KR) = 55 → IRD, og maskinen vil hoppe til ordren i hac 1028.

Undersekvensen skal da slutte med ordren 1 D 10 (her lagret i hac 1033), der bevirker ubetinget hop til hac (1 + C(IRD))=56, hvilket jo netop var formålet.

NB. De videre regninger i hovedsekvensen vil nu foregå med C(IRD) = 55, hvad koderen naturligvis må tage i betragtning ved senere anvendelser af IRD.

Eksempel 5.48 16-hop til undersekvens, der selv benytter IRD.

Problemstilling og hovedsekvens nøjagtig som i det lige viste eksempel. Undersekvensen kan da se således ud- idet vi som før antager, at selve regningerne i denne kan udføres ved hjælp af fem ordrer:

1028	1034 A 74	C(IRD)=55 læses ud til 1034 i pos. 1-11
1029	----------	
1030	----------	
1031	----------	Regningerne i
1032	----------	undersekvensen
1033	----------	
1034	(0) A 75	I koden skrives f.eks. 0 (af ordren i hac 1028 anbringes 55 i adressepos.)
1035	1 D 10	Tilbagehop til ordren i 56 (af ordren i 1034 sættes C(IRD) = 55)

Eksempel 5.49 Rettelse af fejl i koden ved hjælp af parentes.

Lad os antage, at hovedsekvensen skal placeres i hac 20-hac 97. Efter at koden er nedskrevet, opdager man, at der mangler ordren 43 A 40 mellem ordrerne i hac 47 og hac 48. Da koden helt er opbygget på grundlag af de adresser, hvori ordrerne skal lagres, nytter det jo ikke bare at putte den manglende ordre ind og derefter ændre adresserne på de efterfølgende ordrer. (Eventuelt skal talrige adresser ændres). Man klarer sig da i stedet ved anvendelse af en parentes.

46		120 A 10	47	Man fjerner den korrekte ordre i hac 47 og skriver i stedet 120 A 10 (idet vi antager at hac 120, 121 og 122 er ledige). Altså hoppes der til hac 120.
120	opr. indh.af hac 47	43 A 40	121	
122	48 A 10			

1) I hac 120 anbringer man det oprindelige indhold af hac 47, og

2) i hac 121 indsætter man den glemte ordre, her 43 A 40.

3) Endelig slutter man med 48 A 10 i hac 122.

Øvelse 5.50 $y = a_8x^8+a_7x^7+ \ldots +a_1 \cdot x+a_0$ skal lagres i hec 96.

x står i hec 98, $a_0, a_1, \ldots, a_8$ i henholdsvis hec 100, hec 102, ... , hec 116.

Vi benytter, at $y = (\ldots (((a_8x+a_7)x+a_6)x+a_5) \ldots a_1)\ x+a_0$

Der skal benyttes løkkeregning. De hjælpepolynomier, der dannes, skal benyttes som multiplikatorer.

Øvelse 5.51 Samme opgave som i øvelse 5.50. Denne gang skal x benyttes som multiplikator.

Øvelse 5.52 Alle halvcellerne fra 1000 til 1199 indeholder DASK-tal
Der ønskes en kode, der dels giver antallet af negative tal, dels bytter disse ud med deres numeriske værdi.
Obs! $|-1| \sim 0.111\ldots1 = -C(2040)$.

Øvelse 5.53 En kode tænkes lagret fra hac 0 til hac 77. Ordrerne i halvcellerne 19, 31, 54 og 63 skal bevirke udhop til en undersekvens, hvis første og sidste ordre er lagrede i henholdsvis hac 307 og hac 341. Efter regningerne i denne skal der hoppes tilbage til henholdsvis hac 20, 32, 55 og 64.
Idet der skal anvendes Wheelerhop, skal følgende spørgsmål besvares.

1) Hvilke ordrer skal der stå i hac 18, 19, 30, 31, 53, 54, 62 og 63?

2) Hvordan skal begyndelsen og slutningen af undersekvensen se ud?

Øvelse 5.54 Samme opgave som i øvelse 5.53, men med en subsekvens lagret i hac 513-hac 577. Undersekvensen benytter ikke IRD, men forudsætter indhop fra hovedsekvensen ved benyttelse af 16-hop.

Øvelse 5.55 Samme opgave som 5.54. Undersekvensen benytter IRD.

KAPITEL 6.

Ydre enheder og kontrolbordet.

6.1 Ved de ydre enheder forstår vi de apparater, der benyttes til at overføre information til og fra arbejdslageret. De ydre enheder må nødvendigvis omfatte indlæsnings- og udlæsningsapparatur. Derudover har de fleste elektronregnemaskiner et reservelager for arbejdslageret.

De ydre enheder i DASK er, som nævnt side 1.11:

Fotoelektrisk strimmellæser til indlæsning af ordrer og tal kodet i 5-hulstrimmel.

Elektrisk skrivemaskine og perforator til udlæsning af resultater, idet disse henholdsvis skrives, eller hulles i strimmelen.

Tromlelager og magnetisk bånd som reservelager for arbejdslageret.

Hulkortenheder til ind- og udlæsning af data fra og til hulkort.

Disse forskellige ydre enheder skal nu omtales i den udstrækning der kræves, for at koderen kan anvende de operationer, der knytter sig til dem. Operationerne og konkrete eksempler på deres anvendelse findes i det følgende kapitel 7.

6.2 Hulstrimmelen.

Som hulstrimmel benyttes en fjernskriverstrimmel med plads til 5 huller i hver række på tværs af strimmelen. Desuden indeholder hver række et mindre føringshul.

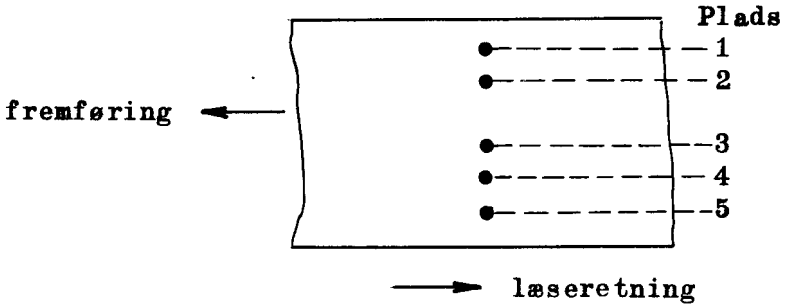
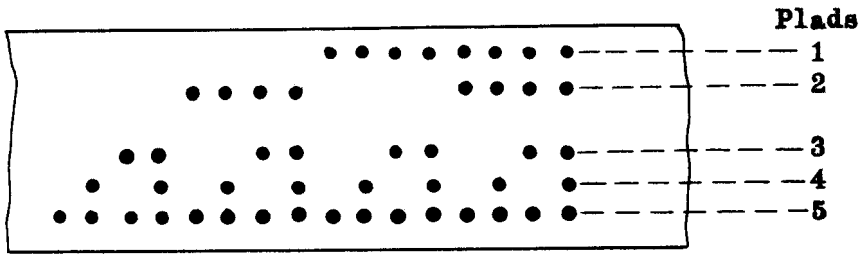


fig. 6.1 Strimmelen.

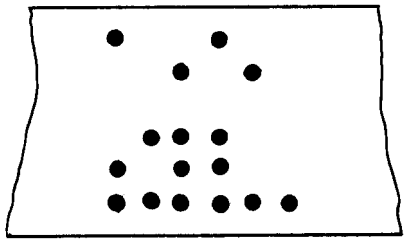
Sammenhængen mellem de 16 forskellige tegn, (tallene 0,1,...,9 og bogstaverne A,B,C,D,E og F) ved hjælp af hvilke vi udtrykker alle ordrer til DASK, og hullerne i en række er vist på fig. 6.2, hvor • betyder et hul på den pågældende plads.



0 1 2 3 4 5 6 7 8 9 A B C D E F

Fig. 6.2

Eksempel. Ordren 927 B 40 vil da, på strimmelen, se således ud



9 2 7 B 4 0

Fig. 6.3

6.3 Indlæsning ved hjælp af strimmellæseren.

Når et program er kodet færdigt, stanses det ordre for ordre (eller rettere, tegn for tegn) på strimmelen. Den færdige strimmel føres nu gennem strimmellæseren, der læser tegnene et for et. Et særligt indlæseprogram, der i forvejen findes i maskinen, sørger nu for, at ordrerne anbringes i konsekutive halvceller i lageret.

De læsere, der måtte ønske yderligere forklaring af indlæseprogrammets virkemåde, henvises til afsnit 10.4.

6.4 Udlæsning på strimmel eller på skrivemaskine.

Efter endt beregningsgang kan resultaterne, der nu findes i maskinens lager, hulles på strimmelen. Et udlæseprogram dirigerer en perforator, således at strimmelen hulles i overensstemmelse med sammenhængen mellem huller og tegn (fig. 6.2).

Den således hullede strimmel indsættes senere i en særlig skrivemaskine, en Flexowriter, der, styret af hullerne i strimmelen, omsætter disse til sædvanlige taltegn og bogstaver.

Resultaterne kan også udlæses direkte til en elektrisk skrivemaskine; denne dirigeres ligeledes af det lagrede udlæseprogram.

I DASK kan det være den samme ordre, der bevirker hulning i strimmel og udskrift på skrivemaskine.

Ved hjælp af en omskifter på kontrolbordet kan koderen vælge den af de to muligheder han foretrækker, eller han kan lade begge former for udskrift ske samtidigt.

Ved automatisk udskrift til skrivemaskinen er det imidlertid påkrævet at kunne styre denne. Ligeledes er det nødvendigt at have visse tegn til rådighed (f.eks. +,-,.) hvis ikke arbejdet med at tyde skrivemaskinens tal skal blive helt overvældende.

Som det vil fremgå af det følgende kapitel 7, findes der operationer, der kan få skrivemaskinen til at skrive +, - o.s.v. samt styre denne (vognretur, tabulator, mellemslag).

For også at sætte strimmelen i stand til på samme måde at styre Flexowriteren er det da nødvendigt, at visse kombinationer af huller på strimmelen kan give den information om at slå mellemslag, skrive + m.v.

Hulkoden (fig. 6.2) udvides da, hvad der er muligt, idet 5 huller giver $2^5 = 32$ kombinationer, af hvilke vi kun har anvendt de 16. Den udvidede hulkode ses på fig. 6.4. Vi bemærker, at der endnu er 7 ubenyttede kombinationer.

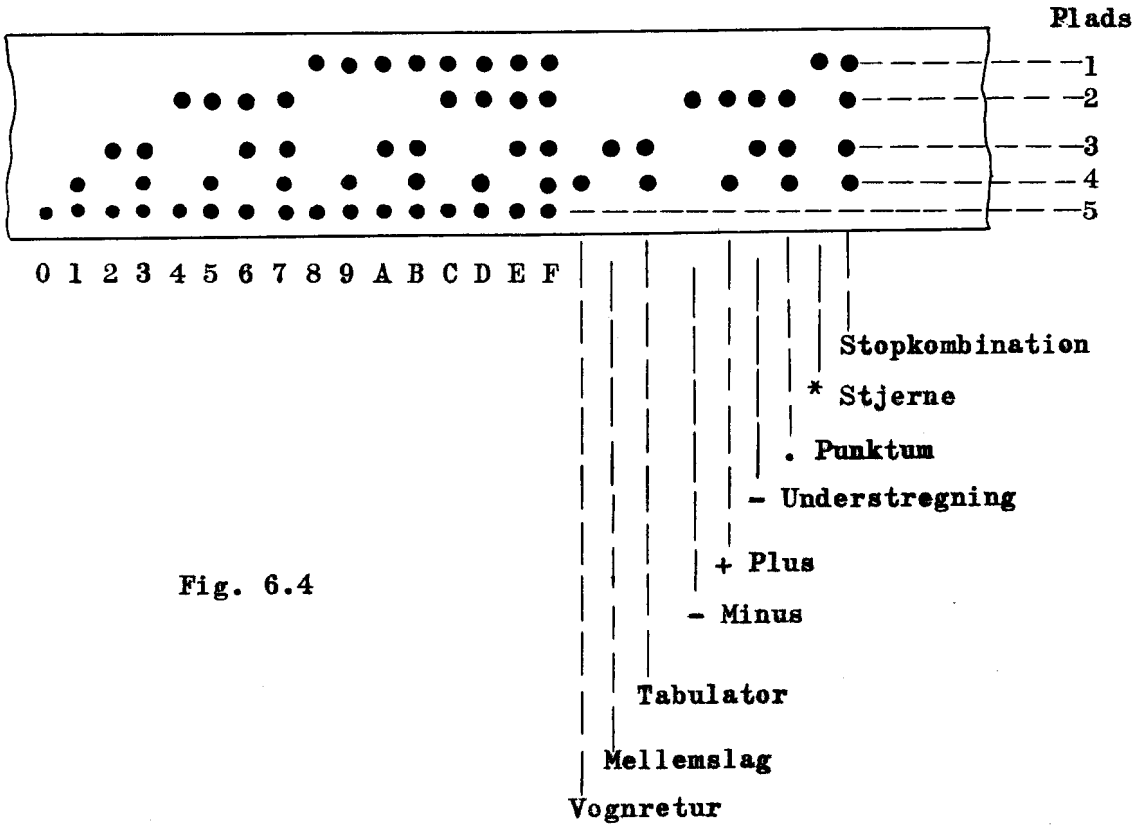


Fig. 6.4

Stopkombinationen er udelukkende et signal til Flexowriteren.

Vi bemærker, at f.eks. den ordre, der får skrivemaskinen til at gøre mellemslag, samtidig kan få perforatoren til at hulle strimmelen i overensstemmelse med hulkoden.

For fuldstændighedens skyld kan det tilføjes, at en strimmel, der er udlæst i overensstemmelse med den udvidede hulkode, ikke alene kan læses af Flexowriteren, men også - hvis man senere ønsker at regne videre på de opnåede resultater - kan indlæses i DASK. Ved denne indlæsning negligerer strimmellæseren så alle rækker, der ikke har et hul på plads 5.

6.5 Tromlelageret

Arbejdslageret har, som nævnt, en kapacitet på 1024 helord. For mange programmer vil denne lagerplads ikke være tilstrækkelig til at kunne rumme både programmets ordrer og de tal, der skal anvendes i regningerne.

Maskinen er derfor forsynet med endnu et lager, tromlelageret. I DASK har tromlelageret en kapacitet på 8192 helord. Dette svarer til 8 gange indholdet af arbejdslageret, og kapaciteten kan principielt forøges flere gange, hvis det viser sig påkrævet, hvorimod arbejdslagerets kapacitet ikke kan udvides uden væsentlige ændringer af hele maskinen (f.eks. ville lagerplads til dobbelt så mange halvord jo kræve, at man, for at angive adresse, måtte have 12 binære cifre til rådighed i stedet for de 11 man har.).

Overførsel mellem de to lagre sker gennem AR, på den måde at 32 helord overføres i serie umiddelbart efter hinanden. Disse udfylder en kanal på tromlen. Tromlelageret indeholder ialt 256 kanaler med numrene 0,2,4,...510. (Det kan iøvrigt bemærkes, at tromlelageret i DASK egentlig består af 2 tromler, der hver har 128 kanaler.)

Til overførsel af de omtalte 32 helord, kræves der to ordrer. Den førstes adresse, d.v.s. m = n + C(IR), skal være nummeret på den tromlekanal, der skal sættes i forbindelse med arbejdslageret, og operationsdelens indhold får maskinen til at sætte den pågældende tromlekanal i forbindelse med AR.

I adressedelen i den anden ordre skal der som sædvanlig stå adressen m = n + C(IR) på en celle i arbejdslageret.

Maskinen kan da, styret af indholdet i operationsdelen,

1) anbringe det første helord fra den valgte kanal i helcelle m; umiddelbart derefter det næste helord fra kanalen i helcelle m + 2 o.s.v., indtil samtlige helord fra kanalen er anbragt i konsekutive helceller i arbejdslageret, eller

2) tage helordet i celle m og anbringe dette på første plads i den valgte kanal; umiddelbart derefter helordet i celle m + 2 og anbringe dette på anden plads i kanalen, o.s.v. indtil alle kanalens 32 pladser er fyldt op.

Det fremhæves, at maskinen først går videre til næste ordre, når en kanal er helt tømt eller helt fyldt op.

De to nævnte ordrer behøver ikke at følge umiddelbart efter hinanden i programmet. Når den første ordre engang er gennemløbet, vedbliver den pågældende tromlekanal at være tilknyttet indtil en anden tromlekanal (eller en anden ydre enhed) vælges.

Endelig skal det bemærkes, at overførselen mellem de to lagre ikke behøver foregå med helord. Angående dette forhold henvises til afsnittet om ordrens opbygning samt operationsliste II.

KAPITEL 7.

Operationsliste II

Ud over de i kapitel 4 omtalte operationer er der i DASK indbygget endnu en række operationer, der benyttes f. eks. ved indlæsning til arbejdslageret, udlæsning til skrivemaskine, overførsler mellem arbejdslageret og tromlelageret m.v. Disse operationer, der alle vedrører maskinens ydre enheder, beskrives i dette kapitel.

Opstillingen og beskrivelsen af operationerne er foretaget på samme måde som i kapitel 4.

Vi bemærker, at DASK, inden indlæsningen af en kode, normalt ikke er tom. For at lette kodearbejdet er der i forvejen lagret et indlæseprogram, et udlæseprogram m.v. i maskinen. Dette forhold gør, at koderen næppe har brug for at lære betegnelserne for alle operationerne i dette kapitel udenad, men i stedet betragter operationsliste II som en opslagstabel.

Læs 10 rækker fra strimmel.

Operationsbetegnelse: 19

Kode: n,I,19

Virkning: Først læses et helord (= 10 sedecimale cifre = 10 rækker på strimmelen) ind i AR. Er m lige læses helordet derefter til hec m. Er m ulige, læses C(ARh) til hhac m.

Indholdet ændres i: AR samt for
m lige: hec m
m ulige: hhac m

Tid: ca. 450 AT.

Læs 10 rækker fra strimmel.

Operationsbetegnelse: 39

Kode: n,I,39

Virkning: Først læses et helord (= 10 sedecimale cifre = 10 rækker på strimmelen) ind i AR. Er m lige læses C(ARv) derefter til vhac m. Er m ulige, læses C(ARv) derefter til hhac m.

Indholdet ændres i: AR samt for
m lige: vhac m
m ulige: hhac m

Tid: ca. 450 AT.

Læs 1 række fra strimmel.

Operationsbetegnelse: 59

Kode: x,x,59

(79 har samme virkning som 59)

Virkning: AR nulstilles. Derefter læses en række fra strimmelen (= 1 sedecimalt ciffer = 4 bits) ind i AR_{36-39}

Indholdet ændres i: AR

Tid: ca. 50 AT.*)

NB. m betegner den effektive adresse m = n+C(IR)

*) 50 AT er den tid der går, før næste ciffer kan indlæses, men efter 3 AT er DASK fri, de øvrige additionstider kan eventuelt udnyttes til mellemliggende regninger.

Anvendes en af følgende otte operationer: 1A,3A,5A,7A,1B,3B,5B og 7B, afhænger virkningen af den stilling, en omskifter på kontrolbordet indtager. Der er tre muligheder:

Stilling 0, stilling S + P og stilling P.

Man må følgelig, for at kunne bedømme virkningen af en af disse operationer, vide i hvilken af de tre stillinger omskifteren befinder sig, eller bedre: samtidig med at operationen vælges, bestemmer man omskifterens stilling.

Tryk (eller stans) ciffer

Operationsbetegnelse: 1A

Kode: x, x, 1A

Virkning: Stilling 0 og Stilling S + P: $C(AR_{36-39})$ trykkes som 1 sedecimalt ciffer.

Stilling P: $C(AR_{36-39})$ stanses som 1 række på strimmelen.

Tid:

Skrivemaskine ca.	1400 AT. *)
Perforator I: ca.	700 AT.
Perforator II: ca.	120 AT.

Stans (eller tryk) ciffer

Operationsbetegnelse: 3A

Kode: x, x, 3A

Virkning: Stilling 0: $C(AR_{36-39})$ stanses som 1 række på strimmelen.

Stilling S+P: $C(AR_{36-39})$ stanses og trykkes.

Stilling P: $C(AR_{36-39})$ stanses.

Tid:

Skrivemaskine ca.	1400 AT. *)
Perforator I: ca.	700 AT.
Perforator II: ca.	120 AT.

Tryk (eller stans) nul

Operationsbetegnelse: 5A

Kode: x, x, 5A

Virkning: AR nulstilles. Derefter som ved 1A.

Indholdet ændres i: AR

Tid:

Skrivemaskine ca.	1400 AT. *)
Perforator I: ca.	700 AT.
Perforator II: ca.	120 AT.

Stans (eller tryk) nul

Operationsbetegnelse e: 7A

Kode: x, x, 7A

Virkning: AR nulstilles. Derefter som ved 3A.

Indholdet ændres i: AR

Tid:

Skrivemaskine ca.	1400 AT. *)
Perforator I: ca.	700 AT.
Perforator II: ca.	120 AT.

NB. Pseudoadressen og indeksmærkningen irrelevant (f. eks. kan 0, A, benyttes)

*) De angivne tider er den tid, som kræves til mekanisk at udføre den pågældende operation. Maskinen er fri til at regne 5 - 6 AT efter afgivelsen af en 1A- eller 1B-ordre, forudsat at en tidligere afgiven 1A- eller 1B-ordre er afgivet mindst den ovenanførte operationstid før. Er dette ikke tilfældet må maskinen afvente frigøring af skrivemaskine- eller perforatorkredse. Man har ved denne anordning mulighed for at regne samtidig med at selve den mekaniske udskriftfunktion foregår.

NB. I tilknytning til brug af standardprogram for trykning bør varianterne 3A og 7A benyttes.

I de følgende fire operationer 1B, 3B, 5B og 7B har tallet m = n + C(IR) en speciel betydning. Som det fremgår af skemaet nederst på siden bestemmer dette hvilken typografisk operation der udføres, eller hvilket typografisk tegn, der trykkes eller stanses. Om der sker trykning eller stansning (eller begge dele) afhænger som før af omskifterens stilling på kontrolbordet.

Tryk (eller stans) tegn

Operationsbetegnelse: 1B

Kode: n, I, 1B

Virkning: Den ved m modulo 16 givne instruktion (jvf. nedenfor) adlydes i stilling 0 og stilling S+P af skrivemaskinen, i stilling P kun af perforatoren.

Tid:

Skrivemaskine ca.	1400 AT.*)
Perforator I: ca.	700 AT.
Perforator II: ca.	120 AT.

Stans (eller tryk) tegn

Operationsbetegnelse: 3B

Kode: n, I, 3B

Virkning: Den ved m modulo 16 givne instruktion (jvf. nedenfor) adlydes i stilling 0 af perforatoren, i stilling S+P af både skrivemaskine og perforatoren, og i stilling P kun af perforatoren.

Tid:

Skrivemaskine ca.	1400 AT.*)
Perforator I : ca.	700 AT.
Perforator II: ca.	120 AT.

Tryk (eller stans) tegn

Operationsbetegnelse: 5B

Kode: n, I, 5B

Virkning: AR nulstilles. Derefter som ved 1B.

Indholdet ændres i: AR.

Tid:

Skrivemaskine ca.	1400 AT.*)
Perforator I: ca.	700 AT.
Perforator II: ca.	120 AT.

Stans (eller tryk) tegn

Operationsbetegnelse: 7B

Kode: n, I, 7B

Virkning: AR nulstilles. Derefter som ved 3B.

Indholdet ændres i: AR

Tid:

Skrivemaskine ca.	1400 AT.*)
Perforator I: ca.	700 AT.
Perforator II: ca.	120 AT.

Funktionen af m modulo 16 i tryk- eller stanseoperationer fremgår af vedstående skema.
Skrivemaskinen udfører de nævnte instruktioner 1-4 og 6-8 og stopper ved de øvrige. Perforatoren huller strimmelen i overensstemmelse med m modulo 16 uden 5. hul, jævnfør afsnit 6.4.
Ved den strimmelstyrede flexowriter har alle værdier af m funktion.

m modulo 16	funktion
0	blank
1	ny linie (vogn retur)
2	mellemslag
3	tabulator
4	- (Disse tegn skrives;
5	+ understregningen skal
6	– sendes før det ciffer,
7	. der skal understreges).
8	*
9-14	(reserveret)
15	Stopkombination

---

*) Se kommentar foregående side

NB. I tilknytning til brug af standardprogrammer for trykning bør varianterne 3B og 7B benyttes.

Vælg ydre enhed

Operationsbetegnelse: 1C

Kode: n, I, 1C

(3C, 5C og 7C har samme virkning som 1C)

Virkning: m = n + C(IR) opfattes som nummeret på en bestemt ydre enhed (se skemaet umiddelbart herunder). Denne ydre enhed sættes i forbindelse med aritmetisk enhed, men der sker ingen overførsler mellem den aritmetiske og den ydre enhed før dette beordres.
NB. Den pågældende YE står i forbindelse med AE indtil næste 1C-ordre gennemløbes.

m	ydre enhed
0	Tromlekanal nr. 0
.	.
510	Tromlekanal nr. 510
512 . 1278	reserveret udvidelse af tromlekanalantal
1280	Start af hulkortmaskine
1281	Stop hulkortmaskine
1282	Send kort i læsebane
1283	Send kort i hullebane
1284	Vælg bufferregister L_1
1286	- - L_3
1288	- - H_1
1290	- - H_3

efterfølges af ID-ordre

m	ydre enhed
1292	Vælg bufferregister H_2 efterfølges af IF-ordre
1294 . . . 1784	reserveret udvidelse af hulkort apparatur
1786 . . . 2047	reserveret magnetbåndenheder m.v.

Tid: $2\frac{1}{2}$ AT

Læs fra aktuel ydre enhed

KRAV:*) Den pågældende - aktuelle - ydre enhed skal være sat i forbindelse med aritmetisk enhed gennem en 1C-ordre. Kun når dette er opfyldt får vi følgende

Operationsbetegnelse: 1D

Kode: n, I, 1D

(5D har samme virkning som 1D)

Virkning: Operationen indledes med, at det første ord på den pågældende blok eller kanal overføres til AR. Derefter, vil, hvis m er lige, C(AR) overføres til hec m i lageret. Så hentes næste ord fra blokken til AR og derfra videre til hec m+2 i lageret. Således fortsættes, indtil samtlige ord i blokken eller kanalen står i konsekutive helceller i lageret. Er m ulige sker overførslen til AR som før. Derefter vil C(ARh) overføres til hhac m. Næste ord går til AR og C(ARh) derefter til hhac m+2. o.s.v.

Indholdet ændres i: AR, samt for
m lige: hec m, hec m+2, o.s.v. (indtil alle helord i den pågældende blok eller kanal er lagret).
m ulige: hhac m, hhac m+2, o.s.v. (indtil alle hhao på den pågældende blok eller kanal er lagret).

Tid: Tromle: Kanallængde 32 helord
Minimum 360 AT, maximum 720 AT.
Hulkortbuffer: ca. 80 AT
Magnetbånd:

*)Ved læsning fra magnetbånd skal yderligere 1E (3E, 5E, 7E) være benyttet.

Læs fra aktuel ydre enhed

Operationsbetegnelse: 3D

Kode: n, I, 3D

(7D har samme virkning som 3D)

*)Ved læsning fra magnetbånd skal yderligere 1E (3E, 5E, 7E) være benyttet

KRAV: *) Først tilknytning ved 1C (se iøvrigt 1D)
Virkning: Overførslen til AR som ved 1D. Er m lige vil C(ARv) første gang overføres til vhac m. Næste gang til vhac m+2 o.s.v. Er m ulige vil C(ARv) første gang overføres til hhac m. Næste gang til hhac m+2 o.s.v.
Indholdet ændres i: AR, samt for
m lige: vhac m, vhac m+2, o.s.v. (indtil alle vhao på den pågældende blok eller kanal er lagret).
m ulige: hhac m, hhac m+2, o.s.v. (indtil alle vhao på den pågældende blok eller kanal er lagret).
Tid: Tromle: Kanallængde 32 helord. Minimum 360 AT maximum 720 AT
Hulkortbuffer:
Magnetbånd:

Skal der læses fra eller skrives på et af magnetbåndene, tilknyttes det aktuelle magnetbånd først ved hjælp af operation 1C. Derefter må yderligere en bestemt blok på det aktuelle magnetbånd tilknyttes.

Søg blok på magnetbånd

Operationsbetegnelse: 1E

Kode: n, I, 1E

KRAV: Først tilknytning ved 1C (se iøvrigt 1D)
Virkning: m lige:

Indholdet ændres i: AR.

Tid:

NB. Næste blok på det pågældende magnetbånd vil være tilsluttet og klar til brug efter anvendelsen af den søgte blok.

Søg blok på magnetbånd

Operationsbetegnelse: 3E

Kode: n, I, 3E

KRAV: Først tilknytning ved 1C (se iøvrigt 1D)
Virkning:

Indholdet ændres i: AR.

Tid:

NB. Næste blok på det pågældende magnetbånd vil være tilsluttet og klar til brug efter anvendelsen af den søgte blok.

Søg blok på magnetbånd

Operationsbetegnelse: 5E

Kode: n, I, 5E

KRAV: Først tilknytning ved 1C (se iøvrigt 1D)
Virkning:

Indholdet ændres i: AR

Tid:

Søg blok på magnetbånd

Operationsbetegnelse: 7E

Kode: n, I, 7E

KRAV: Først tilknytning ved 1C (se iøvrigt 1D)
Virkning:

Indholdet ændres i: AR

Tid:

Skriv på aktuel ydre enhed

Operationsbetegnelse: 1F

Kode: n, I, 1F

(5F har samme virkning som 1F)

*)Ved skrivning på magnetbånd skal yderligere 1E (3E, 5E, 7E) være benyttet.

KRAV: *) Som ved 1D. (Først tilknytning ved 1C)
Virkning: m lige: C(hec m) overføres til AR, og C(AR) føres videre til første plads i den aktuelle blok eller kanal. Derefter overføres C(hec m+2) til AR og derfra videre til næste plads i den aktuelle blok eller kanal o.s.v. indtil hele blokkens eller kanalens helceller er fyldt op.

m ulige: C(hhac m) overføres til ARh i tom AR. Derefter helt som for m lige, idet hec skal erstattes med hhac.

Indholdet ændres i: AR og den aktuelle ydre enhed.

Tid: Tromle: Kanallængde 32 helord
Minimum 360 AT, maximum 720 AT.
Hulkortbuffer:
Magnetbånd:

Skriv på aktuel ydre enhed

Operationsbetegnelse: 3F

Kode: n, I, 3F

(7F har samme virkning som 3F)

*)Ved skrivning på magnetbånd skal yderligere 1E (3E, 5E, 7E) være benyttet.

KRAV: *) Som ved 1D. (Først tilknytning ved 1C)
Virkning: m lige: C(vhac m) til ARv i tom AR og C(AR) derefter videre til første plads i den aktuelle blok eller kanal. Derefter overføres C(vhac m+2) til ARv og derfra videre til næste plads i den aktuelle blok eller kanal o.s.v. indtil hele blokkens el.er kanalens vhac er fyldt op.

m ulige: C(hhac m) til ARv i tom AR. Derefter helt som for m lige idet vhac erstattes med hhac.

Indholdet ændres i: AR og den aktuelle ydre enhed.

Tid: Tromle: Kanallængde 32 helord
Minimum 360 AT, maximum 720 AT.
Hulkortbuffer:
Magnetbånd:

I tilknytning til operationsliste II følger nu nogle simple eksempler og øvelser, i hvilke vi anvender de ydre enheder.

Operationerne 19, 39 og 59 anvendes normalt ikke af koderen, idet der i DASK, som tidligere omtalt, inden den aktuelle kode indlæses, er lagret et særligt indlæseprogram. (De nævnte indlæseprogrammer "oversætter" ordrer og decimale tal til det binære talsystem).

Anvendelse af 1A, 5A og 1B.

Eksempel 7.1 $a = a_0, a_1, a_2, \ldots, a_{39}$, der tænkes lagret i hec 100, skal trykkes decimalt med 8 decimaler.
Der foretages ingen afrunding af a.
Omskifteren på kontrolbordet i stilling 0.
I hec 102 lagres $10 \cdot 2^{-39}$

	0	1	A 1B	Vogn retur
	1	100	A 42	|a| → AR
17 ←	2	17	A 51	hop på minus d.v.s. hvis a = - 1
	3	100	A 44	a → AR og MR
7 ←	4	7	A 51	hop på minus d.v.s. hvis a < 0
	5	5	A 1B	tryk +
9 ←	6	9	A 10	hop
4 →	7	100	A 45	- a → AR og MR hvis a < 0
	8	4	A 1B	tryk minus
6 →	9	0	A 5A	tryk 0
	10	7	A 1B	tryk .
	11	2040	A 35	- 8 til IRB
15 →	12	1	B 35	C(IRB) + 1 → IRB
	13	102	A 4A	1. ciffer i AR_{36-39}
	14	0	A 1A	tryk
12 ←	15	12	A 33	hop hvis C(IRB) ≠ 0
	16	22	A 30	stop
2 →	17	4	A 1B	tryk minus
	18	2	A 40	?
	19	20	A 0D	?
	20	0	A 1A	?
	21	22	A 30	stop

Hvad sker der ved ordrerne 18, 19 og 20?

Øvelse 7.2 Indholdet i hec 238 opfattes som et tal med kommaet placeret mellem pos. 13 og pos. 14 (d.v.s. det står i hec 238 med skalafaktoren 2^{-13}, jvfr. afsnit 8.2 og 8.3). Tallet skal trykkes decimalt med 4 decimaler. Ingen afrunding inden trykningen.

Øvelse 7.3 Samme opgave som i 7.2 men denne gang skal tallet afrundes inden trykningen.

Øvelse 7.4 Alle tallene i halvcellerne 700 - 749 skal trykkes i overensstemmelse med følgende typografiske disposition for trykningen. Lagringsadresserne skal også trykkes. Tabulatorindstillingen på skrivemaskinen benyttes.

700	C(700)	701	C(701)
702	C(702)	703	C(703)
-	-	-	-
-	-	-	-
748	C(748)	749	C(749)

Der laves 3 specielle trykprogrammer:

T1 der trykker lagringsadresserne
T2 der trykker de tal, der ikke er - 1
T3 der trykker - 1

Dernæst laves en hovedsekvens, der først sørger for skrivemaskinens rigtige stilling og derefter sørger for hop til den rette tryksekvens. 16-hoppet bør benyttes i tryksekvenserne.

Anvendelse af 1C, 1D og 1F

Eksempel 7.5 Tromlen antages fyldt. Ethvert af dens 256·32 helord skal erstattes med sin numeriske værdi.

$C(2040) = -(1 - 2^{-39})$ (permanent konstant)

Sekvensen anvender 32 hec som arbejdsceller. Disse er lagt i forlængelse af sekvensen.

```
       0   512 A 35     2·256 →IRB
12 →   1  2046 B 35     C(IRB) - 2 →IRB
       2     0 B 1C     Tromlekanalen, hvis nummer er lig
                        C(IRB), vælges.
       3    14 A 1D     De 32 helord på kanalen læses til
                        32 hec, begyndende med hec 14.
       4    64 A 55     2·32 til IRC
10 →   5  2046 C 55     C(IRC) - 2 →IRC
       6    14 C 42     |C(14 +C(IRC))| eller - 1 i AR
 9 ←   7     9 A 11     hop på +
       8  2040 A 41     + 1 ~ 0,11.....1 →AR
 7 →   9    14 C 08     |C(n+ ..... ) | på plads i lageret
 5 ←  10     5 A 53     hop hvis C(IRC) ≠ 0
      11    14 A 1F     De numeriske værdier tilbage på tromlen
 1 ←  12     1 A 33     hop, hvis C(IRB)≠ 0
      13    78 A 30     Stop
      14  .........  }
       .            }
       .            }
       .            }   arbejdsceller
       .            }
       .            }
       .            }
      77  .........  }
```

Øvelse 7.6 Tromlen antages fyldt. Tæl antallet af tal, der ligger i intervallet $1.11.....100\ .....0 \leq x < 0$.

Øvelse 7.7 Tromlen antages fyldt. Find det <u>største halvord</u> og det <u>mindste halvord</u>, der findes på tromlen. De fundne halvord lagres i henholdsvis vhac 20 og hhac 21.

KAPITEL 8.

8.1 I kapitel 2 omtaltes den talværdi, indholdet i et register eller en celle tillægges, en værdi, der ligger i DASK-intervallet: $-1 \leq t < 1$. Da det imidlertid er klart, at man ikke kan være begrænset til at behandle opgaver, hvor tallene af sig selv ligger mellem -1 og +1, skal vi nu se, hvordan man sætter sig ud herover. Der vil blive vist ialt tre veje at gå, nemlig

transformation med skalafaktorer,+

regning med fast men anden kommaplacering end maskinens egen,

regning med såkaldt flydende kommaplacering: flydende regning.

For at belyse hele spørgsmålet vil vi indlede med at vise, hvad det er, der binder maskinen til intervallet $[-1, +1]$ og forklare, hvorfor man har valgt at lade de 40 positioner repræsentere tal netop i dette interval.

Vi tænker os, at intervallet f. eks. havde været $-2^k \leq t < 2^k$. Dette ville medføre, at kommaet i celler og registre var placeret mellem position k og k+1, mens selvfølgelig fortegnscifret stadig ville stå i position 0. For de indbyggede aritmetiske operationer ville en sådan kommaplacering have betydet følgende:

addition (subtraktion) ville forløbe uændret, idet spild optræder og behandles på samme måde ved en hvilken som helst kommaplacering.

multiplikation ville kræve, at det færdige produkt blev skiftet k positioner til venstre for at bringe dets komma på plads mellem position k og k+1. Spild vil kunne optræde for alle værdier af k undtagen $k \leq 0$, hvor vi udelukkende har at gøre med ægte brøker, og (ægte brøk)·(ægte brøk) = ægte brøk. Specielt vil der med k = 0 samtidig være opnået, at produktet står på plads uden først at skulle skiftes.

+) Der findes andre, men samtidig mere komplicerede transformationer; det ligger dog udenfor dette kompendiums rammer at komme ind herpå.

division kræver for alle k, at divisor eventuelt skiftes nogle pladser, s, til venstre, for at processen rent teknisk kan forløbe, nemlig så $2^s |\text{divisor}| \geq |\text{dividend}|$. Kvotienten skal derefter skiftes k-s positioner til højre (altså, når k-s er negativ, s-k positioner til venstre) for at bringe kommaet på plads. Kun når $2^k |\text{divisor}| \geq |\text{dividend}|$, vil der ikke være spild. Man ser, at for $k \geq 39$ er denne betingelse altid opfyldt. Kun når k = s skal kvotienten ikke skiftes for at bringes på plads, d.v.s. intet k kan sikre, at der ikke skal skiftes såvel før processen udføres som efter.

Ønsket om i så stor udstrækning som muligt at få operationerne til at give resultater, der igen ligger i intervallet $-2^k \leq t < 2^k$, gør, at man må foretrække kommaplaceringer enten til venstre for position 1 eller til højre for position 39. At gå udover en placering henholdsvis umiddelbart til venstre for position 1 eller umiddelbart til højre for position 39 giver ingen fordele, men tværtimod en dårligere udnyttelse af kapaciteten (i første tilfælde, hvor k er negativ, vil produkterne numerisk aldrig kunne komme over $(2^k)^2 < 2^k$, i andet tilfælde, hvor $k > 39$, gælder det, at kvotienterne aldrig vil overstige 2^{39}). Når man tager i betragtning, hvor simpelt multiplikation forløber med en kommaplacering mellem position 0 og 1, mens en sådan simpelhed aldrig kan opnås ved divisionen, forstår man, at valget er faldet på denne placering.

Samtidig er det blevet klart, hvorfor man ikke uden videre kan "lade som om" tallene i maskinen havde den kommaplacering, man rent øjeblikkelig måtte ønske: en anden (tænkt) placering kræver, at vi i forbindelse med multiplikation og division selv koder de skift, der er nødvendige (stod maskinens eget komma mellem position k og k + 1, var disse skift selvfølgelig indbygget i operationen). På den anden side ser man, at der intet er til hinder for at lade de binære cifre i maskinen repræsentere de tal, vi ønsker, når blot sådanne skift kodes, og man samtidig tager i betragtning, at multiplikationen ikke længere er spildfri. (Bemærk dog at vi også med DASK-tal kan få spild, nemlig med (-1) (-1))

8.2. Anvendelse af skalafaktorer

Det simpleste er tilfælde, hvor man forlods kan forsyne udgangsstørrelserne, konstanter som variable, med sådanne faktorer, skalafaktorer, at udgangsværdier såvel som mellemregninger og slutresultater alle kommer til at ligge i DASK-intervallet. Man har herved transformeret regningerne, der skal udføres, så de forløber i DASK-intervallet, og når det kan gennemføres, bør denne fremgangsmåde foretrækkes for andre, da det taget generelt gælder, at det fører til de korteste regnetider i maskinen.

I det følgende gennemgås et eksempel på brugen af skalafaktorer, og for de, der måtte ønske at få indblik i, hvordan overgangen fra 10-tals til 2-talssystemet i forbindelse med maskinens begrænsede kapacitet indvirker på regnenøjagtigheden, er eksemplet efterfulgt af en diskussion heraf.

I maskinen skal udregnes værdier af polynomiet

$$y = + 103.28872 + 148.015313 \cdot x - 1.1576389 \cdot x^2 + 0.00383525 \cdot x^3$$

hvor det altid gælder, at $|x| \leq 9.9$

Et hurtigt overslag giver, at $|y| < 1600$.

a. x transformeres til DASK-tal:

$$u = 10^{-1} \cdot x \quad , \qquad |u| \leq 0.99$$

og så bliver

$$y = + 103.28872 + 1480.15313 \cdot u - 115.76389 \cdot u^2 + 3.83525 \cdot u^3$$

b. koefficienterne transformeres til DASK-tal

$$v = 10^{-4} \cdot y = + 0.010328872 + 0.148015313 \cdot u - 0.011576389 \cdot u^2 + 0.000383525 \cdot$$

c. Herved er samtidig venstre side, $v = 10^{-4} \cdot y$, blevet DASK-tal, idet $|v| < 0.16$.

Dette sidste behøvede selvfølgelig ikke at være indtruffet, og man måtte så have brugt en endnu stærkere faktor.

Man overvejer let, at det i eksemplet her er unødvendigt at tage forbehold overfor hvordan udregningen ledes, for at sikre, at mellemregningerne også holder sig i DASK -intervallet (kun helt bizarre fremgangsmåder kunne give vanskeligheder).

Skal resultaterne bruges udenfor maskinen, må man gennem maskinens trykprogram (se kap. 15) sørge for, at de udregnede DASK-tal, v, afleveres med faktoren 10^4. Skulle samtidig de pågældende x-værdier trykkes, måtte man også her gennem trykprogrammet lade maskinen levere $10 \cdot u$.

Diskussion og anvendelse af 2-potenser som skalafaktorer

Er koefficienterne afrundede tal og anvendes x-værdier, der i 10-talsystemet er uafkortede (altså ikke $\frac{1}{3}$, $\sqrt{2}$ etc.), ser man, at det givne udtryk, ved udregning i 10-talsystemet, bestemmer y med en maksimal beregningsusikkerhed på:

$$5 \cdot 10^{-6} + 5 \cdot 10^{-7}(9.9) + 5 \cdot 10^{-8}(9.9)^2 + 5 \cdot 10^{-9}(9.9)^3 \sim 4 \cdot (5 \cdot 10^{-6}) = 2 \cdot 10^{-5},$$

gældende for det uafkortede regneresultat. Ved afrunding til 4 decimaler fås i ugunstigste fald

$$2 \cdot 10^{-5} + 5 \cdot 10^{-5} = 7 \cdot 10^{-5}.$$

Det til 4 decimaler afrundede resultat vil altså med sikkerhed være rigtigt indenfor $\pm$ 1 i sidste ciffer. Man må følgelig kunne stille som et krav, at beregningen i maskinen bevarer denne nøjagtighed.

Nu er det klart, at omsætningen af de givne (transformerede), decimale koefficienter og x-værdier til binære tal med 39 binaler kun kan foretages indenfor en fejl på $\pm \frac{1}{2} \cdot 2^{-39} = \pm 2^{-40} \sim \pm 0.91 \cdot 10^{-12}$.

Vi vil først se, hvad en sådan fejl i koefficienterne maksimalt kan bevirke, idet vi benytter den maksimale værdi for u:

$(5 \cdot 10^{-10} + 2^{-40}) + (5 \cdot 10^{-10} + 2^{-40})\ (0.99) + (5 \cdot 10^{-10} + 2^{-40})\ (0.99)^2 +$

$(5 \cdot 10^{-10} + 2^{-40})\ (0.99)^3 \sim 4 \cdot (5 \cdot 10^{-10}) + 4 \cdot 2^{-40} \sim 2 \cdot 10^{-9} + 3.6 \cdot 10^{-12}$ i v og altså

$2 \cdot 10^{-5} + 3.6 \cdot 10^{-8}$ i y, hvor første led selvfølgelig er den samme størrelse, som vi før kom til. Bidraget fra omregningen af koefficienterne til binære tal er således kun $3.6 \cdot 10^{-8}$, hvilket er helt uden betydning.

Dernæst vil vi se på den tilsvarende fejl stammende fra u. Ud fra de størrelser koefficienterne i polynomiet har, ser man, at det er nok at tage bidragene fra første og anden-grads leddene. Med en i maskinen direkte indført u-værdi giver disse led maksimalt: $+ 0.15 \cdot 2^{-40} + (0.012) \cdot 2 \cdot (0.99) \cdot 2^{-40} \sim 0.18 \cdot 2^{-40} \sim 0.17 \cdot 10^{-12}$ i v, d.v.s. $0.17 \cdot 10^{-8}$ i y selv, igen helt forsvindende.

Vi kan nu sammenstille de maksimale fejlbidrag:

$2 \cdot 10^{-5}$	fra det givne udtryk,
$3.6 \cdot 10^{-8}$	fra den binære omregning af koefficienterne,
$0.17 \cdot 10^{-8}$	fra den binære omregning af et i maskinen direkte indført x,
$5 \cdot 10^{-5}$	fra afrunding til 4 decimaler,

hvilket viser, at overgangen til 2-talsystemet her intet har betydet.

Ofte vil imidlertid de u-værdier, man arbejder med, være frembragt i maskinen selv, f.eks. ved at der til en udgangsværdi u_0 et vist antal gange adderes en tilvækst Δ, altså ved at $u = u_0 + n \cdot \Delta$. Lad os f.eks. tænke os, at vi i vort tilfælde kunne have n-værdier op til 50 000. Det ville betyde en maksimal usikkerhed i u på $5 \cdot 10^4 \cdot 2^{-40} = 4.6 \cdot 10^{-8}$. Gennem 1ste og 2den grads leddene indføres herved maksimalt

$+ 0.15\ (4.6 \cdot 10^{-8}) + 0.012 \cdot 2 \cdot (0.99) \cdot (4.6 \cdot 10^{-8}) \sim 8 \cdot 10^{-9}$ i v, d.v.s.
$8 \cdot 10^{-5}$ i y selv, og sammenstiller vi igen de maksimale fejlbidrag, får man nu

$$\left.\begin{array}{r} 2 \cdot 10^{-5} \\ 0.0036 \cdot 10^{-5} \\ 8 \cdot 10^{-5} \\ 5 \cdot 10^{-5} \end{array}\right\} \text{ fra omregningen til 39 binaler}$$

hvilket er en ret ugunstig balance mellem de a priori tilstedeværende og de af maskinen indførte fejl.

Det er let at se flere veje ud af denne vanskelighed, f.eks. kunne man indføre mere end den ene udgangsværdi, u_o, så n bliver mindre. Men vi vil vise, hvordan man ved at benytte 2-potenser i stedet for 10-potenser ved transformationen af den kritiske størrelse $\triangle$, radikalt kan omgå vanskelighederne. Først skal bemærkes, at 2-potenser som skalafaktorer i mange tilfælde kan være nyttige, alene fordi man med dem bedre kan gradere transformationen. F. eks. kunne man indvende, at vi med transformationen $v = 10^{-4} \cdot y$ ikke udnytter maskinens 39 binaler så godt, som vi burde, når vi stiller så forholdsvis store krav til nøjagtighed (antallet af betydende cifre er 8) og at $v = 2^{-11} \cdot y$ havde været bedre (så havde $|v| < 0.85$). Men iøvrigt er det klart, at en hvilken som helst konstant kan bruges som skalafaktor, og fordelen ved 10-potenser består egentlig kun deri, at overgangen til koefficienter etc. i det transformerede problem er så simpel.

Lad os dernæst tænke os, at $\triangle x = q \cdot 10^{-5}$, hvor q er et helt tal; vi får heraf $x = x_o + n \cdot q \cdot 10^{-5}$. Det, det nu gælder om, er at benytte en skalafaktor, som for alle hele n lader $n \cdot q \cdot 10^{-5}$ stå uafkortet i maskinens 39 binaler. Man ser, at dette vil være opnået med faktorer af typen $10^5 \cdot 2^{-p}$ (eller med $10^6 \cdot 2^{-p}$, $10^7 \cdot 2^{-p}$ o.s.v.). Nu skal p bare vælges så stor, at x bliver et DASK-tal; vi tager p = 20.

Herved bliver

$$u = x \cdot \frac{10^5}{2^{20}} \quad , \quad |u| \leq \frac{(9.9) \cdot 10^5}{2^{20}} < \frac{1}{1.048\ 576} \sim 0.95$$

og

$$y = +103.28872 + 148.015313 \cdot \left(\frac{2^{20}}{10^5}\right) \cdot u$$

$$-1.1576389 \cdot \left(\frac{2^{20}}{10^5}\right)^2 \cdot u^2 + 0.00383525 \cdot \left(\frac{2^{20}}{10^5}\right)^3 \cdot u^3$$

der udregnet giver

$$y = +103.288\ 72$$

$$+1552.053\ 048\ 443 \cdot u \quad (.000\ 005\ 243)$$

$$-\ 128.283\ 743\ 158 \cdot u^2 \quad (.000\ 005\ 498)$$

$$+\ \ 4.421\ 742\ 202 \cdot u^3 \quad (.000\ 005\ 765)$$

Tallene i parentes under koefficienterne viser, hvad disses maksimale afrundingsfejl bliver, når skalafaktoren anvendes, således at man hurtigt kan afgøre hvor mange cifre man vil medtage i den transformerede koefficient, for ikke at forøge fejlen mærkbart ved denne nye afrunding. Vi vælger at tage 7 decimaler i koefficienterne og sætter samtidig $v = 10^{-4} \cdot y$:

$$v = +0.010\ 328\ 872 + 0.155\ 205\ 304\ 84 \cdot u$$
$$-0.012\ 828\ 374\ 32 \cdot u^2 + 0.000\ 442\ 174\ 22 \cdot u^3$$

Uafhængig af n's værdi bliver her de maksimale bidrag til fejlen:

fra det givne udtryk	$2 \cdot 10^{-5}$
afrunding til 7 decimaler af de transformerede koefficienter: ($\sim \frac{1}{100}$ af ovenstående)	$0.02 \cdot 10^{-5}$

omregning af koefficienter til 39 binaler:

$$\left[2^{-40} + 2^{-40} \cdot (0.95) + 2^{-40}(0.95)^2 + 2^{-40} \cdot (0.95)^3\right] \cdot 10^4 \sim 4 \cdot 2^{-40} \sim 0.0036 \cdot 10^{-5}$$

omregning af $x = x_0 + n \cdot q \cdot 10^{-5}$ til 39 binaler:

$$\left[0.16 \cdot 2^{-40} + 0.013 \cdot 2 \cdot (0.95) \cdot 2^{-40}\right] \cdot 10^4 \sim 0.17 \cdot 10^{-8} \sim 0.0002 \cdot 10^{-5}$$

afrunding til 4 decimaler $5 \cdot 10^{-5}$

således at der denne gang "fra maskinen" kommer et fejlbidrag på højst $0.024 \cdot 10^{-5}$ mod før over $8 \cdot 10^{-5}$.

Skal resultaterne bruges udenfor maskinen, må man også her lade trykprogrammet levere $10^4 \cdot v$ og for x's vedkommende $\frac{2^{20}}{10^5} \cdot u = 10.48576 \cdot u$.

Vi gør for fuldstændighedens skyld opmærksom på, at den største nøjagtighed, trykprogrammet er indrettet til at reproducere et indhold i en celle med, er $\pm 5 \cdot 10^{-12}$. I vort tilfælde betyder det intet, da vi kun har brug for at lade y trykke med 4 decimaler ($\pm 5 \cdot 10^{-12}$ ville i y svare til $\pm 5 \cdot 10^{-8}$).

Man må forstå, at eksemplet her har stillet store krav til maskinens ciffermæssige kapacitet, ved mindre krav vil 10-potenserne oftest kunne vælges så hensigtsmæssigt, at der ingen vanskeligheder kommer.

8.3. Fast kommaplacering.

Hvis man, ved at tage et overblik over det problem der skal regnes, kan se, at alle størrelser, der optræder, også mellemregninger, vil være mindre end f.eks. 2^k, og man samtidig ikke behøver mere plads, end den 39 binære cifre giver (11-12 decimale cifre), betyder det, at man kan lade hele talmaterialet stå i maskinen som binære tal med kommaet mellem position k og k+1 og udføre regningerne direkte med de værdier, der er givet. Alle tal skal så blot indlæses til lageret med faktoren 2^{-k}, og hvad der senere skal ud af maskinen, må ved udlæsningen (trykningen) forsynes med faktoren 2^k. Kodningen kan under disse forhold gennemføres, uden at man behøver at tænke på spild, hverken ved additioner, multiplikationer eller divisioner, idet forundersøgelsen har sikret, at det ikke vil indtræffe. Men selvfølgelig må der sørges for skift ved multiplikation og division.

Vi vil vise, hvordan den enkelte multiplikation og division kan kodes, når denne fremgangsmåde benyttes.

Multiplikation: $a \cdot b = c$.

Kommaet er valgt mellem position k og k+1, og vi ved at $|a|, |b|$ og også $|c|$ er $< 2^k$.

Tallene står efter indlæsningen i maskinen som DASK-tallene: $a_1 = a \cdot 2^{-k}, b_1 = b \cdot 2^{-k}$ og c skal efter endt proces stå som $c_1 = c \cdot 2^{-k}$.

I første omgang udregner maskinen

$(a \cdot 2^{-k})(b \cdot 2^{-k}) = c \cdot 2^{-k} \cdot 2^{-k} = c_1 \cdot 2^{-k}$

altså skal, jævnfør afsnit 8.1, dette resultat skiftes k positioner tilvenstre for at blive til $c_1 = c \cdot 2^{-k}$; først så er cifrene placeret rigtigt omkring det valgte komma.

Kode:

$C(96) = a_1 = a \cdot 2^{-k}$

$C(98) = b_1 = b \cdot 2^{-k}$

$c_1 = c \cdot 2^{-k}$ ønskes i hec 100

0	96 A 44	a_1	til MR
1	98 A 4A	$b_1 \cdot a_1$	uafkortet i AR_{0-39}, MR_{1-39}
2	k A 4C	k skift tilvenstre i lang akkumulator	
3	100 A 08	c_1	til hec 100.

Division: $\frac{a}{b} = c$.

Igen er a,b og c numerisk $< 2^k$. Vi lader maskinen udregne en kvotient C(MR), der bestemmes af

$$\frac{a}{b} = \frac{a \cdot 2^{-k}}{b \cdot 2^{-k}} = 2\,\frac{\frac{1}{2}a_1}{b_{1N} \cdot 2^{-s}} = \frac{\frac{1}{2}a_1}{b_{1N}} \cdot 2^{1+s} = C(MR) \cdot 2^{1+s} = c$$

(jfr. eks. 5.37 side 5.10).

Da vi skal have $c_1 = c \cdot 2^{-k}$, så cifrene står rigtigt omkring det valgte komma, ser man, at C(MR), efter at være overført til AR, skal skiftes k-1-s positioner til højre. Hvis k-1-s er negativ, skal der skiftes s-(k-1) positioner til venstre.

$C(96) = a_1 = a \cdot 2^{-k}$
$C(98) = b_1 = b \cdot 2^{-k}$ c_1 skal til hec 100
$C(105) = (k-1)2^{-11}$. Arbejdsceller: 102 og 104v.

		0	98 A 40	$b_1 \longrightarrow$ AR
		1	104 A 0E	normaliser, s $\longrightarrow$ 104v
		2	102 A 08	$b_{1N} \longrightarrow$ 102
		3	96 A 40	$a_1 \longrightarrow$ AR
		4	1 A 0D	$\frac{a_1}{2}$ i AR
		5	102 A 0B	$\frac{a_1}{2} : b_{1N}$ i MR
		6	104 A 60	s $\longrightarrow$ AR
		7	105 A 21	s - (k-1) $\longrightarrow$ AR
13	$\longleftarrow$	8	13 A 51	hop på ÷
		9	11 A 29	s-(k-1) i adressepos. i 11.
		10	0 A 07	$\frac{a_1}{2} : b_{1N} \longrightarrow$ AR
		11	(0)A 0C	s-(k-1) venstre skift
18	$\longleftarrow$	12	18 A 10	hop
8	$\longrightarrow$	13	105 A 60	k-1 $\longrightarrow$ AR
		14	104 A 21	(k-1)-s $\longrightarrow$ AR
		15	17 A 29	(k-1)-s i adressepos. i 17
		16	0 A 07	$\frac{a_1}{2} : b_{1N} \longrightarrow$ AR
		17	(0)A 0D	(k-1)-s højre skift
12	$\longrightarrow$	18	100 A 08	c_1 på plads
		19	20 A 30	stop.

b skal være $\neq 0$ (men a kan godt være 0 modsat eks. 5.37).

8.4. Flydende regning.

Ved denne metode, som også kaldes regning med flydende binær komma, går man sådan frem, at man til hvert tal lader svare en individuel placering af kommaet i maskinens registre og celler. Oplysning om denne placering gives i form af et heltal, der viser mellem hvilke af maskinens positioner kommaet skal stå, og her bliver brug for tænkte positioner, såvel til venstre som til højre for de 40 egentlige positioner. Det er altså nødvendigt, for hvert tal, der optræder i regningerne, at lagre og arbejde med to størrelser: tallets cifre og heltallet, der angiver kommaet.

Fastsættelsen af disse størrelser til bestemmelse af en given talværdi, kan foretages på flere måder. Vi vil her vise, hvordan det er gjort ved et bestemt program-kompleks, hørende til DASK Normalleje I (se kap. 13), hvorimellem findes hjælpeprogrammer til brug ved flydende regning.

Til hvert tal, x, indføres to tal x' og x", således at

$$x = x' \cdot 2^{x''-1024}$$

hvor

$$\text{enten} \quad -1 \leqq x' \leqq -\frac{1}{2}$$
$$\text{eller} \quad x' = 0$$
$$\text{eller} \quad \frac{1}{2} \leqq x' < 1 ,$$

x', taldelen, er m.a.o. normaliseret. x", der kaldes eksponenten, og som er et heltal, er af praktiske grunde begrænset til $0 \leqq x'' \leqq 2047$. Til et vilkårligt x i intervallet $2^{-1025} \leqq |x| \leqq 2^{1023}$ svarer der efter dette et bestemt sæt (x',x"); dog ville når x = 0, ethvert sæt (0,x") kunne bruges, men man kan her pr. definition sætte x" = 0, altså $x = 0 = 0 \cdot 2^{0-1024}$, så der også i dette tilfælde er entydighed. For negative 2-potenser se dog kapitel 14.

Skrevet med 10-potenser bliver intervallet, det flydende interval:

$$2.78 \cdot 10^{-309} < |x| \lesssim 8.99 \cdot 10^{307}$$

En given talværdi, x, står efter dette i maskinen som cifrene i DASK-tallet x', men med kommaet mellem position (x"-1024) og (x"-1023).

Man kan også opfatte det hele på den måde, at der til tallet x er benyttet skalafaktoren $2^{-(x''-1024)}$. Efter dette består flydende regning i, at hvert tal får sin egen skalafaktor.

Det væsentlige ved selve udførelsen af regningerne med de flydende tal er nu, at man gennem programmet overlader arbejdet med bestemmelsen af den rette skalafaktor og den dertil svarende taldel til maskinen.

Det er klart, at en sådan fremgangsmåde giver en betydelig længere kode og forlænget regnetid. I næste afsnit vil det blive drøftet, hvornår det er hensigtsmæssigt eller endog nødvendigt at lade maskinen på denne måde selv bestemme de skalafaktorer, der skal anvendes.

Vi skal nu vise, hvordan addition, multiplikation og division kan kodes ved flydende regning.

Først skal gøres nogle bemærkninger om x". Det er næsten på forhånd klart, at bestemmelsen af x' og x" skal ske ved normalisering. Det betyder, at x" må knyttes til det tal, der efter normaliseringen står i en eller anden halvcelles adressedel med enheden i position 11 og angiver antallet af venstreskift, x" lagres derfor praktisk med enhed i position 11. Der skulle på denne måde umiddelbart kunne opereres med x"-værdier fra 0 til 4095 eller ved benyttelse af fortegn fra -2048 til +2047, så man i dette sidste tilfælde direkte kunne sætte $x = x' \circ 2^{x"}$. Kodetekniske grunde gør dog, at man i den celle, hvori x" lagres, foretrækker at have 0 i position 0 og samtidig udelukkende regner med positive værdier af x". På denne måde forklares både at intervallet er $0 \leq x" \leq 2047$, og at man anvender eksponenten x"-1024.

Det flydende interval vil være overskredet når $x" < 0$ eller $x" > 2047$, og kendetegnet herpå bliver derfor at x" viser cifret 1 i position 0 (ved en enkel addition, multiplikation eller division kan overskridelsen aldrig blive så stor, at der "igen" kommer 0 i position 0). Da det er nødvendigt på den ene eller den anden måde at skride ind, når dette indtræder, vil der i koderne blive indført et udhop, som tager sig heraf.

Det tilføjes, at når et resultat, x, af en regning har givet $x" < 0$, vil det i mange tilfælde være rimeligt at sætte $x = 0$, d.v.s. $x' = x" = 0$. Det ses at $x" < 0$ og $x" > 2047$ kan skilles fra hinanden ved, at $x" > 2047$ nok har 1 i position 0, men samtidig viser spild ($C(AR_{00}) = 0$).

Vi vil ikke i de koder, der bliver vist, tage stilling til om $x'' < 0$ skal bevirke, at x' og x" sættes til 0. Derimod behandler vi selvfølgelig den mulighed, at en eller begge operander er 0 og medtager de konsekvenser, det får.

Addition: $x = a+b$.

a' i hec 100, a" i hhac 103.

b' i hec 104, b" i hhac 107.

x' ønskes i hec 108, x" i hhac 111.

Man må regne med, at en af størrelserne a' eller b' skal "forskydes" før additionen af dem kan finde sted. Hvilken og hvor meget bestemmes således:

$$a + b = a' \cdot 2^{a''-1024} + b' \cdot 2^{b''-1024} = \begin{cases} (a' + b' \cdot 2^{b''-a''}) \cdot 2^{a''-1024} \\ (b' + a' \cdot 2^{a''-b''}) \cdot 2^{b''-1024} \end{cases}$$

I den første parentes er vist, hvor meget b' skal forskydes for at det forskudte tal kan adderes til a', i den anden parentes hvormeget a' skal forskydes for at kunne adderes til b'. Her skal selvfølgelig vælges den af de to muligheder, der benytter forskydning tilhøjre, andet lader sig ikke gøre, d.v.s.

$$a' + b' \cdot 2^{b''-a''} = a' + b' \cdot 2^{-(a''-b'')} \quad \text{hvis } a'' \geq b''$$

$$b' + a' \cdot 2^{a''-b''} = b' + a' \cdot 2^{-(b''-a'')} \quad \text{hvis } a'' (\leq) b''$$

Da skifteoperationen udfører det skrevne antal skift modulo 128, kan man ikke uden videre lade a"-b" eller b"-a" angive antallet af skift. I stedet kan man f.eks. lade a"-b" >38 bevirke, at a+b sættes til a (den korrekte værdi, indenfor 39 binaler, vil være enten $a-2^{-39}$, a eller $a+2^{-39}$, men der ses overalt i koden bort fra afrundingsspørgsmålet). Efter at summen er dannet, kommer spørgsmålet om at få den normaliseret, og her må fremgangsmåden rette sig efter, om der er opstået spild eller ikke, sådan som det vil fremgå af koden. Det er let at se, hvordan x" skal bestemmes af a" eller b" og antallet af venstreskift under normaliseringen.

Tilsidst bemærkes, at når a+b er eksakt 0, skal x" sættes til 0.

	0	110 A 48	hec 110 tømmes
	1	39 A 35	39 til vhac 110 's adressepositioner.
	2	110 A 34	
	3	103 A 60	hop, hvis a" $<$ b", gå videre hvis a" $\geq$ b"
	4	107 A 21	
16 ←	5	16 A 51	
	6	110 A 21	$(a"-b"-39)\cdot 2^{-11}$ i AR
43 ←	7	43 A 11	hop hvis a"-b" $\geq$ 39
	8	110 A 20	antal højreskift: $38 \geq a"-b" \geq 0$ til adressen i ordre 13
	9	13 A 29	
	10	103 A 60	a" til hhac 111
	11	111 A 28	
	12	104 A 40	$a' + b'\cdot 2^{-(a"-b")}$ i AR
	13	(0) A 0D	
	14	100 A 00	
27 ←	15	27 A 10	hop til normalisering
5 →	16	103 A 61	$(b"-a"-39)\cdot 2^{-11}$ i AR
	17	107 A 20	
	18	110 A 21	
48 ←	19	48 A 11	hop hvis b"-a" $\geq$ 39
	20	110 A 20	antal højreskift: $38 \geq b"-a" \geq 0$ til adressen i ordre 25
	21	25 A 29	
	22	107 A 60	b" til hhac 111
	23	111 A 28	
	24	100 A 40	$b' + a'\cdot 2^{-(b"-a")}$ i AR
	25	(0) A 0D	
	26	104 A 00	
15 → 33 ←	27	33 A 52	hop til normalisering hvis ikke-spild
	28	108 A 08	summen normaliseret (nemlig halveret) til hec 108
	29	2039 A 60	$+2^{-11}$ i ARv
	30	111 A 26	$x" = \begin{matrix} a" \\ b" \end{matrix}\}$ +1 til hhac 111
	31	n_1 A 51	hop til kontrol (i celle n_1), hvis det flydende interval er overskredet.
	32	n A 30	stop.
27 →	33	108 A 08	
	34	108 A 43	$-\lvert a'+b'\cdot 2^{-(a"-b")}\rvert$ eller $-\lvert b'+a'\cdot 2^{-(b"-a")}\rvert$ i AR
53 ←	35	53 A 11	hop hvis 0

	36	108 A 40	x' normaliseret til hec 108
	37	110 A 0E	
	38	108 A 08	
	39	110 A 61	x"=a" / b" -antallet af venstrekift → hhac 111
	40	111 A 26	
	41	n_1 A 51	hop til kontrol hvis det flydende interval er overskredet
	42	n A 30	stop
7 →	43	100 A 40	x sættes lig a
	44	108 A 08	
	45	103 A 60	
	46	111 A 28	
	47	n A 30	stop
19 →	48	104 A 40	x sættes lig b
	49	108 A 08	
	50	107 A 60	
	51	111 A 28	
	52	n A 30	stop
35 →	53	111 A 68	x" sættes lig 0
	54	n A 30	stop

Multiplikation: $x = a \cdot b$

Vi tænker os samme lagring som ved additionen.

Man har $a \cdot b = a' \circ b' \circ 2^{a''+b''} - 1024 - 1024$

$= a' \cdot b' \cdot 2^{(a''+b'' - 1024)} - 1024$

hvor det vil gælde at

enten $-1 < a' \cdot b' < -\frac{1}{4}$ eller $\frac{1}{4} \leq a' \circ b' \leq +1$ eller $a' \circ b' = 0$.

Heraf ses 1) at i tilfældet: $a' = b' = -1$ vil spild indtræffe $(a' \circ b' = +1)$.

2) at $a' \circ b'$ eventuelt må normaliseres for at blive til x'.

Ved 1) får man gennem et højreskift den korrekt normaliserede værdi x' nemlig $+\frac{1}{2}$.

Tilfælde, hvor en eller begge faktorerne er 0, må behandles særskilt, da x'' her skal tvinges til at blive 0.

	0	111 A 68	hhac 111 tømmes
	1	100 A 44	
	2	104 A 0A	$a' \circ b'$ til hec 108
	3	108 A 08	
	4	108 A 43	$-\lvert a' \circ b' \rvert$ til AR
20 ←	5	20 A 11	hop ud hvis (mindst) en af faktorerne er 0, $(C(108)=x', C(111)=x'')$.
11 ←	6	11 A 52	hop hvis tilfældet $a'=b'=-1$ ikke foreligger.
	7	108 A 08	$+\frac{1}{2}$ til hec 108 hvis $a'=b'=-1$
	8	2039 A 61	-2^{-11} til hhac 111, $(C(2039) = 2^{-11})$
	9	111 A 28	
14 ←	10	14 A 10	hop til udregning af x''
6 →	11	108 A 40	
	12	111 A 0E	$a' \circ b'$ normaliseret; antallet af venstreskift til hac 111
	13	108 A 08	
10 →	14	111 A 61	
	15	103 A 20	
	16	2043 A 21	$x'' = -2^{11} \circ C(111) + a'' - 1024 + b''$ til hhac 111.*)
	17	107 A 20	$(C(2043) = +\frac{1}{2} = 1024 \circ 2^{-11})$
	18	111 A 28	
n_1 ←	19	n_1 A 51	hop til kontrol (i celle n_1) hvis det flydende interval er overskredet
5 →	20	n A 30	stop

*) Bemærk at rækkefølgen af de fire ordrer 14 til 17 er meget væsentlig, idet en spildindikation ikke må forsvinde alene som følge af maskinens specielle behandling af $C(AR_{00})$, jvfr. øverst side 2.8.

Division: $x = \frac{a}{b}$.

Man har

$$x = \frac{a' \cdot 2^{a''-1024}}{b' \cdot 2^{b''-1024}} = \frac{a'}{b'} 2^{a''-b''} = \begin{cases} \frac{a'}{b'} \cdot 2^{(a''-b''+1024)} - 1024 \\ \frac{\frac{1}{2}a'}{b'} 2^{(a''-b''+1+1024)} - 1024 \end{cases}$$

$$= x' \cdot 2^{x''-1024}$$

Da både a' og b' er normaliserede, vil, når $|a'| \leq |b'|$, også $\frac{a'}{b'}$ være normaliseret*), og når $|a'| > |b'|$ vil $\frac{1/2\ a'}{b'}$ være normaliseret. Resultatet af divisionen vil altså i begge tilfælde være en kvotient, der af sig selv er normaliseret.

Tilfældet b = 0 må skilles helt fra. a = 0 (men b ≠ 0) behandles særskilt, da x'' så skal være 0.

Lagringen er som i de to foregående eksempler.

	0	111 A 48	hac 111 tømmes
	1	104 A 43	$-\|b'\|$ til AR
n_2 ←	2	n_2 A 11	hop ud hvis b = 0
	3	100 A 43	$-\|a'\|$ til AR
7 ←	4	7 A 51	hop hvis a ≠ 0
	5	108 A 48	x' = 0
	6	n A 30	divisionen færdig, stop
4 →	7	104 A 42	$\|b'\| - \|a'\|$ i AR
	8	100 A 03	
15 ←	9	15 A 11	hop til division
	10	2039 A 60	$+2^{-11}$ til hhac 111. ($C(2039)=2^{-11}$).
	11	111 A 28	
	12	100 A 40	$\frac{1}{2}a'$ i AR
	13	1 A 0D	
16 ←	14	16 A 10	hop til division
9 →	15	100 A 40	x', d.v.s. $\frac{a'}{b'}$ eller $\frac{\frac{1}{2}a'}{b'}$, dannes og lagres
14 →	16	104 A 0B	
	17	0 A 07	
	18	108 A 08	
	19	2043 A 60	x'' = 1024+{1 / 0} -b''+a'' dannes i ARv og lagres
	20	111 A 20	(angående ordrerne 19 til 22 se fodnote på forrige side)
	21	107 A 21	
	22	103 A 20	
	23	111 A 28	$C(2043) = \frac{1}{2} = 1024 \cdot 2^{-11}$.
n_1 ←	24	n_1 A 51	hop til kontrol (i celle n_1) hvis det flydende interval er overskredet
	25	n A 30	stop

*) idet selv a' = b' i maskinen giver $\frac{a'}{b'} = 0.11.....1 < 1$!

8.5 Diskussion.

Vi skal i det følgende give retningslinier for bedømmelsen af, hvornår et problem bedst lader sig behandle efter den første, anden eller tredje af de metoder, der er vist.

Som disse er blevet fremstillet, har der kun været tænkt på opgaver, hvor kravet til antallet af betydende cifre ingen steder i regningerne overstiger de 39 binaler. Overfor en sådan sprængning af maskinens ciffer-kapacitet, som dette ville være, vil alle tre forholde sig i det væsentlige ens: de ville alle i den form, hvori de er vist, kræve betydelig udbygning i kodemæssig henseende for at gøre det muligt at udregne selv de simpleste ting, (lagring af og operationer på tal i dobbeltceller: regning med dobbelt nøjagtighed). Vi lader derfor stadig et sådant forøget krav til cifferantallet ude af betragtning.

1. Når hvert skridt i regningerne kan overses i den forstand, at man kan give sikre vurderinger af de talværdier, der opstår, vil skalafaktorer kunne anvendes.

   De tilfælde, hvor en fast kommaplacering kan komme på tale, falder ind herunder, nemlig som de relativt simplere tilfælde, hvor størrelsernes "samlede cifferkrav" (bestemt af det yderste ciffer tilvenstre og det yderste tilhøjre, det bliver nødvendigt at medtage) holder sig indenfor 39 binaler.

   Der er således i princippet intet i vejen for, når en opgave er kodet med anvendelse af fast komma, at den så også kunne være kodet ved anvendelse af skalafaktorer og skiftene derved undgået. Det omvendte gælder derimod ikke.

2. Den anden mulighed er, at de enkelte skridt i regningerne rent principielt ikke kan vurderes, f.eks. ved at der optræder divisorer, for hvilke der ikke kan gives nogen numerisk nedre grænse større end 0. Her er området, hvor flydende regning kommer til anvendelse. I virkeligheden kan man betragte denne regning som den mest gennemførte form for brug af skalafaktorer, idet størrelserne ikke blot deles i kategorier, der hver har sin bestemte skalafaktor, men som vi ved hvert tal overhovedet får sin individuelle skalafaktor, og dette besørges af maskinen selv, uden at koderen behøver at kende endsige have forudset dens størrelse.

Det er på forhånd klart, at kan man ved en opgave anvende skalafaktorer eller fast kommaplacering, kan den også behandles ved anvendelse af flydende regning, hvorimod det altså må understreges, at det omvendte ikke gælder.

Det kan være et spørgsmål, om man kun vil anvende flydende regning, når andet principielt ikke er muligt, eller om man trods den store forøgelse i køretid, vil anvende denne form også i tilfælde, hvor de nødvendige skalafaktorer blot er blevet talrige og bl.a. ikke indskrænker sig til (som formuleret i afsnit 8.2) kun at skulle påføres udgangsværdierne, men også må påføres grupper af mellemresultater undervejs. Afgørelsen kan være afhængig af mange ting, som f.eks., hvor store forberedelser problemet tåler for at være løst økonomisk.

Betragtninger af lignende art gør sig gældende ved valget mellem at anvende skalafaktorer eller fast komma. Rent kodemæssigt kunne det se ud som om skalafaktorer altid burde foretrækkes, men så enkle er forholdene ikke. Der vil være tilfælde, hvor multiplikationer og divisioner kun forekommer sjældent, hvorfor besværet med at lade skiftene udføre ikke er værd at tage i betragtning. Ligeledes kan det på grund af talstørrelsernes natur i visse tilfælde være en væsentlig lettelse når f.eks. faktorer og produkt optræder med samme faktor 2^{-k} i maskinen. Videre kan det ske, at man vil have trykt værdier fra mange af de forskellige kategorier, talmaterialet deles i, og at man derfor ved anvendelse af skalafaktorer kunne komme ud for at skulle bruge adskillige faktorer under trykningen, mens der ved fast kommaplacering normalt kun vil være den ene faktor 2^{k}.

Det er klart, at man til løsning af en opgave kan blande de forskellige metoder, således at man til de enkelte afsnit udvælger den, der passer bedst. På denne måde kan der opstå overgangsformer, hvor man f.eks. vil kunne opfatte et og samme ordreforløb snart som udtryk for en fast kommaplacering, snart som udtryk for, at der anvendes en skalafaktor. Med den i det foregående gennemførte skarpe inddeling i tre metoder, er der da også kun tænkt på at skabe en inddeling, som kan lette overblikket og således formindske vanskelighederne ved at træffe et konkret valg i de ofte meget komplicerede situationer, der opstår i forbindelse med numeriske arbejder.

KAPITEL 9.

Kodning af ordrer. Den ydre ordrekode.

9.1 Indledning.

Udover de tidligere omtalte tre bestanddele af enhver ordre skal vi i indeværende kapitel omtale endnu et kodesymbol, etiketteмærket, der kan benyttes ved nedskrivningen af ordren. Vi fremhæver, at medens den skrevne - ydre - ordre således kan indeholde fire bestanddele, (der alle fire omsættes til huller på strimmelen) vil den samme ordre, når den er anbragt i maskinens lager, altid bestå af tre og kun tre dele. Det sidst tilkomne symbol har kun betydning under indlæsningen af ordren, og reglerne for anvendelsen af etikettemærket er udarbejdet i tilknytning til det indlæseprogram, der, som tidligere omtalt, er anbragt i maskinens lager, inden indlæsningen af den aktuelle kode finder sted. Indlæseprogrammets virkemåde omtales nøjere i afsnit 10.4.

9.2 Etikettemærkerne.

Der findes otte forskellige etikettemærker, nemlig 8, 9, A, B, C, D, E, og F.

Når en ordre skal etikettemærkes, skriver man det pågældende etikettemærke mellem indeksmærket og operationsdelen. En fuldstændig ydre ordre kan da f.eks. se således ud

73B962.

Ofte skal ordrerne ikke etikettemærkes, men blot nedskrives som det er vist i de foregående kapitler.

9.3 Etikettecellerne.

Til hvert etikettemærke svarer der en halvcelle i lageret. Disse "etiketteceller", der ikke adskiller sig fra de øvrige celler i lageret, er halvcellerne 2008, 2009, 2010, 2011, 2012, 2013, 2014 og 2015.

Etikettemærkets funktion.

Er en ordre etikettemærket, vil der, under indlæsningen, ske en addition af indholdet af den til etikettemærket svarende etiketcellehalvcelles adressepositioner til ordrens pseudoadresse modulo 2048. (Da pseudoadresserne står i positionerne 1 til 11 vil en normal addition jo kunne give menteoverføring til position 0, hvad der vil betyde ændret indeksmærkning).

Eksempel. Ordren 1424 B9 63 indlæses, medens C(2009) = 1862 A 00. Under indlæsningen vil ordren da ændres. Pseudoadressen bliver: 1424 + 1862 - 2048 = 1238, hvorefter ordren efter indlæsningen står som 1238 B 63. (Sædvanlig addition ville give følgende resultat:

$$\overbrace{0\ 1424\ 1}^{B}\ 63 \qquad \overbrace{1\ 1238\ 1}^{D}\ 63 \sim 1238\ D\ 63)$$

$$\underbrace{0\ 1862\ 0}_{A}\ 00 =$$

Vi fremhæver endnu engang, at etikettemærket ikke genfindes i den indlæste ordre.

Etikettemærkets funktion er altså følgende:

1) Pseudoadressen ændres, idet indholdet af etikettecellens adressepositioner adderes til pseudoadressen modulo 2048.

2) Indeksmærkningen ændres ikke.

3) Operationsdelen ændres ikke.

9.5 Etikettemærke 8 i forbindelse med biblioteksekvenser

Alle etikettemærkerne fungerer på samme måde, men 8-mærkede ordrer forekommer særlig hyppigt, idet de såkaldte biblioteks-sekvenser benytter etikettemærke 8. Under omtalen af DASK-biblioteket, kapitel 12, behandles disse sekvenser nærmere, men da betydningen af at kunne etikettemærke ordrer træder tydeligt frem netop i forbindelse med anvendelsen af biblioteksekvenserne, vil vi allerede nu omtale det principielle i sagen.

Biblioteksekvenser er færdige koder, der løser hyppigt forekommende problemer. Som eksempel kan nævnes kvadratrodsud-dragning. Skal koderen et sted i sit program beregne en kvadrat-rod, behøver han ikke selv kode denne beregning, men kan i ste-det låne biblioteksekvensen til kvadratrodsuddragning i biblio-teket og benytte denne som en del af sin egen kode. Koderen be-stemmer så hvor i lageret han vil have biblioteksekvensen anbragt. Herved er vi ved det afgørende punkt. Biblioteksekvenserne må jo nødvendigvis være kodet ud fra forudsætningen om en bestemt beliggenhed i lageret, og nu ønsker koderen den placeret et ganske andet sted, hvilket jo kræver, at pseudoadresserne i mange af biblioteksekvensens ordrer skal ændres. Men samtlige ændringer består i en addition af samme konstant i adressedelen. Da alle biblioteksekvenser er kodet i OA8-form (d.v.s. med første ordre placeret i hac. 0 med etikettemærke 8) skal koderen blot sørge for, at etikettecelle 2008 indeholder adressen på den halvcelle, hvor man ønsker den første af biblioteksekvensens ordrer lagret; under indlæsningen vil da samtlige 8-mærkede ordrer få den rigtige pseudoadresse og biblioteksekvensen fungerer som tilsigtet.

Anvender koderen kun en enkelt biblioteksekvens, kan denne naturligvis lagres netop fra halvcelle 0, og resten af koden kan da lagres efter biblioteksekvensen. Herved vil etikettemærkningen være overflødig; men da der normalt anvendes flere bibliotekse-kvenser i samme program, betyder etiketteringen en stor lettelse for koderen.

Som eksempel på denne brug af etikettemærke 8 betragter vi først følgende sekvens, hvis funktion vi beder læseren finde.

```
          0   98 A 48
          1 1948 A 35
6 ---->   2    2 B 35
          3  198 B 44
          4  298 B 0A
          5   98 A 06
2 <----   6    2 A 33   hop til ordren i hac 2, hvis C(IRB)≠ 0
8 <----   7    8 A 10   hop til ordren i hac 8
7 ---->   8 - - - - -
```

Denne sekvens kan kun benyttes, hvis den netop placeres fra halvcelle 0 til halvcelle 7. Skal sekvensen placeres med første ordre i halvcelle 20, kan koden se således ud

```
         20   98 A 48
         21 1948 A 35
26 ---->  22    2 B 35
         23  198 B 44
         24  298 B 0A
         25   98 A 06
22 <----  26   22 A 33   hop til ordren i hac 22 hvis C(IRB)≠ 0
28 <----  27   28 A 10   hop til ordren i hac 28
27 ---->  28 - - - - -
```

Skal sekvensen anvendes med første ordre f.eks. i hac 60, skal koden ændres igen. I stedet koder vi sekvensen på 0A8-form.

```
          0   98 A 48
          1 1948 A 35
6A8 ---->  2    2 B 35
          3  198 B 44
          4  298 B 0A
          5   98 A 06
2A8 <----  6    2 A8 33
8A8 <----  7    8 A8 10
           8 - - - - -
```

Nu indlæses sekvensen, medens C(2008) = 60, med første ordre i hac 60.

De første seks ordrer anbringes så uden ændringer i hac 60 til hac 65,

idet disse ordrer ikke er etikettemærkede.

Derimod ændres de to sidste ordrer, idet C(2008) = 60 adderes til pseudoadresserne. Disse ordrer, der anbringes i hac 66 og hac 67, kommer da til at se således ud

66 62 A 33
67 68 A 10

og sekvensen står nu rigtigt i halvcellerne 60 til 67. Vi bemærker, at 8-mærkede ordrer er sådanne, der henviser til en anden adresse indenfor programmet selv. Vi siger, at 8-mærkningen er anvendt til relativ adressering.

9.6 Styring af indlæseprogrammet. Etiketter.

Når koderen er færdig med at nedskrive programmet skal dette, tilligemed de konstanter, der indgår i regningerne, anbringes på de planlagte steder i lageret. For at dette kan ske, må indlæseprogrammet styres. Da indlæsningen normalt foregår fra hulstrimlen, kan vi på denne anbringe de instrukser, indlæseprogrammet kræver.

Disse instruktioner, der kan være af forskellig art, kaldes etiketter. De er altså ikke sædvanlige ordrer, men har det ene vigtige formål at lede den korrekte indlæsning.

9.7 Ydre programetiketter.

En ydre programetikette består af tre eller fire bestanddele:

adresse
etikettesymbol (altid et E),
etikettemærke (der eventuelt kan udelades)
samt funktionsangivelse (se skemaet på de følgende sider)

Eksempler: 27 E B 0 ; 1424 E 9 42 ; 987 E 19.

Disse programetiketter har altså helt samme struktur som ordrerne, men indeksmærkninger A, B, C eller D er erstattet med et E. (Det er netop dette E, der "fortæller" indlæseprogrammet, at det her drejer sig om en ydre etikette!) Adressen har den sædvanlige decimale form og den sædvanlige betydning. Etikettemærkningen fungerer også helt som ved ordrene. Derimod har funktionstegnene slet intet med de sædvanlige operationer at gøre. De forskellige funktioner omtales på de følgende sider. Da flere af funktionerne har en meget betydningsfuld anvendelse - f.eks. kontrol af om indlæsningen foregår korrekt - er det af stor vigtighed, at koderen virkelig udnytter etikettefunktionerne i fuld udstrækning, og ikke begrænser sig til kun at anvende det mindst mulige antal ydre programetiketter.

9.8 Etikettefunktionerne.

I det følgende skema over etikettefunktionerne anvendes begrebet: løbende adresse, hvorved vi forstår den adresse, hvortil indlæseprogrammet næste gang vil indlæse. Løbende adresse forkortes til lb. adr. Vi anvender iøvrigt betegnelserne E_p eller E_q for indholdet i etikettehalvcellerne svarende til p eller q; p og q kan da antage værdierne 8, 9, A, B, C, D, E eller F (Eks.: E_B = C(2011))

Hvor etikettemærket (q) udelades, kan E_q passende fortolkes lig 0.

	Funktions-betegnelse	Ydre etikette (q kan udelades)	Virkning
	0	n,E,(q),0	Sæt E_8=(lb.adr. + n + adressen i E_q),A,00.
1)	1p	n,E,(q),1p	Sæt E_p=(lb.adr. + n + adressen i E_q),A,00.
2)	2	n,E,(q),2	Gør lb.adr. lige (dvs. adder eventuelt 1 til lb.adr.) Sæt derefter E_8=(lb.adr.+n + adressen i E_q), A,00.
	3	n,E,(q),3	Sæt lb.adr.=n + adressen i E_q, klar til indlæsning til AL.
	40	n,E,(q),40	Hop til programmets ordre i halvcelle (n + adressen i E_q).
	41	n,E,(q),41	**Umiddelbart efter denne etikette skal følge en "kontroletikette" af formen d, A, p, 0; hvor d angiver en decimal adresse. Indlæseprogrammet undersøger da, om det sidst indlæste halvord er indlæst til adressen (d + adressen i E_p). Er dette tilfældet fortsættes som ved 40 ellers udskrives d samt den sidste adresse hvortil der er udlæst, hvorefter maskinen stopper. Ved påfølgende start fortsættes som ved funktion 40.**

Fortsættes på side 9.7

Bemærkninger.

1) Ved de funktioner, hvor både p og q forekommer, må disse gerne antage samme værdi.

2) Hvis lb.adr. er lige, har (n, E, (q), 2) samme virkning som (n, E, (q), 0).

3) Ved adressen i E_q forstås C((2000 + q) adressepositioner).

Etikettefunktionerne (fortsat)

Funktions-betegnelse	Ydre etikette(q kan udelades)	Virkning
42	n,E,(q),42	Umiddelbart efter denne etikette skal følge en kontroletikette bestående af 10 sedecimale cifre. Endvidere bør E_8 være lige. Indlæseprogrammet danner nu summen af indholdet i hec E_8, hec$(E_8 + 2)$,....., indtil den helcelle, der sidst er indlæst til. (Er der sidst indlæst til en vhac optræder summen som om den tilsv. hhac var nulstillet). Er E_8 ulige summeres kun indholdet af højrehalvcellerne. Summen, der dannes modulo 2, sammenlignes nu med "kontroletiketten". Er der overensstemmelse mellem de to tal fortsættes som ved 40, ellers udføres, med 3 A-ordre, sedecimal udskrift af den dannede sum og derefter stop. Ved start fortsættes som ved 40. NB. Se bemærkningen nederst på siden.
43	n,E,(q),43	Umiddelbart efter denne etikette skal følge de to kontroletiketter, der er nævnt under funktionerne 41 og 42. Først udføres funktion 41. Er der indlæst som ønsket, fortsættes med funktion 42, ellers stop. Ved tryk på startknap fortsættes da med funktion 42.
50	n,E,(q),50	Stop, ved tryk på startknap som ved 40.
51 52 53	n,E,(q),51 n,E,(q),52 n,E,(q),53	Samme krav og samme virkning som ved henholdsvis 41, 42 og 43 med den tilføjelse at der, ved rigtig indlæsning, stoppes inden hoppet til ordren i halvcelle (n + adressen i E_q).

Fortsættes på side 9.8

Bemærkning. 42 (43, 52, 53) er tænkt anvendt på den måde, at maskinen ved 1. indlæsning skal danne den omtalte sum, der da - ved følgende indlæsninger - benyttes som kontroletikette.

Indlæseprogrammet er lagret med første ordre i hac 1792. Kontrol af indlæsningen indtil et vilkårligt sted på strimmelen kan da udføres ved, på det pågældende sted på strimmelen at indføre en af følgende etiketter

$$1792\,E \begin{cases} 41 \\ 42 \\ 43 \end{cases}$$

eller

$$1792\,E \begin{cases} 51 \\ 52 \\ 53 \end{cases}$$

Etikettefunktioner (fortsat)

Funktions-betegnelse	Ydre etikette (q kan udelades)	Virkning
6p	n,E,(q),6p	Sæt E_p=(n + adressen i E_q), A, 00.
7	n,E,(q),7	(n + adressen i E_q) opfattes som nummeret på en tromlekanal. 32 helord indlæses til hec 1472 og de flg. 31 helceller, hvorefter de 32 helord overføres til den pågældende tromlekanal. Derefter indlæses de 32 næste helord til samme pladser i lageret, hvorfra de overføres til næste tromlekanal. o.s.v. indtil hopetikette eller etikettefunktion 3. NB. Indlæseprogrammet kan kun bevirke overførsel til tromlen, når de 32 helceller i lageret alle er fyldt. Kommer hop- eller 3 etiketten før dette er sket, overføres hec 1472 - hec 1534 til tromlen før disse etiketter adlydes. NB. Sættes ny 7-etiketter før hec 1472-1534 er fyldt op, overføres disse heo ikke til tromle.

9.9 Programetikette E

Funktionen "indlæsning slut" kan kun udføres gennem ydre programetiketter. Som nævnt bruges

n,E,(q),40 eller n,E,(q),50

eller varianterne heraf til dette formål.

Da indlæsningen af større programmer normalt foregår i flere omgange (dette gælder i særlig grad ved indlæsning af talmaterialer), har man indført en kortere ydre programetikette, der bevirker at indlæsningen stoppes.

Her benyttes kun et tegn, nemlig bogstavet E. Den ydre programetikette E bevirker, at der hoppes tilbage til ordren efter den,

hvorfra der hoppes til indlæseprogrammet analogt tilbagehoppet fra biblioteksekvenser med indekshop .*)

Inden vi gennem en række øvelser og eksempler viser anvendelsen af etikettefunktionerne, skal vi omtale endnu en metode, der kan anvendes til styring af indlæseprogrammet.

Indlæseprogrammet arbejder med løbende adresse i IRB. I stedet for at begynde indlæsningen af en programdel med etiketten

n,E,(q),3

kan koderen da, før der hoppes fra det allerede indlæste hovedprogram, ved den sædvanlige ordre (n + adressen i E_q) A 35 indsætte den løbende adresse i IRB, hvorefter indlæseprogrammet vil fungere som planlagt.

9.10 Øvelser og eksempler.

Eksempel 9.1 Sammenhængende program, der ikke anvender etiketmærker. Programmet er kodet til lagring med den første ordre i hac 0. Efter indlæsningen skal der hoppes til ordren i hac 0. (Dette tilfælde dækker omtrent de tidligere øvelser og eksempler.)

Det færdige manuskript

```
0 E 3    Sæt lb.adr.=0 (første ordre læses til hac 0)
Program
0 E 40   Hop til 0
```

Vi fremhæver, at koden virkelig er færdig; samtlige de nedskrevne tal og bogstaver skal nu hulles på strimmelen og alt er parat til indlæsningen og den efterfølgende beregning.

*) Hop til indlæsningsprogrammet kodes med ordren 1987 A 16.

NB. Ved indkørsel af nye programmer anbefales anvendt som strimmelslutetikette de to etiketter

```
1792 E 50
n    E 40
```

Øvelse 9.2 Sammenhængende program, der ikke anvender etikettemærker. Programmet, der er kodet med første ordre i hac 200, skal indlæses. Efter indlæsningen skal der hoppes til ordren i hac 227.

Eksempel 9.3 Sammenhængende program, der anvender 8-mærkningen, til relativ adressering (side 9.4). Programmet er kodet i 0A8-form og skal indlæses med første ordre (nr. 0) i celle 300. Efter indlæsningen skal der hoppes til programmets ordre nr. 20. Den færdige kode:

```
300 E 3 sæt lb.adr.=300 (læs første ordre til 300)
  0 E 0 sæt E8=(lb.adr.; A, 00) dvs.(300, A, 00)
Program
 20 E8 40  hop til ordren i hec (20 + adressedel af E8)
                                                =320
```

Øvelse 9.4 Samme opgave som i eks. 9.3 idet ordre nr. 0 skal anbringes i hac 150. Efter indlæsningen skal der hoppes til programmets ordre nr. 7.

Øvelse 9.5 Samme opgave som i 9.3, idet alle 8-mærkede ordrer i stedet er E-mærkede.

Øvelse 9.6 Programmet består af to dele, der begge anvender 8-mærkning til relativ adressering, d.v.s. begge er kodet i 0A8-form. De ordrer i programdel I, der henviser til programdel II, er A-mærkede. De ordrer i programdel II, der henviser til programdel I, er 9-mærkede.

Programdel I skal indlæses med ordre nr. 0 i hac 100.

Programdel II skal indlæses med ordre nr. 0 i hac 200.

Efter indlæsningen skal der hoppes til ordre nr. 4 i programdel I.

Den færdige kode:

```
100 E 69  sæt E9 = (100, A, 00)
200 E 6A  sæt EA = (200, A, 00)
  0 E9 3  sæt lb.adr. = 0 + 100 = 100
  0 E  0  sæt E8 = (lb.adr., A, 00) = (100, A, 00)
Programdel I
  0 EA 3  sæt lb.adr. = 0 + 200 = 200
  0 E  0  sæt E8 = lb.adr., A, 00
Programdel II
  4 E9 40 hop til ordren i hac (4 + 100) = hac 104.
```

Øvelse 9.7 Et program består af tre dele I, II og III, der alle er kodet i 0A8-form. Hver del indeholder ordrer, der henviser til de to øvrige programdele.
Ordrer, der henviser til del I, er 9-mærkede,
ordrer, der henviser til del II, er A-mærkede og
ordrer, der henviser til del III, er B-mærkede.
Programmet skal indlæses med del I begyndende i hac 97, del II begyndende i hac 144 og del III begyndende i hac 207. Efter indlæsningen skal der hoppes til ordre nr. 2 i del III.

Eksempel 9.8

Programmet består af en hovedsekvens HS samt to biblioteksekvenser BSI og BSII. Alle programdele er kodet i 0A8-form. Ordrer i HS, der henviser til BSI er 9-mærkede og ordrer i HS, der henviser til BSII er A-mærkede. Biblioteksekvenserne BSI og BSII bruger ingen parametre (mange biblioteksekvenser kræver visse instruktioner for at kunne arbejde; disse kaldes parametre. Dette forhold beskrives nærmere i eks. 9.10). De tre sekvenser anbringes tæt efter hinanden i rækkefølgen BSI,BSII,HS. BSI skal have ordre nr. 0 i hac 150. Efter indlæsning hoppes til ordre nr. 17 i HS.
Den færdige kode:

```
150 E  69   sæt E9 = 150, A, 00
  0 E9  3   sæt lb.adr. = 150
  0 E   0   sæt E8 = 150, A, 00
 [BSI]
  0 E  1A   sæt EA = (ny lb.adr., A, 00)
  0 E   0   sæt E8 = (ny lb.adr., A, 00)
 [BSII]
  0 E   0   sæt E8 = (ny lb.adr., A, 00)
 [HS]
 17 E8 40   hop til ordren i hac (17 + adressen i Eq).
```

Vi fremhæver, at såvel nedskrivningen af HS som etiketteringen kan foregå, uden at placeringen af de enkelte programdele er kendt.

Øvelse 9.9 Samme opgave som i 9.8 med den tilføjelse, at der ydereligere anvendes en tredie biblioteksekvens BSIII. Ordrer i HS der henviser til BSIII mærkes med B.
Sekvenserne anbringes tæt efter hinanden i rækkefølgen BSIII, BSII, BSI, HS. BSIII skal have ordre nr. 0 i hac 28.
Efter indlæsning hoppes til ordre nr. 0 i HS.
Hvorfor skal HS anbringes til sidst !
Hvordan skal der etiketteres, hvis biblioteksekvenserne have ordre nr. 0 lagret i en vhac ! (Herved må kravet om, at sekvenserne skal ligge helt tæt efter hinanden, evt. opgives).

Eksempel 9.10

Programmet består af en hovedsekvens HS samt to biblioteks-sekvenser BSI og BSII. Alle programdele er kodet i 0A8-form
BSI anvender to parametre, d.v.s. brugen af koden forudsætter, at den første parameter er anbragt i etikettecelle 2009 og den anden i etikettecelle 2010. Under indlæsningen anbringes de på de rette steder i BSI. BSII anvender en parameter. Den skal være anbragt i etikettecelle 2009.
Yderligere forudsætter vi, at BSII skal anbringes med ordre nr. 0 i lige adresse.
Da 9- og A-mærket nu skal bruges til parametrene, bruger vi følgende etikettemærker i HS:
Ordrer, der henviser til BSI, mærkes med B og
ordrer, der henviser til BSII, mærkes med C.
Sekvenserne anbringes så tæt som muligt, med BSI begyndende i 50.
Efter indlæsningen hoppes til ordre 40 i HS.

Det færdige kodemanuskript:

```
2009 E   3   sæt lb.adr. = 2009, d.v.s.:  læs til 2009
- - - - - - første parameter for BSI (der læses til 2009)
- - - - - - anden parameter for BSII (der læses til 2010)
  50 A  00  der læses til 2011, d.v.s. E_B = (50, A, 00)
   0 EB  3  sæt lb.adr. = 50
   0 E   0  sæt E_8 = lb.adr.
  [BSI]
   0 E   2  gør ny lb.adr. lige og sæt E_8 lig denne.
2009 E   3  sæt lb.adr. = 2009, d.v.s. læs til 2009.
- - - - - - parametrene for BSII (der læses til 2009)
   0 E8  3  sæt lb.adr. = E_8, d.v.s. genindsæt korrekt lb.adr.
   0 E  1C  sæt E_C = lb.adr.
  [BSII]
   0 E   0  sæt E_8 = ny lb.adr.
  [HS]
  40 E8 40  hop til ordre nr. 40 i HS.
```

I stedet for 50 A 00 og de to følgende etiketter kan man her skrive

50 E 3; 0 E 1B; 0 E 0 eller

50 E 6B; 0 EB 3; 0 E 0.

Angående brugen af de specielle etikettefunktioner henvises til eksemplerne i kapitel 12. Etikettefunktion 7, der vedrører indlæsningen af tal til tromlen, anvendes i et eksempel i afsnit 10.4.

KAPITEL 10.

Kodning af tal.

Indledning.

10.1 Indlæseprogrammet, der er omtalt flere gange i de foregående kapitler, er også i stand til at indlæse tal i DASK. Ved anvendelse af indlæseoperationerne 19 og 39 kan koderen naturligvis få tal, der er hullet på strimmelen, overført til maskinens lager, men her møder vi den samme vanskelighed, som vi mødte ved indlæsningen af ordrerne: vi arbejder i decimalsystemet - maskinen i det binære talsystem. Også på dette sted holder vi fast ved det tilvante system, d.v.s. vi skriver (og stanser) vore tal i decimalsystemet, hvorefter maskinen - styret af indlæseprogrammet - transformerer tallene til det binære talsystem. Hvorledes dette sker, er i og for sig af mindre betydning for koderen, der blot skal kende konventionerne for den måde tallene skal nedskrives på - egentlig stanses på - for at indlæseprogrammet kan "opfatte" tallene rigtigt. Konventionerne for kodning af tal omtales derfor umiddelbart herefter. Til brug for de læsere, der måtte ønske et indblik i indlæseprogrammets virkemåde, afslutter vi dette kapitel med en kortfattet gennemgang af de forhold, der gør det muligt for indlæseprogrammet at skelne mellem ordrer, etiketter og tal.

Vedrørende nøjagtighed under indlæsningen:
De kodede cifre indlæses alle og talværdien lagres i omregnet binær form:

1) heltal eksakt. Talværdien skal være mindre end henholdsvis 2^{19} og 2^{39}.

2) DASK-tal til halvcelle korrekt afrundet til 19 binaler

3) DASK-tal til helcelle eller flydende tal på oppakket form med en relativ omregningsfejl på indtil $7 \cdot 2^{-39}$ eller $1{,}3 \cdot 10^{-11}$ med påfølgende afrunding til 39 binaler.

4) Flydende tal på pakket form med taldelen korrekt afrundet til 27 binaler.

10.2 Konventioner for kodning af tal.

Som hovedregel gælder, at koderen, ud over cifrene i det tal, der skal kodes, skal nedskrive tre bogstaver.

1) Et foran tallet. Dette, der angiver fortegnet, skal være C for positive tal, og D for negative tal.

2) Et på kommaets plads. Dette skal være et af bogstaverne A,B,C,D eller E. Betydningen af de forskellige "kommaer" er vist i tabellen længere nede på siden.

3) Et, der afslutter tallet. Altid et A.

Vi fremhæver, at koderen, ved at glemme et af de tre bogstaver, "narrer" indlæseprogrammet, hvilket medfører at tallet opfattes forkert.

Hoved-regel	Komma	Virkning
	A	Tallet opfattes som heltal og lagres i en helcelle*) med enhed i position 39.
	B	Tallet opfattes som DASK-tal og lagres i en halvcelle.
	C	Tallet opfattes som DASK-tal og lagres i en helcelle.*)
	D	Tallet opfattes som et flydende tal på pakket form og lagres i en helcelle.*)
	E	Tallet opfattes som et flydende tal på oppakket form og lagres som sådant i en helcelle + den følgende venstre halvcelle.*)
	(F)	(Tallet opfattes som heltal og lagres i en halvcelle med enhed i position 19(39))

*) Indlæsning til helcelle må, gennem omhyggelig etikettering, dirigeres af koderen.

Bemærkning: Når komma A (eller F) benyttes (heltal), udelades det afsluttende A.

Ud over de fem kommavarianter har koderen endnu nogle muligheder for at få læst tal ind i maskinen ved hjælp af indlæseprogrammet.

Vi nævner først to konventioner, der kan benyttes i i stedet for de lige nævnte. De er medtaget, fordi anvendelsen af disse specielle konventioner sparer maskinen for en del beregninger under indlæsningen.

Speciel regel

0 til halvcelle kodes ved et A

Speciel regel

Flydende tal kan altid kodes som nævnt under "komma" D og E, men der er mulighed for at udvide de der viste konventioner, hvad der specielt benyttes ved flydende tal med få betydende cifre og en numerisk stor 10-talseksponent.
Taldelen kodes som vist under D og E. Umiddelbart efter sidste betydende ciffer skrives da:

1) C i tilfælde af positiv 10-talseksponent. *)
   D i tilfælde af negativ 10-talseksponent.
2) Selve 10-talseksponenten.

Talkoden afsluttes også her med et A. (Ikke A mellem taldel og eksponent.)

Endelig er der en speciel variant, der bevirker indlæsning af fem sedecimale cifre til en halvcelle, uden transformation af nogen art :

Sedecimal

Fem sedecimale cifre til en halvcelle kodes med et B efterfulgt af de fem sedecimale cifre.

*) Indlæsning til helcelle må, gennem omhyggelig etikettering, dirigeres af koderen.

Indlæsning med fast skalafaktor.

Ønsker man at indlæse sine tal med en skalafaktor $10^{p_1} \cdot 2^{p_2}$ (jvf. kap. 8) kan indlæseprogrammet trimmes hertil ved lagring af $1024 + p_2$ i adressepositionerne i hac 1844 og $1024 + p_1$ i adressepositionerne i hac 1846.

Dette kan ske ved

ordrerne		eller etikettering	
	$(1024 + p_2)$ A 75		1844 E 3
	1844 A 74		$(1024 + p_2)$ A 00
	$(1024 + p_1)$ A 75		1846 E 3
	1846 A 74		$(1024 + p_1)$ A 00

10.3 Øvelser og eksempler på kodning af tal.

Eksempel 10.1 Tallene i nedenstående tabel skal opfattes som heltal og lagres i helceller begyndende med helcelle 100. Stop efter indlæsningen.

```
                        100 E  3   Etikette: løbende adresse = 100
 1219   kodes   C1219A
- 341     "     D341A
  702     "     C702A    Da der skal 0 til helcellerne, må A
- 91      "     D 91A    gentages. (At de to Aer står efter
    0     "     AA       hinanden her, spiller ingen rolle.
  842     "     C842A    På strimmelen vil samtlige tal jo
                n E 50   have hver sin række - og følge u-
                         middelbart efter hinanden.)
```

Eksempel 10.2 DASKtallene i nedenstående tabel skal lagres i halvceller begyndende med halvcelle 1239.

```
                        1239 E  3   Etikette: løbende adresse = 1239
 0,3413798  kodes   CB3413798A | bemærk at 0 foran kommaet
-0,1700039    "     DB1700039A | ikke skal nedskrives.
 0,9999999    "     CB9999999A |
 0            "     A
-1            "     D1BA
                     n E  50
```

Øvelse 10.3 Tallene i eksempel 10.2 skal lagres i helceller begyndende med hec 1420.

Eksempel 10.4 Tallene i nedenstående tabel skal lagres - som flydende tal på pakket form - i helcellerne 832 til 838.

```
                        832 E  3   Etikette
 8491,3765   kodes   C8491D3765A
-0,0000731674  "     DD0000731674A
 0              "     CD0A      N.B. AA giver samme virkning
-13698742       "     D13698742DA
                      n  E  50
```

Eksempel 10.5 Tallene i nedenstående tabel skal lagres som flydende tal på oppakket form. Første lagringscelle: hec 800

800 E 3 Etikette

-71749,89	D71749E89AA
-3448196	D3448196EAA
0,000068427	CE00068427AA
0	CE0AA(eller AAAA)
	n E 50

Bemærk gentagelsen af A. Det sidste A bevirker, at højre halvcelle nulstilles. Udelades det sidste A, vil det næste flydende tal blive lagret ufuldstændigt.

Øvelse 10.6 Tallene i eksempel 10.4 skal lagres på oppakket form. Taldelen af første tal skal lagres i hec 1208.

Eksempel 10.7 Tallene

$-0{,}123456 \cdot 10^{-26}$ og

$0{,}378145 \cdot 10^{274}$ skal opfattes som flydende tal og lagres på pakket form i hec 22 og hec 24.

Talkoden bliver da:

```
22 E  3 etikette
DD123456D26A
CD378145C274A
 n E  50
```

Vi bemærker, at følgende talkode (der svarer til en simpel omskrivning af tallene) giver samme, rigtige indlæsning. Taldelen behøver således ikke være normaliseret.

```
22 E  3
D1D23456D27A
C37D8145C272A
 n E  50
```

Øvelse 10.8 Tallene i eksempel 10.4 omskrives til ægte decimalbrøker gange en 10 potens, hvorefter tallene kodes som i eks. 10.7.

Eksempel 10.9 De følgende tal skal kodes som flydende tal på oppakket form og lagres i hec 768 og fremefter.

$0{,}891746 \cdot 10^{-17}$

$-0{,}19872638 \cdot 10^{64}$

Talkoden bliver da

```
 768 E  3 etikette
CE891746D17AA  (det sidste A bevirker, at hhac 771 nulstilles)
DE19872638C64AA
   n E  50
```

Taldelen behøver ikke være normaliseret. Hvordan skal talkoden se ud, hvis tallene skrives $891{,}746 \cdot 10^{-20}$ og $-1{,}9872638 \cdot 10^{63}$?

Øvelse 10.10 Tallene i eksempel 10.5 omskrives til ægte decimalbrøker gange en 10-potens, hvorefter tallene kodes som i eks. 10.9.

Eksempel 10.11 Tallene 0,1,2,...,63 skal lagres (som heltal med enhed i pos.39) på tromlekanalerne 18 og 20 (disse følger umiddelbart efter hinanden)

```
 18 E   7   Etikette (se side 9.8)
 AA         0
 C1A        1·2^-39
 C2A        2·2^-39
  .
  .
  .
  .
 C63A       63·2^-39
 n E  50
```

Vi bemærker, at denne direkte indlæsning til tromlen finder sted med grupper af 32 helord.

Øvelse 10.12 DASKtallene 0,001; 0,002; .... 0,128 skal indlæses til tromlekanalerne 12, 14, 16 og 18.
Hvorledes kan talkoden se ud? Hvis tallene skal lagres i halvceller, kan de være på tromlekanalerne 12 og 14.
Hvordan skal talkoden så være?

Eksempel 10.13 Indlæsningen af fem sedecimale cifre til halvcelle anvendes ofte i forbindelse med de biblioteksekvenser, der benytter en eller flere parametre. Disse parametre består nemlig hyppigt af et eller flere sedecimale cifre.
Parametrene F; 7; C; 9; 8 og
0; 0; D; 2; E skal indlæses til etikettehalvcellerne 2009 og 2010 uden transformation (jævnfør eksempel 9.8 side 9.11)
Talkode

```
2009 E   3     Etikette.
BF7C98
B00D2E
   n E  50
```

Hvordan virker talkoden, hvis det første tal afsluttes med et A?
Hvordan virker talkoden, hvis koderen glemmer 8 i det første tal?

Øvelse 10.14 Hvordan kunne ovenstående parametre transformeres inden indlæsningen, og hvordan kunne talkoden se ud, hvis man var tvunget til at anvende en af de decimale talfremstillinger (f. eks. DASK-tal til halvcelle)?

10.4 Indlæseprogrammet.

Vi har valgt at afslutte dette kapitel, der indeholder de sidste oplysninger om den form, på hvilken koderen skal aflevere sit materiale, med en kort omtale af indlæseprogrammets virkemåde.

Indlæseprogrammet står som tidligere nævnt i DASK, når indlæsningen af det aktuelle program begynder. Det aktuelle program må være omsat til huller i strimmelen (jævnfør hulkoden side 6.3). Strimmelen indføres derefter i strimmellæseren, og indlæsningen kan begynde.

Strimmellæseren foretager den første udvælgelse af symbolerne på hulstrimmelen, idet kun de symboler, der har et hul på 5. plads, indlæses. I det følgende beskæftiger vi os kun med disse.

Indlæseprogrammet optager symbolerne fra strimmelen i grupper.

Allerede det første tegn, der indlæses fra en gruppe, (0,1,2....9,A,B,C,D,E eller F), giver indlæseprogrammet information.

Der er følgende seks forskellige muligheder:

I Det første tegn er et decimalt tal 0,1,2,.....,9.
II Det første tegn er et C eller et D.
III Det første tegn er et B
IV Det første tegn er et E
V Det første tegn er et A
VI Det første tegn er et F

I | Første tegn er 0,1,2,3,4,5,6,7,8 eller 9 |

I dette tilfælde angiver det indlæste tal sammen med de følgende tal en decimal adresse. Indlæsningen fortsætter da, indtil en af følgende muligheder indtræffer:

Ia Efter de decimale tal følger A, B, C eller D.
Ib Efter de decimale tal følger E.
(Ic Efter de decimale tal følger F.)

Ia I dette tilfælde er det en ordre, der er ved at blive indlæst. Der sker da følgende:

Den decimale adresse omsættes til det binære talsystem og anbringes modulo 2048 i positionerne 1 til 11 i den halvcelle, der angives ved den løbende adresse (= C(IRB)).

Er	indeksmærket	A,	indsættes	0,0	i	positionerne	0	og	12
"	"	B,	"	0,1	"	"	"	"	"
"	"	C,	"	1,0	"	"	"	"	"
"	"	D,	"	1,1	"	"	"	"	"

Derefter indlæses næste symbol fra strimmelen; er dette 8,9,A,B,C,D,E eller F er ordren etikettemærket og behandles i overensstemmelse med konventionerne (kapitel 9). De to følgende symboler, af hvilke det første skal være 0,1,2,3,4,5,6 eller 7, opfattes som operationsdelen og anbringes i pos. 13 - 19. Er ordren ikke etikettemærket, skal det tegn, der indlæses efter indeksmærket, være $\leq$ 7. Dette tegn og det følgende angiver da operationsdelen.

Når de to operationscifre er anbragt, er hele ordren indlæst, og indlæseprogrammet begynder forfra med næste hulstrimmelsymbol.

Ib I dette tilfælde er det en ydre etikette , der er ved at blive indlæst. Der sker da følgende:

Den decimale adresse omsættes til det binære talsystem, og gemmes (modulo 2048), indtil de følgende cifre (det følgende ciffer), der angiver etikettefunktionen, er indlæst.

Er det ciffer, der følger efter E, større end eller lig 8, er den ydre etikette etikettemærket og behandles i overensstemmelse med konventionerne (kapitel 9).

Efter indlæsningen af de cifre (det ciffer), der specificerer etikettefunktionen, udføres denne. Indlæseprogrammet begynder så forfra med næste hulstrimmelsymbol.

(I_c Benyttes ikke. Iøvrigt samme virkning som E.)

II Første tegn er C eller D

I dette tilfælde er det et decimalt tal, der indlæses. C angiver, at tallet er positivt, D, at tallet er negativt.

Der sker da følgende:

De tal, der følger efter C (eller D), indlæses og gemmes, indtil "kommaet" A, B, C, D eller E er indlæst. Når dette er sket, indlæses de resterende cifre i tallet, som derefter behandles og lagres i overensstemmelse med konventionerne (side 10.2 og 10.3). Når der indlæses et A, er indlæsningen af det pågældende tal forbi, og indlæseprogrammet begynder forfra med næste hulstrimmelsymbol.

III Første tegn er B

I dette tilfælde er det en parameter bestående af fem sedecimale cifre, der indlæses. De fem følgende tegn indlæses og lagres i den halvcelle, der angives ved løbende adresse. Derefter begynder indlæseprogrammet forfra med næste hulstrimmelsymbol.

IV Første tegn er E

I dette tilfælde er indlæsningen forbi. Der sker et hop til den halvcelle, hvis adresse er 1 + C(IRD), altså til ordren umiddelbart efter den, hvormed indlæsningen startedes.

V Første tegn er A

I dette tilfælde nulstilles den halvcelle, der skulle indlæses til, og indlæseprogrammet begynder forfra med næste hulstrimmelsymbol.

VI Første tegn er F

Tegnet negligeres og fornyet indlæsning påbegyndes.

Hermed har vi omtalt de forskellige muligheder. Vi bemærker, at kombinationer af tal, der ikke falder ind under et af de nævnte tilfælde, vil medføre stop i indlæsningen.

På den følgende side 10.11 bringer vi en skematisk oversigt over samtlige ydre koder, ydre etiketter og samtlige talkoder.

Kodning af ordrer, etiketter og tal.

Kategori				
Ordre	adresse	indeksmærke	etikettemærke	operationscifre
	n+1 cifre $d_n \ldots d_0$	A B C D	$(s \geq 8)$	ss
Etikette	adresse	etikettesymbol	etikettemærke	etikettefunktionscifre
	n+1 cifre $d_n \ldots d_0$	E	$(s \geq 8)$	0 1 (skal efterfølges af $s \geq 8$) 2 3 4 (skal efterfølges af 0,1,2 eller 3) 5 (skal efterfølges af 0,1,2 eller 3) 6 (skal efterfølges af $s \geq 8$) 7
		E		
Nul til halvcelle	A			
Sedecimal indlæsning af hao	B ssssss			

Tal	For-tegn	heltals-cifre	komma	decimaler	evt.skalafaktors 10-potenseksponent fortegn	evt.skalafaktors 10-potenseksponent heltalscifre	tal slut
Heltal.2^{-39}	C ~ + D ~ −	n+1 cifre $d_n \ldots d_0$	A				
DASK-tal, hao			B	m cifre $d_1 \ldots d_m$			A
DASK-tal, heo			C				
Flydende tal, pakket		n+1 cifre $d_n \ldots d_0$	D		C ~ + D ~ −	p+1 cifre $d_p \ldots d_0$	
Flydende tal, oppak.			E				
Heltal·2^{-19}		n+1 cifre $d_n \ldots d_0$	F				

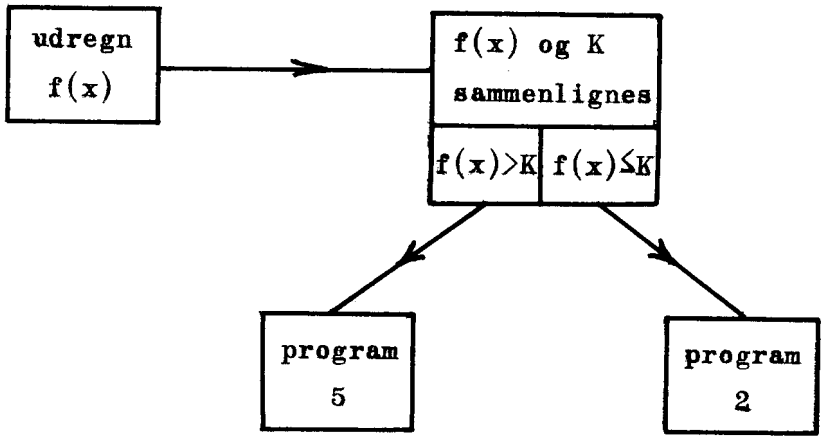
d = decimale cifre s = sedecimale cifre

KAPITEL 11.

1.1 Rutediagram.

Et rutediagram er en kort og anskuelig fremstilling af en kode. Princippet, der vil være velkendt fra andre områder, f.eks. teknik og industri, er dette, at man uden at vise hver enkelt detalje giver en visuelt præget oversigt af "fabrikationsgangen" fra udgangsmateriale til det færdige resultat. Herunder sammenfattes flere eller færre ordrer til en kort tekst eller blot et symbol, som indrammes, og fra denne indramning fører en linie med pil til en lignende beskrivelse af det næste, der skal ske.

Ideen, der vil forstås umiddelbart, kan vi kort illustrere således:



Nu er der mange grader af hvor detaljeret, man ønsker sit rutediagram. Det, der lige er vist, er meget lidt detaljeret; f.eks. kunne man måske ønske at få angivet, hvordan f(x) og K sammenlignes, eller om x kommer fra en celle eller kun findes i AR, etc. Det afgørende for den retningslinie, man i det enkelte tilfælde vil anlægge, er, hvilken funktion rutediagrammet skal have. Ofte vil man finde, at der er tale om en af følgende tre situationer:

a Rutediagrammet skal være et første udkast af koden, muligvis skitseret af en anden end den egentlige koder.

b Koden er under udarbejdelse, og man laver sig detaljerede rutediagrammer for de enkelte programdele.

c Koden er færdig og gennemprøvet, men der skal til senere brug leveres en beskrivelse, som gør den lettilgængelig eventuelt for andre end koderen.

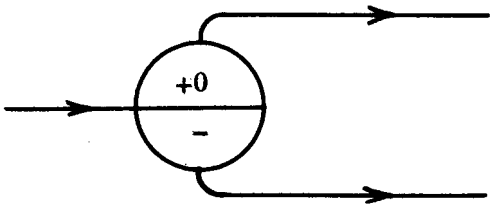
Det vil naturligvis være for teoretisk at mene, at disse tre - og eventuelt flere - former holdes adskilt i koderens kladder.

Imidlertid er det end ikke ved de rene typer let at give faste retningslinier for, hvordan man skal udforme sit rutediagram. Personlige vaner og måder at kode på spiller ind. På den anden side er det vigtigt, at man indenfor en gruppe kodere, der arbejder sammen, holder sig til de samme symboler og konventioner, da rutediagrammerne ellers kan blive ligeså tungt læselige som en egentlig kode.

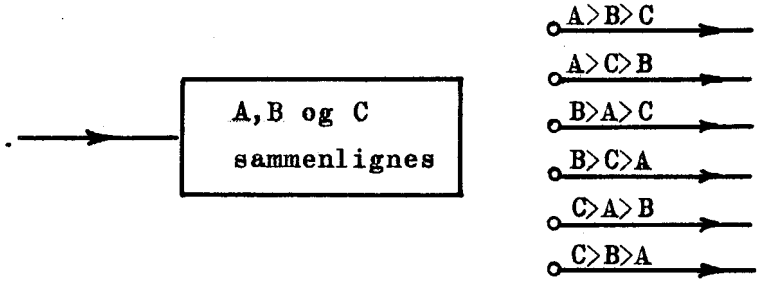
Før vi ved et eksempel viser, hvordan arbejdet med et rutediagram kan forløbe, skal vi pege på nogle af de punkter i et program, man bør trække frem (i hvert fald i de mere detaljerede diagrammer) og sørge for, kommer til at stå præcist og visuelt klart.

1. Den første start, d.v.s. starten i det urørte program, contra start ved gentagelser.
2. Ordrer eller ordregrupper, hvorfra der udgår flere veje (betingede hop eller adressemodificerede hop).
3. Ordrer, hvori der knyttes forbindelse til eller fra andre programdele.
4. Ordrer, hvor der knyttes forbindelse til eller fra maskinens ydre enheder. (Dette er ofte vigtige steder, som det vil være bekvemt hurtigt at kunne lokalisere).

Til 2 : man vil ofte have en særlig figur til at angive et betinget hop, f.eks.:



og i de mindre detaljerede rutediagrammer kan man samle en programdel, der benytter flere sådanne betingede hop, f.eks. således:



Til 3 : i denne forbindelse kan der blive tale om en opdeling, som måske ikke er særlig motiveret i selve koden. Her tænkes på dette, at lader man hele programmet fremstille i et stort rutediagram, vil man ofte få besvær med blot at trække de mange linier samtidig med at overskueligheden kan gå tabt.

11.2 Eks. Fra et styreprogram, HS, hoppes til stadighed til et delprogram, som skal kunne udføre følgende:

Første gang, der hoppes til det, udføres proces A.

De næste p gange udføres proces B.

Derefter igen A og de følgende p gange B.

o.s.v.

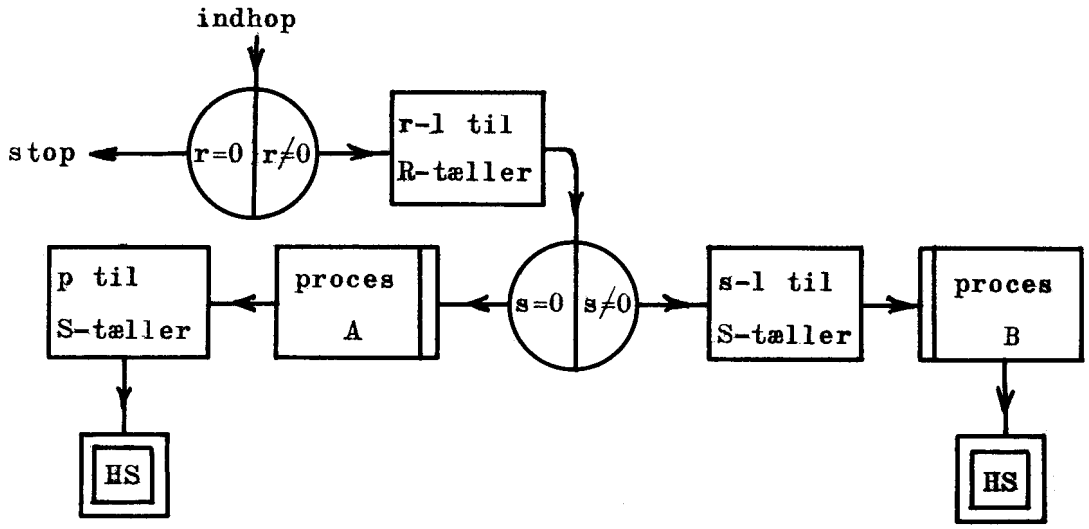
Ialt skal A udføres q gange og altid efterfulgt af p gange B. Man ser, at delprogrammet sammenlagt skal gennemløbes $q + q \cdot p$ gange. Vi tænker os derfor at etablere en R-tæller, der skal tælle $q + q \cdot p$ gange og en S-tæller, der tæller p gange, og kan formulere hovedtrækkene i programmets virkemåde således:

hvis $r \neq 0$, men $s = 0$ så A

" $r \neq 0$ og $s \neq 0$ så B

" $r = 0$ stop

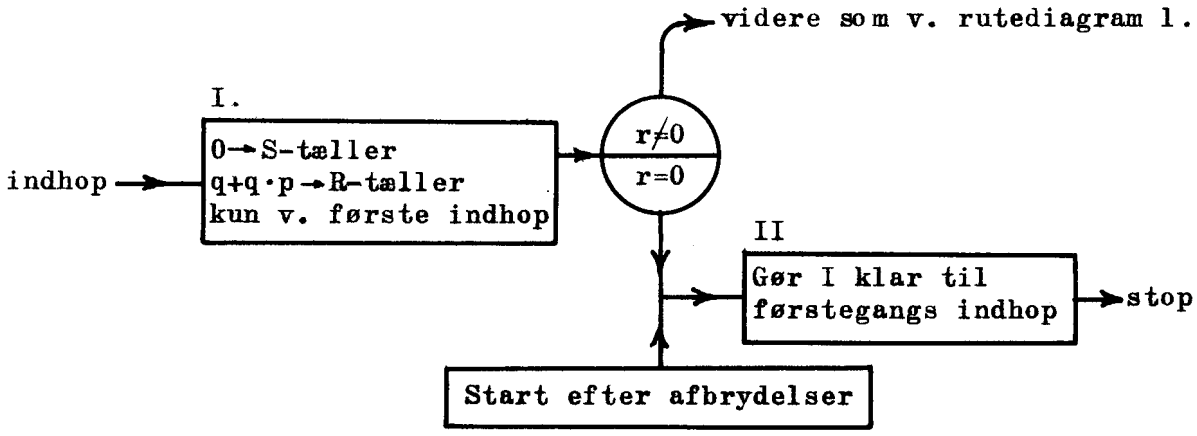
Rutediagram 1. Udkast.



Man ser, at etableringen af tællerne ikke er med, d.v.s. man må i forvejen have sørget for, at R-tælleren indeholder r og S-tælleren 0. Dette kan ske under indlæsningen eller med en indledende programstump.

Et sådant krav til at noget i forvejen skal være bragt i orden, vil altid betyde en indskrænkning af mulighederne for at lade programmet gentage (uden ny indlæsning). Det må imidlertid anbefales i de fleste tilfælde at indrette sig således, at de enkelte delprogrammer selv skaber alle nødvendige betingelser for, at de kan fungere.
Vi ændrer derfor udkastet til :

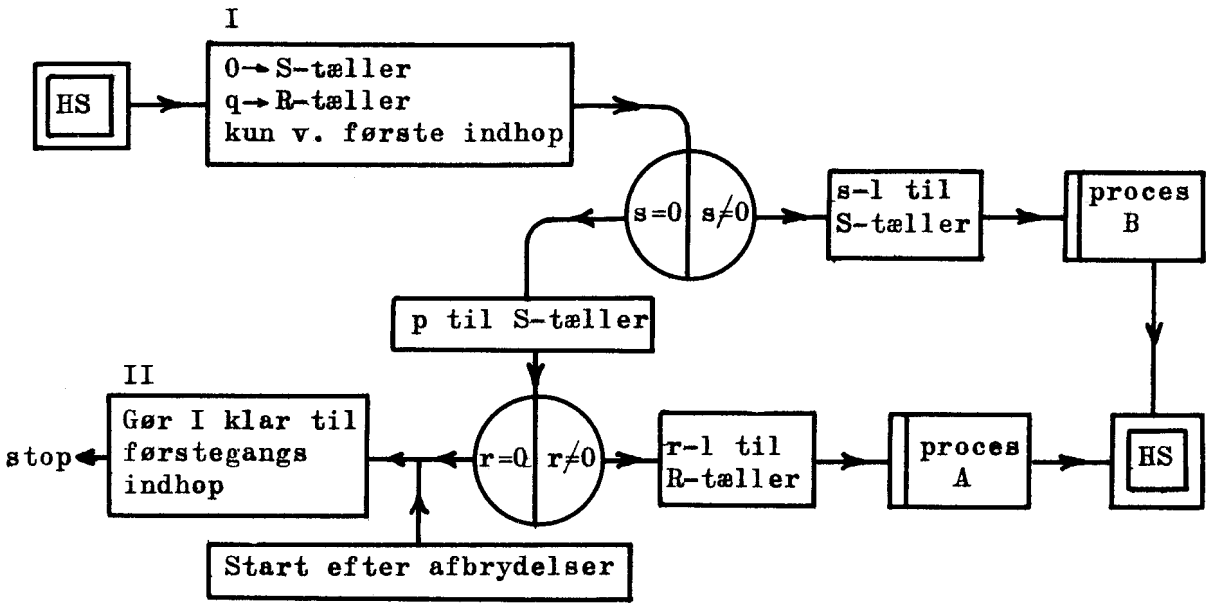
Rutediagram 2.



Man bemærker, hvorledes det først er indramning II, der sikrer, at programmet kan gentages. D.v.s. stopper man før regningerne er afsluttet og ønsker at starte pånyt, skal det ske ved indhop til der, hvor der står "start efter afbrydelse"; efter det STOP man herved lander i, kan der så startes i HS.

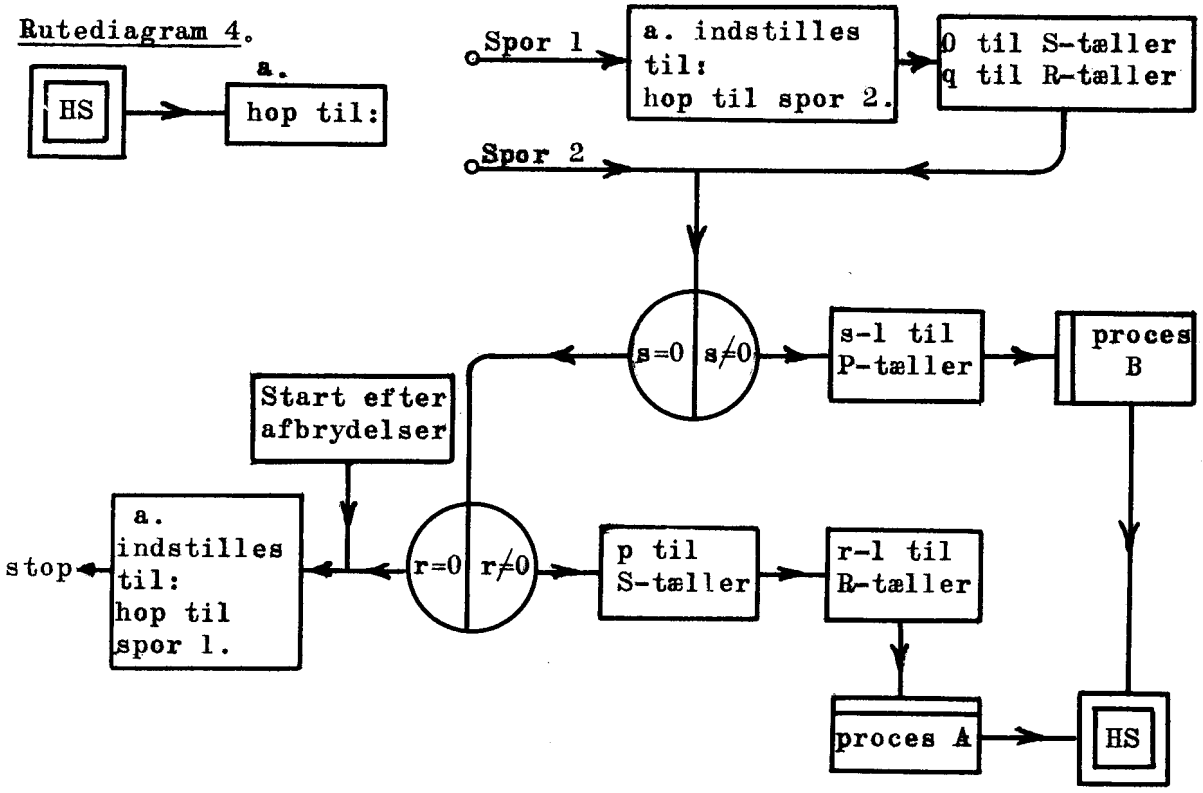
Man kunne nu udfra dette ret grove rutediagram gå i gang med detaljerne. Imidlertid vil vi efter at have studeret det få den ide, at der må kunne tjenes noget ved ikke at lade R-tælleren gå i funktion alle q + q·p gange, men kun netop q gange. Det giver følgende diagram:

Rutediagram 3.



Som afslutning vil vi vise et diagram, hvor ordreforløbet i indramning I og II er nøjere specificeret. Herved når vi til et rutediagram, der nærmer sig den type, der sigtes til under pkt. c i afsnit 11.1. I virkeligheden mangler der kun at angive af hvilken art, talleværkerne er, og hvordan proces A og B er placeret (f.eks. om der hoppes til dem).

Rutediagram 4.



Obs: strimmelen hulles selvfølgelig således, at det indlæste program i a. har "hop til Spor 1".

Det valgte eksempel har behandlet en ting, der vil være velkendt og indgå i mange koder. Meningen har været at vise, hvordan man alene v.h.a. rutediagrammer og uden endnu at have nedskrevet én ordre kan komme på sporet af væsentlige ting (som her indhoppet efter afbrydelser) og eventuelt opdage forbedringer (omlægningen af R-tællingen), samtidig med at man får en fast og klar plan, hvorefter de enkelte ordrer tilsidst kan nedskrives. Gør man disse ting samtidig, altså planlægger og koder på en gang, vil det tage mere tid og ofte føre til dårligere resultater.

KAPITEL 12.

DASK-biblioteket.

12.1. Indledning.

Dask-biblioteket indeholder sekvenser og programmer, som løser hyppigt forekommende opgaver.

Biblioteket vil for det første lette brugen af DASK, idet man undgår dels det pågældende kodearbejde, dels den dertil hørende fejlfinding. For det andet vil det medvirke til en effektiv udnyttelse af DASK, idet bibliotekssekvenserne og -programmerne som regel vil være nær det optimale med hensyn til køretid og kodelængde. For det tredie vil biblioteket gøre et større kodemateriale tilgængeligt for studium.

12.2. Sekvenser og programmer.

En bibliotekssekvens løser en underopgave. Sekvenserne er derfor opbygget på en sådan måde, at de let kan passes ind i de forskellige programmer, hvor der bliver brug for dem. De kan aldrig bruges alene, men vil altid virke som undersekvenser i et program.

Et biblioteksprogram løser en afsluttet opgave. Programmerne kræver for at virke kun indlæsning af en datastrimmel med indgangsværdier, parametre m.m. samt eventuelt visse manuelle operationer ved kontrolbordet.

Det er klart, at der findes opgaver, der både kan optræde som underopgave og afsluttet opgave. Som eksempel kan nævnes løsning af lineære ligninger. Her må findes både en bibliotekssekvens, der let kan indpasses i andre programmer, og et biblioteksprogram, der også besørger trykning, kontrol m.m.

En særstilling indtager de programmer, der vedrører maskinens kontakt med omverdenen: indlæse- og udlæseprogrammer, kontrolprogrammer til indkørsel o.s.v. På grund af deres opgaves afsluttede karakter regnes de med i gruppen af biblioteksprogrammer.

12.3. Bibliotekets inddeling.

Bibliotekssekvenser og -programmer betegnes med en forkortelse (henholdsvis to bogstaver og eet bogstav), der angiver emnegruppen, samt et nummer:

Sekvenser

AF	Algebraiske funktioner
XF	Eksponentialfunktioner
LF	Logaritmefunktioner
TF	Trigonometriske funktioner
CF	Cirkulære funktioner
HF	Hyperbolske funktioner
EF	Elliptiske funktioner
BF	Besselfunktioner
ØF	Øvrige funktioner
LL	Lineære ligninger
AL	Algebraiske ligninger
DL	Differentialligninger
IL	Integralligninger
FR	Flydende regning
DR	Regning med dobbelt nøjagtighed
KR	Kompleks regning
MR	Matrixregning
IG	Integration
HA	Harmonisk analyse
IP	Interpolation
ST	Statistik
FN	Faste del af normalleje

Programmer

L	Ligninger
S	Statistik
O	Operationsanalyse
M	Anden matematik
I	Indlæsning
U	Udlæsning
H	Hulkortadministration
A	Autokodning
K	Kontrol ved indkørsel

12.4. DASK-biblioteksspecifikationer.

For hver sekvens og hvert program udgives en specifikation. Disse specifikationer indeholder følgende:

a) Grundoplysninger.

b) En beskrivelse af det matematiske eller kodningsmæssige grundlag for sekvensen (programmet).

c) En beskrivelse af sekvensens (programmets) funktion. som supplement til pkt. a.

samt for sekvensernes vedkommende:

d) Et rutediagram.

e) Koden med forklaring.

12.5. Grundoplysninger om sekvenser

Grundoplysningerne findes i skemaform på specifikationens forside. Følgende oplysninger gives (se eksemplerne side 12.5 og 12.6):

1. Indhopsadresser. Det bemærkes, at hop til en bibliotekssekvens altid er indekshop.

2. Udhopsadresser.

3. Indgangsbetingelser, som må være opfyldt inden indhoppet til bibliotekssekvensen.

4. Udgangsresultat. Her gælder den regel, at de registre eller pseudoregistre, hvis indhold ændres, bliver nævnt. (Dette gælder naturligvis kun de relevante registre eller pseudoregistre; f.eks. må det betragtes som en selvfølge, at C(AR) og C(MR) ødelægges ved flydende regning.) Det bemærkes, at bibliotekssekvenserne aldrig ødelægger indholdet i indeksregistrene (naturligvis berøres 1RD af indekshoppet).

5. Det maksimale antal ordrer, der gennemløbes fra indhop til udhop.

6. Minimal og maksimal køretid.
   Der findes sekvenser, hvor den minimale (maksimale) køretid kun forekommer i visse, specielle tilfælde. Her anføres den normalt forekommende minimale (maksimale) køretid, medens den absolut minimale (maksimale) køretid tilføjes i parentes.
   Køretiden angives i additionstider (AT), millisekunder (ms) eller sekunder (s).

7. Kodelængde. Her kan eventuelt angives flere kodelængder. F.eks. gælder det for nogle biblioteksekvenser, at den sidste del af sekvensen kun bruges ved flydende regning, men er overflødig ved regning med DASK-tal.

8. Eventuelt krav om lige begyndelsesadresse.

9. Grundparametre, hvorved forstås parametre, der skal placeres i etikettecellerne.

10. Programparametre, hvorved forstås parametre, der skal stå i hovedsekvensen umiddelbart efter den pågældende indekshopordre. Er der n programparametre, vil udhopsordren i biblioteksekvensen have formen n+1 D 10.

11. Undersekvenser, d.v.s. andre biblioteksekvenser, der skal være lagret sammen med den pågældende biblioteksekvens. For undersekvenserne angives eventuelt et etikettemærke; undersekvensens begyndelsesadresse skal altså stå i den tilsvarende etikettecelle under indlæsningen af den pågældende biblioteksekvens.

12. Arbejdsceller.

13. Benyttelse af permanente konstanter i den faste del af normallejet.

14. Eventuelt andre oplysninger.

REGNECENTRALEN

DANSK INSTITUT FOR MATEMATIKMASKINER

DASK - BIBLIOTEKSSPECIFIKATION

SEKVENS-BETEGNELSE
AF 1
side 1/

Kodet af NIB d. 15-4 1957

Indkørt af PWP d. 8-10 1957

Udgivet d.

$$y = \sqrt{x}$$

Indhops-adresser	Udhops-adresser	Indgang	Udgang	Max. ordre-antal	Køretid min.	Køretid max.
0A8	21A8	C(AR) = x (0 ≤ x < 1)	$\sqrt{x}$ → AR & MR	22	111 AT (7 AT)	ca.114 AT (117½ AT)

Kodelængde	0 - 25	Undersekvenser	Ingen
Begyndelsesadresse	Lige	Arbejdsceller	I sekvensen
Grundparametre	Ingen	Perm. konstanter	C(2042v)
Programparametre	Ingen		

REGNECENTRALEN

DANSK INSTITUT FOR MATEMATIKMASKINER

DASK - BIBLIOTEKSSPECIFIKATION

SEKVENS-BETEGNELSE
LF 1
side 1/

Kodet af PA JB d. 29-4 1958

Indkørt afWH HBH d. 13-5 1958

Udgivet d.

$y = \log_a x$

(a = 2, e, 10)

Indhops-adresser	Udhops-adresser	Indgang	Udgang	Max. ordre-antal	Køretid min.	Køretid max.
0A8	30A8	C(AR) = x ($\frac{1}{2} \leq x < 1$), k = 1	$\log_2 x$ → AR & MR	31	106½ AT	106½ AT
		k = 2	$\log_e x$ → AR & MR	34	115 AT	115 AT
		k = 4	$\log_{10} x$ → AR & MR	34	115 AT	115 AT
56A8	78A8	C(FAR) = x, k = 1	$\log_2 x$ → FAR	51	ca.129 AT (128½ AT)	ca.130 AT (134½ AT)
		k = 2	$\log_e x$ → FAR	54	ca.138 AT (137 AT)	ca.139 AT (143 AT)
		k = 4	$\log_{10} x$ → FAR	54	ca.138 AT (137 AT)	ca.139 AT (143 AT)

Kodelængde	0 - 55 (DASK-tal) 0 - 78 (flydende tal)	Undersekvenser	Ingen
Begyndelsesadresse	Lige	Arbejdsceller	I sekvensen, samt 2002v
Grundparametre	Ingen	Perm. konstanter	C(2040v), C(2042v), C(2043)
Programparametre	kA00		

KAPITEL 13.

DASK - normalleje 1

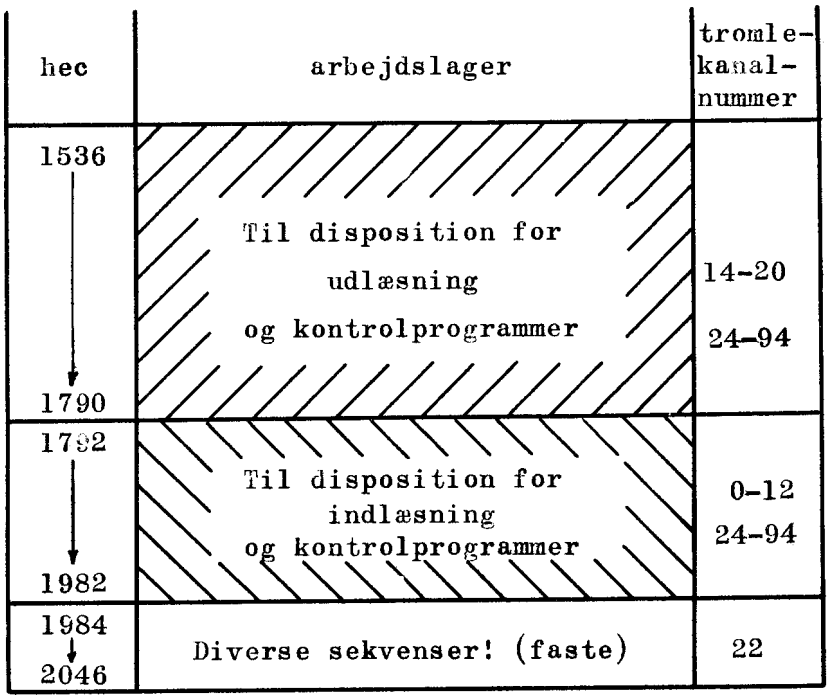
For at kunne indlæse et program fra hulstrimmel til arbejdslageret, må der forud være lagret et indlæseprogram som nævnt i kapitel 9.

Vi skal ikke nærmere diskutere dette tilsyneladende paradoksale problem, men blot fastslå at "DASK - klar til brug" har lagret et indlæseprogram samt iøvrigt et udlæseprogram og visse andre sekvenser i en del af arbejdslageret og på nogle af tromlekanalerne.

Dette kompleks af sekvenser, som kort benævnes "DASK - normalleje I", deles naturligt i en fast del lagret i helcellerne 1984-2046 samt i kanal 22 og en variabel del lagret på tromlekanalerne 0 - 94. Udlæsning af tromlekanalprogrammer til arbejdslageret sker til helcellerne 1536-1982.

"DASK - normalleje 1" optager altså en forholdsvis stor del af hele arbejdslageret, men kun helcellerne 1984-2046 må betragtes som absolut urørlige, idet den variable del - helcellerne 1536-1982 - kun behøver at stå til disposition under indlæsning og udlæsning samt under indkøring. Ind- og udlæseprogrammerne er sammen med diverse kontrolprogrammer (se kapitel 16) lagret i tromlekanalerne, hvis indhold udlæses til disse helceller.

En nærmere omtale af ind- og udlæseprogrammerne er givet i kapitlerne 9 og 15; vi skal derfor blot indskrænke os til at specificere den faste del af normallejet, d.v.s. sekvenserne i hec 1984-2046.



Skitse over disponeringer af arbejds- og tromlelager i DASK - normalleje 1.

Hac 1984 - 1986:

Program til overførsel af udlæseprogrammet.
Indeksindhop til ordre 1984 med tilbagehop til HS.

Hac 1987 - 1989:

Fast del af indlæseprogrammet: Alle hopordrer til 1987 bevirker at indlæsning fra strimmel påbegyndes hvorefter strimmelens etiketter styrer det videre forløb.

Hac 1990 - 1995:

Reservation af administration af kontrolprogrammer.

Hac 1996 - 2007:

Reservation af flydende registre: FMD, FAR og FMR.

Hac 2008 - 2015:

Reservation af etikettecellerne.

Hac 2016 - 2020:

Program for pakning u. afrunding af C(FAR) ⟶ AR
Indeksindhop til ordre 2016.

Hac 2021 - 2025:

Program for oppakning af pakket flydende tal i AR til FMD.
Indeksindhop til ordre 2021.

Hac 2026 - 2030:

Program for oppakning af pakket flydende tal i AR til FAR.
Indeksindhop til ordre 2026.

Hac 2031 - 2035:

Program for oppakning af flydende tal i AR til FMR.
Indeksindhop til ordre 2031.

Hac 2036 - 2038:

Program for korrektion af talværdi hvis der er spild ved fortegnsskifte. Ved indekshop til dennesekvens (kun 3 ordrer) hoppes straks tilbage til hovedsekvens hvis ikke-spild i AR. Hvis spild i AR erstattes indholdet med talværdien $1 - 2^{-39}$.

Hac 2039 - 2043:

Det har vist sig hensigtsmæssigt at lagre fast i arbejdslageret visse konstanter og talværdier, som ofte finder anvendelse i hovedprogrammer eller biblioteksekvenser:

$\pm$ C(2039) = $\pm 2^{-11}$ eller $\pm 2^{-31}$
C(2040v) = - 1
$\pm$ C(2041) = $\pm 2^{-19}$ eller $\pm 2^{-39}$
C(2042v) = 0
$\pm$ C(2043) = $\pm 2^{-1}$ eller $\pm 2^{-21}$
$\pm$ C(2040) = $\pm (1 - 2^{-39})$

Hac 2044 - 2047:

Reservation for administration af kontrolprogrammer.

Resumé

Indhops-adresser 1984	Udhops-adresser (1542)	Indgang -	Udgang Udlæsnings-program → hec 1536-1790	Max. ordre-antal (10)	Køretid 150 ms
1987	(1802 1980)	(evt. lb.adr. i IRB)	del af indlæsningsprogram → hec 1792 - 1854 hop t.indl.	3	ca. 30 ms
1991 1992	1981	jfr. kap.16	jfr. kap. 16	-	-
2016	2020	x = C(FAR)	x pakket uden afr. → AR	5	6 $\frac{1}{2}$ AT
2021	2025	x = C(AR)	x oppakket → FMD	5	7 AT
2026	2030	x = C(AR)	x oppakket → FAR	5	7 AT
2031	2035	x = C(AR)	x oppakket → FMR	5	7 AT
2036	2036 (2048)	x = C(AR)	x → AR hvis ikke spild $1-2^{-39}$ → AR hvis spild	3	1 AT

Permanente konstanter.

hac			hac
		1 A 00 ~ 2^{-11} (2^{-31})	2039
2040	0 C 00 ~ -1	0 A 01 ~ 2^{-19} (2^{-39})	2041
2042	0 A 00 ~ 0	1024 A 00 ~ 2^{-1} (2^{-21})	2043

Etiketteceller: 2008 - 2015 (adresse pos.)

Flydende registre.

FMD = {hec 1996 , hac 1999} = FMD1 , FMD2.
FAR = {hec 2000 , hac 2003} = FAR1 , FAR2.
FMR = {hec 2004 , hac 2007} = FMR1 , FMR2.

Kode.

HS indhop	→	1984	14 A 1C	vælg tromlekanal 14
		1985	1536 A 1D	læs kanal til hec 1536-1598
viderehop før tilbagehop	←	1986	1537 A 10	hop til ordre i hac 1537
HS indhop	→	1987	2 A 1C	vælg tromlekanal 2
		1988	1792 A 1D	læs kanal til hec 1792-1854
udhop	←	1989	1792 A 10	hop til indlæsning
		1990	(0) A 13	
		1991	(0) A 13	
		1992	2037 A 74	administration af kontrolprogrammer (jfr. kap. 16)
		1993	1984 A 08	
		1994	2045 A 12	
		1995	2044 A 10	
		1996	A	
		1997	A	
		1998	A	
		1999	A	
		2000	A	FMD
		2001	A	FAR
		2002	A	FMR
		2003	A	
		2004	A	
		2005	A	
		2006	A	
		2007	A	
		2008	A	
		2009	A	
		2010	A	
		2011	A	etiketteceller
		2012	A	
		2013	A	
		2014	A	
		2015	A	

C(FAR) til AR u.afr.	→	2016	2001 A 40	C(FAR1h) → ARh
		2017	12 A 0F	12 skift th. uden tegn
		2018	2003 A 00	z" adderet til ARhadr (tom)
		2019	2000 A 20	C(FAR1v) + C(AR) → AR
udhop	←	2020	1 D 10	udhop
C(AR) til FMD:	→	2021	1996 A 28	a_1' → FMD1v
		2022	1999 A 09	a" → FMD
		2023	12 A 0C	a_2' → ARh
		2024	1997 A 08	a_2' → FMD1h
udhop	←	2025	1 D 10	hop til HS
C(AR) til FAR:	→	2026	2000 A 28	a_1^{φ} → FAR1v
		2027	2003 A 09	a" → FAR
		2028	12 A 0C	a_2' → ARh
		2029	2001 A 08	a_2^{φ} → FAR1h
udhop	←	2030	1 D 10	hop til HS
C(AR) til FMR	→	2031	2004 A 28	a_1' → FMR1v
		2032	2007 A 09	a" → FMR
		2033	12 A 0C	a_2 → ARh
		2034	2005 A 08	a_2'' → FMR1h
udhop	←	2035	1 D 10	hop til HS
korrektion v.spild v.fortegnsskifte	→	2036	1 D 52	tilbagehop hvis ikke spild
		2037	2040 A 41	$1-2^{-39} \sim 1$ → AR
udhop	←	2038	1 D 10	udhop
		2039	1 A 00	permanente konstanter
		2040	0 C 00	
		2041	0 A 01	
		2042	0 A 00	
		2043	1024 A 00	
		2044	2038 A 68	administration af kontrol-programmer.
		2045	24 A 1C	
		2046	1836 A 1D	
		2047	1886 A 10	

KAPITEL 14.

Flydende regning

14.1. Indledning

I kapitel 8 er forklaret, hvordan man kan regne med tal uden for DASK-intervallet $-1 \leq x < 1$. En af metoderne var regning med flydende komma, flydende regning, hvor der til hvert tal hører en individuel kommaplaceringsangivelse.

For at lette flydende regning er der udarbejdet en bibliotekssekvens FR 1; desuden findes hjælpesekvenser m.m. i den faste del af normalleje 1. I dette kapitel skal anvendelse af disse hjælpemidler forklares og illustreres.

14.2. Definition

Først skal man minde om definitionen af flydende tal.

Et flydende tal x repræsenteres af to størrelser x' og x", taldelen og eksponenten, defineret ved

$$x = x' \cdot 2^{x''-1024} .$$

x' er et normaliseret DASK-tal, evt. 0 eller $-\frac{1}{2}(-1 \leq x' \leq -\frac{1}{2}$, $x' = 0$, $\frac{1}{2} \leq x' < 1)$. x" er et heltal $0 \leq x'' \leq 2047$.

Denne definition er ikke entydig for $x = 0$ og $x = -2^{-n}$. For $x = 0$ er x" vilkårlig; her udvides definitionen med vedtægten $x'' = 0$ for $x = 0$. For $x = -2^{-n}$ er der to mulige taldele, $x' = -1$ og $x' = -\frac{1}{2}$.

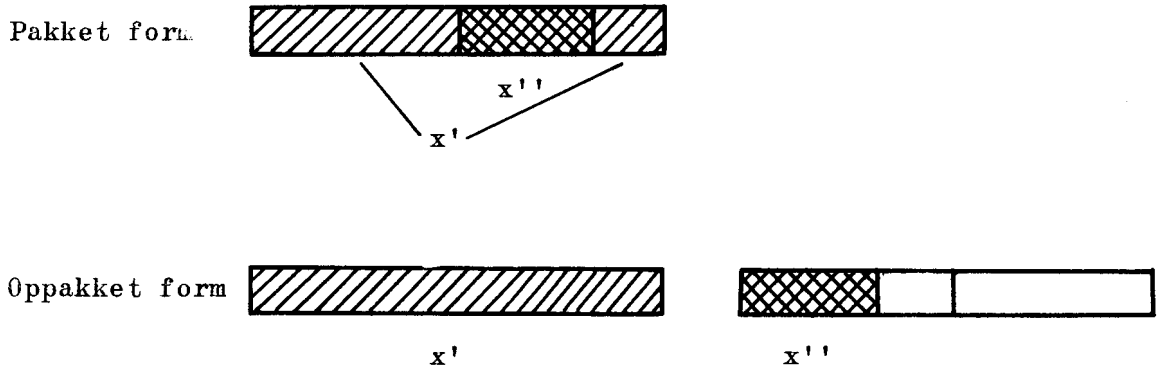
Definitionen omfatter intervallet

$$10^{-309} \approx 2^{-1025} \leq |x| \leq 2^{1023} \approx 10^{308} ,$$

det flydende interval.

14.3. Lagringsformer for flydende tal

Flydende tal lagres på pakket eller oppakket form. Et pakket tal fylder en helcelle; x' placeres i venstrehalvcellen plus højrehalvcellens operationsdel, x" placeres i højrehalvcellens adressedel. Et oppakket tal fylder en helcelle og den efterfølgende venstrehalvcelle; x' placeres i helcellen, x" i den efterfølgende venstrehalvcelles adressedel. Denne lagringsform knytter sig til indlæseprogrammets virkemåde.



Et tal på oppakket form har altså 39 binaler i taldelen, medens et tal på pakket form har 27 binaler.

I den faste del af normalleje 1 findes sekvenser, der besørger transformation mellem den pakkede og den oppakkede form.

4 Flydende registre

I den faste del af normalleje 1 er der reserveret celler til 3 pseudoregistre, nemlig

flydende multiplikandregister	FMD:	FMD 1, FMD 2 = hec 1996, hhac 1999
flydende akkumulatorregister	FAR:	FAR 1, FAR 2 = hec 2000, hhac 2003
flydende multiplikatorregister	FMR:	FMR 1 FMR 2 = hec 2004, hhac 2007

I disse registre anbringes tallene, når der skal regnes med dem.

Flydende tal lagres på oppakket form i pseudoregistrene, idet x" dog - af kodetekniske grunde - står i højrehalvcelle.

Registrene er indført, fordi man af pladshensyn normalt lagrer talmaterialet på pakket form, medens man naturligvis kun kan regne med tal på oppakket form.

5 Biblioteksekvensen FR 1

Sekvensen FR 1 udfører operationerne addition, multiplikation og division på flydende tal anbragt i pseudoregistrene.

Forsiden af biblioteksspecifikationen findes side 14.3.

Vedrørende enkeltheder henvises til selve specifikationen. Her skal blot nævnes, at FR 1 bevirker stop af maskinen, hvis den numerisk øvre grænse for det flydende interval overskrides, eller hvis en divisor er nul. Hvis den numerisk nedre grænse for det flydende interval overskrides, sættes tallet lig nul.

REGNECENTRALEN

DANSK INSTITUT FOR MATEMATIKMASKINER

DASK - BIBLIOTEKSSPECIFIKATION

SEKVENS-BETEGNELSE
FR 1
side 1/7

Kodet af studie-kreds d. 1-2 1957

Indkørt af JHH WH d. 10-10 1957

Udgivet d. -6-1958

Flydende tal:

addition, multiplikation, division

Indhops-adresser	Udhops-adresser	Indgang	Udgang	Max. ordre-antal	Køretid min.	Køretid max.
0A8	45A8	C(FMD) = y C(FAR) = x	x+y → FAR & FMD	39	32 AT (19 AT)	ca.39 AT (46 AT)
2A8	20A8 37A8 39A8		x+y → FAR	31	24 AT (11 AT)	ca.31 AT (38 AT)
50A8	37A8 39A8		$\frac{x}{y}$ → FAR	23	31 AT (31 AT)	31 AT (38 AT)
57A8	37A8 39A8	C(FAR) = x C(FMR) = y	xy → FAR	19	$21\frac{1}{2}$ AT (21 AT)	$21\frac{1}{2}$ AT ($28\frac{1}{2}$ AT)

Kodelængde	0 - 62	Undersekvenser	Ingen
Begyndelsesadresse	Lige	Arbejdsceller	I sekvensen, samt 1998v, 2002v, 2006v
Grundparametre	Ingen	Perm. konstanter	C(2039)
Programparametre	Ingen		

6 Korrektion af fortegn

Sekvensen FR 1 udfører ikke operationerne subtraktion, numerisk addition og numerisk subtraktion. Disse udføres ved i hovedsekvensen at korrigere taldelens fortegn. Ønsker man f.eks. at skifte fortegn på C(FAR), koder man således:

```
2000  A  41    - C(FAR 1) → AR
2036  A  16    hop til korr. for -(-1)
2000  A  08    C(AR) → FAR 1
```

(NB. Man kan ikke korrigere fortegn på et tal på pakket form! Derimod kan man - på grund af vedtægten x" = 0 for x = 0 - afgøre, om et tal på pakket form er nul.)

7 Eksempler og øvelser

Den praktiske anvendelse af FR 1 og den faste del af normalleje 1 kan nu illustreres ved eksempler. Desuden gives nogle øvelsesopgaver.

Eksempel 1

Givet de to flydende pakkede tal a og b. Beregn a+b.

L(a) = 100
L(b) = 102
L(a+b) = 104

FR 1 i 0AB.

Kode:

```
0     100  A  40    a → AR
1    2021  A  16    a → FMD
2     102  A  40    b → AR
3    2026  A  16    b → FAR
4       2  AB 16    a+b → FAR
5    2016  A  16    a+b → AR
6     104  A  08    a+b → 104
7       0  A8 30    stop
```

Eksempel 2

Givet de to 20-dimensionale vektorer $\bar{a} = (a_0\ a_1\ \ldots\ldots a_{19})$ og $\bar{b} = (b_0\ b_1\ \ldots\ldots b_{19})$. Dan det skalære produkt

$$\bar{a} \cdot \bar{b} = \sum_{i=0}^{19} a_i b_i$$

$L(a_i) = 100 + 2i$

$L(b_i) = 200 + 2i$

$L(\bar{a} \circ \bar{b}) = 98$

FR 1 i 0A9.

Kode:

		0	1	A8	50	0 → AR
		1	2021	A	16	0 → FMD
		2	40	A	55	40 → IRC, d.v.s. 20 ⇒ i
10	→	3	2046	C	55	C(IRC)-2 → IRC, d.v.s. i-1 ⇒ i
		4	100	C	40	a_i pakket → AR
		5	2031	A	16	a_i → FMR
		6	200	C	40	b_i pakket → AR
		7	2026	A	16	b_i → FAR
		8	57	A9	16	$a_i b_i$ → FAR
		9	0	A9	16	$\sum_{v=i}^{19} a_v b_v$ → FAR og FMD
3	←	10	3	A8	53	hop til ordre 3 hvis i ≠ 0.
		11	2016	A	16	$\bar{a} \cdot \bar{b}$ pakket til AR
		12	98	A	08	$\bar{a} \cdot \bar{b}$ → hec 98
		13	0	A8	30	stop.

Den færdighullede hulstrimmel får da følgende form, idet vi vælger at lagre det samlede program fra hac 714.

714	E	3	714 → lb.adresse
0	E	19	lb.adresse = 714 → hac 2009
0	E	0	lb.adresse = 714 → hac 2008
FR 1			program for flydende regning
0	E	0	lb.adresse → hac 2008
HS			program for skalært vektorprodukt
0	E8	50	stop, hop til ordre 0A8 i HS

Eksempel 3

Beregn polynomiet

$$b = \sum_{i=0}^{39} a_i x^i$$

$L(a_i) = 100 + 2i$

$L(x) = 98$

$L(b) = 96$

FR1 i 0A9

Kode:

		0	98	A	40	x pakket → AR
		1	2031	A	16	x → FMR
		2	78	A	35	78 → IRB, d.v.s. 39 ⟹ i
		3	100	B	40	a_{39} pakket → AR
		4	2026	A	16	a_{39} → FAR
10	→	5	2046	B	35	C(IRB)-2 → IRB, d.v.s. i-1 ⟹ i
		6	57	A9	16	$\sum_{v=i+1}^{39} a_v x^{v-i}$ → FAR
		7	100	B	40	a_i pakket → AR
		8	2021	A	16	a_i → FMD
		9	2	A9	16	C(FAR)+a_i = $\sum_{v=i}^{39} a_v x^{v-i}$ → FAR
5	←	10	5	A8	33	hop til ordre 5 hvis i ≠ 0
		11	2016	A	16	b pakkes til AR
		12	96	A	08	h pakket til hec 96
		13	0	A8	30	stop.

Sammensætning af FR 1 og HS samt etiketter er ganske analog sammensætningen i eks. 1.

Eksempel 4

Beregn determinanten af matricen

$$\begin{Bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{Bmatrix}$$

Elementerne forudsættes flydende, pakkede og lagret rækkevis i helcellerne 108, 110, 112 og 114. Determinanten lagres pakket i hec 116.

FR 1 i 0A9

Kode:

0	108	A	40	a_{11} pakket → AR
1	2031	A	16	a_{11} → FMR
2	114	A	40	a_{22} pakket → AR
3	2026	A	16	a_{22} → FAR
4	57	A9	16	$a_{11}\cdot a_{22}$ → FAR
5	2016	A	16	$a_{11}\cdot a_{22}$ pakket → AR
6	2021	A	16	$a_{11}\cdot a_{22}$ → FMD
7	110	A	40	a_{12} pakket → AR
8	2031	A	16	a_{12} → FMR
9	112	A	40	a_{21} pakket → AR
10	2026	A	16	a_{21} → FAR
11	2000	A	41	$-a'_{21}$ → AR
12	2036	A	16	korrektion for spild (-(-1))
13	2000	A	08	$-a'_{21}$ → FAR1
14	57	A9	16	$-a_{21}\cdot a_{12}$ → FAR
15	2	A9	16	$a_{11}\, a_{22} - a_{21}a_{12}$ → FAR
16	2016	A	16	determinant pakkes til AR
17	116	A	08	determinant til hec 116
18	0	A8	30	stop.

Øvelse 1 Sammensæt FR1 og HS med etiketter i eks. 4.

Øvelse 2 Beregn determinanten af en matrix af 3'de orden med lignende forudsætninger som i eks. 4.

Øvelse 3 Beregn værdien af kædebrøken x_{16}

$$x_{16} = b_o + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \cfrac{\cdots\cdots\cdots}{b_{15} + \cfrac{a_{16}}{b_{16}}}}}}$$

$$\left.\begin{aligned} L\,(b_i) &= 300 + 2i \\ L\,(a_i) &= 400 + 2i \end{aligned}\right\}\ a_i \text{ og } b_i \text{ flydende pakkede tal}$$

x_{16} pakket til helcelle 1012.

Eksempel 5

Beregn ln $\Gamma(x) \longrightarrow$ FAR.

$L(x)$ = FAR $\quad (x \geq 2)$.

Forudsætninger:

1) Benyt approksimationsformlen

$\ln \Gamma(x) = \ln \sqrt{2\pi} - x + (x - \frac{1}{2})\ln x + P(\frac{1}{x})$,

hvor $P(\frac{1}{x})$ angiver et polynomium med argumentet $\frac{1}{x}$.

2) Bibliotetssekvens FR1.

Begyndelsesadresse 0A9.

3) Biblioteksekvens LF1.

lnC(FAR) $\longrightarrow$ FAR. Programparameter 2A00.

Indhopsadresse 56A8.

Begyndelsesadresse: 0AA.

4) Undersekvens til beregning af $P(\frac{1}{x})$

foreligger. C(FAR) = $\frac{1}{x}$ før indhop

$P(\frac{1}{x})$ = C(FAR) efter udhop.

Indhopsadresse 0A8.

Begyndelsesadresse 0AB.

0	2016	A	16	}	x $\longrightarrow$ arbc
1	36	A8	08		
2	56	AA	16	}	lnx $\longrightarrow$ FAR
3	2	A	00		
4	2000	A	40	}	C(FAR) $\longrightarrow$ FMR
5	2004	A	08		
6	2003	A	60		
7	2007	A	28		
8	32	A8	40	}	$-\frac{1}{2} \longrightarrow$ FAR
9	2026	A	16		
10	36	A8	40	}	x $\longrightarrow$ FMD
11	2021	A	16		
12	2	A9	16		$x - \frac{1}{2} \longrightarrow$ FAR
13	57	A9	16		$(x - \frac{1}{2})\ln x \longrightarrow$ FAR

14	1996	A	41	$-x \longrightarrow$ FMD
15	1996	A	08	
16	2	A9	16	$-x + (x - \frac{1}{2})\ln x \longrightarrow$ FAR
17	34	A8	40	$\ln \sqrt{2\pi} \longrightarrow$ FMD
18	2021	A	16	
19	2	A9	16	$K - \ln \sqrt{2\pi} - x + (x - \frac{1}{2})\ln x \longrightarrow$ FAR
20	2016	A	16	$K \longrightarrow$ arbc
21	38	A8	08	
22	48	A9	40	$1 \longrightarrow$ FAR
23	2026	A	16	
24	36	A8	40	$x \longrightarrow$ FMD
25	2021	A	16	
26	50	A9	16	$1/x \longrightarrow$ FAR
27	0	AB	16	$P(1/x) \longrightarrow$ FAR
28	38	A8	40	$K \longrightarrow$ FMD
29	2021	A	16	
30	2	A9	16	$\ln \Gamma(x) \longrightarrow$ FAR
31	0	A8	30	stop

32	DD5A			$-\frac{1}{2}$
33				
34	CD91893...A			$\ln \sqrt{2\pi}$
35				
36		A		arbc
37		A		
38		A		
39		A		

Sammensætningen af biblioteksekvenser, undersekvens og hovedsekvens med etiketter bliver

```
200  E   69
  0  E9   3
  0  E    0

   FR 1

  0  E    2
  0  E   1A

   LF 1
```

```
0  E   2
0  E   1B

   US

0  E   2

   HS

0  E8  40
```

Øvelse 4

Fortolk etikettefunktionerne i eksempel 5.

KAPITEL 15.

15.1 Udlæsningsprogrammet i DASK-NORMALLEJE 1.

Programmet udlæser kun talværdier og gengiver dem udelukkende i 10-talsystemet. I de ganske specielle tilfælde, hvor der måtte ønskes udskrivning i 2-talsystemet eller det sedecimale system, må man benytte andre programmer.

Vi giver i afsnittet, der følger efter denne indledning, en kortfattet anvisning i brugen af udlæsningsprogrammet, idet vi først viser, hvordan forbindelsen mellem koden (HS) og programmet etableres, og derefter gennem eksempler, hvad der yderligere skal gøres.

Hensigten er at lette oversigten og vise omfanget af, hvad koderen skal foretage sig for at anvende programmet. Der vil herved blive udeladt visse - ikke altid uvæsentlige - ting, men det vil dog ofte være muligt at jævnføre et foreliggende talmateriale med eksemplerne og deraf uddrage alle fornødne oplysninger til at få en tilfredsstillende udlæsning (d.v.s. trykning eller stansning) i stand.

Vi fremhæver imidlertid, at en effektiv eller blot sikker udnyttelse af udlæsningsprogrammet kræver, at man også sætter sig ind i afsnit 15.3, hvor der gives en udførlig beskrivelse af de mange muligheder, programmet indeholder.

15.2 Kortfattet beskrivelse.

Forbindelse mellem koden (HS) og udlæsningsprogrammet.

Den typiske brug af programmet foregår på følgende måde.

a. i HS indfører koderen et indekshop til hac 1566 og i de to næste halvceller i HS, - som ikke kommer til at indeholde egentlige ordrer, men såkaldte pseudoordrer - skriver man parametre, der skal benyttes i udlæsningsprogrammet. Ved indhoppet til hac 1566 overføres parametrene til dette program, der herved indstilles, "trimmes", så det kan løse den specielle trykopgave, det i øjeblikket drejer sig om.

b. Tilbagehoppet til HS, når trimningen er fuldført, sker til ordren efter de to pseudoordrer i HS, og efter dette tilbagehop kan man straks eller senere hoppe til det egentlige udlæsningsprogram ved at kode et indekshop til hac 1550 dog efter at have anbragt tallet, der skal trykkes i AR eller FAR. (C(AR) ødelægges ved trimning, C(FAR) derimod ikke).

Først da foregår udlæsningen, sådan som den er blevet forberedt gennem trimningen. Det er altid det øjeblikkelige indhold i AR eller i FAR, der danner basis for det, der udlæses. C(FAR) vil være bevaret efter udlæsningen, mens C(AR) går tabt.

I HS kommer således til at stå:

	∘ ∘ ∘		
Til trimning ←	r 1566 A 16		
	r+1 B $s_1 s_2$ h b d	Sedecimal indlæsning af de 5 cifre efter tegnet B.	
	r+2 n A t k	Pseudoordre på almindelig ordreform.	
fra trimning →	r+3 ∘ ∘ ∘	vilkårligt antal ordrer til typografi, regning eller andet	
	∘ ∘ ∘		
	∘ ∘ ∘		
til udlæsning ←	r+q 1550 A 16	et tal svarende til indholdet i AR eller FAR udlæses. C(AR) bliver ødelagt, C(FAR) ikke.	
fra udlæsning →	r+q+1 ∘ ∘ ∘		
	∘ ∘ ∘		

Det skal straks fremhæves, at de 8 størrelser: $s_1 s_2$ h b d t k og n, hvoraf de syv første er sedecimale cifre det sidste et decimalt tal, kun vedrører selve tallet, der skal trykkes, nemlig cifferantal, kommaplacering, etc., mens typografi som linieskift o.lign. må kodes særskilt, og her vil altså det, der skal ske før tallet trykkes, kunne kodes enten før ordre r eller et passende sted mellem ordrerne r+2 og r+q.

Specifikation af størrelserne s_1 s_2 h b d n t k.

Vi skal nu i tilknytning til nogle konkrete talmaterialer gennemføre bestemmelsen af de 8 størrelser i pseudoordrerne. Eksemplerne er valgt, så de til en vis grad kan betragtes som typiske.

Det decimale tal n og cifret k kan vi behandle samlet for dem alle:

n angiver adressen på den ordre, man ønsker udlæsningsprogrammet skal hoppe til, når der opstår fejl ved udlæsningen. Dog er der her kun tale om en bestemt type fejl, nemlig sådanne, der bevirker en indre modstrid under udlæsningen. Hvad der menes hermed, vil senere blive forklaret.

Vi tænker os, man har valgt adressen n_1.

k sættes lig 0; de øvrige værdier, k kan antage, er knyttet til specielle forhold, som vi her vil lade ude af betragtning.

Vi nævner lige, at fortegnet +, i modsætning til fortegnet -, ikke udlæses (altså heller ikke stanses), men erstattes af mellemslag.

Eks. 1. DASK-tallene + 0.015 913 4....
- 0.000 338 1....
+ 0.008 155 6....

skal udlæses med 5 decimaler, altså som

0.01591
-0.00034
0.00816

De anbringes efter tur i AR umiddelbart før indekshoppet til hac 1550.

$s_1=0$, $s_2=0$ { s_1 og s_2 tager sig hver af flere ting, men for begges vedkommende vil værdien 0 være den, der svarer til almindelig praksis og altid giver en talmæssig fuldt anvendelig udlæsning.

h=1 antallet af cifre foran kommaet (heltalscifre, her nullet)

b sættes til summen af h og det følgende d, altså her 6

d=5 antallet af decimaler

$n=n_1$

t=1 tallagring 1, som bl.a. dækker DASK-tal

k=0

I koden skal derfor stå:

```
    .     .    .  .
←r    1566  A 16
  r+1 B 00165
  r+2   n1  A 10
→r+3    .    .  .
    .     .    .  .


    .     .    .  .
←r+q  1550  A 16     hop til udlæsning
→r+q+1  .    .  .
    .     .    .  .
```

Eks. 2. De samme DASK-tal som i eks. 1. og afrundet på samme måde, skal udlæses; men nu med faktoren 10^4, d.v.s. som:

159.1
- 3.4
81.6

DASK-tallene, altså +0.0159134.. o.s.v., anbringes som før i AR umiddelbart inden hoppet til 1550

s_1=0
s_2=0
h=3 det maksimale antal cifre foran kommaet
b=4 =h+d
d=1. antallet af decimaler
n=n_1
t=1 tallagring 1, som dækker bl.a. 10^p.DASK-tal.
k=0

I koden skal derfor stå:

```
                    .     .    .  .
til trimning ←—  r    1566  A 16
                 r+1  B00341
                 r+2  n1    A 10
              samt yderligere:

fra trimning —→  r+3  1028  A 75   eksponenten 4 + 1024 til IRD
                 r+4  1607  A 74        "     4 + 1024  "  hac
                 .     .    .  .   1607 adresse pos.
                 .     .    .  .
              ←— r+q  1550  A 16   hop til udlæsning.
              —→ r+q+1  .    .  .
                 .     .    .  .
```

NB. Når udlæsningen af en sådan gruppe tal er færdig, bør man altid sørge for at indsætte 1024 til hac 1607's adressedel ved ordrerne

1024 A 75
1607 A 74 ,

inden man går videre.

Eks. 3. Følgende flydende tal (gengivet ved deres værdi) ønskes udlæst med 4 betydende cifre:

$$\left.\begin{array}{r} -\ 4\ 318\ 613.6 \\ +\ \ \ \ \ \ 896.821\ 13 \\ \ \ \ \ \ \ \ \ 0.005\ 681\ 3577 \end{array}\right\} \text{d.v.s. som} \left\{\begin{array}{r} -\ 4\ 319\ 000 \\ \ \ \ \ \ \ \ \ 896.8 \\ \ \ \ \ \ \ \ -\ 0.005\ 681 \end{array}\right.$$

De anbringes efter tur i FAR, d.v.s. på oppakket form, umiddelbart før indekshoppet til 1550.

$s_1=0$

$s_2=0$

h =7 det maksimale antal cifre foran kommaet

b =4 det faste antal betydende cifre

d =6 det maksimale antal decimaler, når kravet b=4 overholdes!

$n = n_1$

t =2 tallagring 2, som i det væsentlige dækker flydende tal

k =0

I koden:

	°	°	°	°
←	r	1566	A	16
	r+1	B	00746	
	r+2	n_1	A	20
→	r+3	..	°	°
	°	°	°	°
	°	°	°	°
←	r+q	1550	A	16
→	r+q+1	°	°	°
	°	°	°	°

Eks. 4. De samme flydende tal som i forrige eksempel ønskes nu udlæst med

5 betydende cifre

og som talpar d.v.s.

en taldel, som vi kan bestemme skal have f.eks. 1 ciffer foram kommaet

og en 10-potens eksponent (hel) stående for sig selv bag taldelen,

nemlig således:

$$-\ 4.3186 \quad 6$$
$$8.9682 \quad 2$$
$$-\ 5.6814 \quad -3$$

Ligesom i eks. 3 anbringes de oppakkede tal i FAR før hoppet til 1550.

$s_1=2$ for at få udlæsning med 10-potensfaktoren

$s_2=0$

h =1

b =5

d =4 nemlig b-h

$n = n_1$

t =2 stadig tallagring 2

k =0

I koden:

```
    .      .    .   .
←—  r    1566   A  16
    r+1  B 20154
    r+2    n1   A  20
—→  r+3    .    .   .
    .      .    .   .

    .      .    .   .
←—  r+q  1550   A  16
—→  r+q+1  .    .   .
    .      .    .   .
```

Eks. 5. Følgende heltal, med enheden i position 39 skal udlæses:

$$+\ 19\ 356$$
$$-\ \ \ 817$$
$$+206\ 158$$

DASK-tallene, altså $+ 19356 \cdot 2^{-39}$ o.s.v., anbringes efter tur i AR umiddelbart før hoppet til 1550.

$s_1=0$

$s_2=2$

h =6 det maksimale antal cifre foran kommaet

b =6 sættes lig summen af h og det efterfølgende d

d =0

$n = n_1$

t =3 tallagring 3, som i det væsentlige dækker $2^{39} \cdot C(AR)$

k =0

I koden:

```
        .      .  .  .
<-- r    1566 A 16
    r+1  B 02660
    r+2    n1 A 30
--> r+3    .  .  .
    .      .  .  .

    .      .  .  .
<-- r+q  1550 A 16
--> r+q+1  .  .  .
    .      .  .  .
```

Almindelige bemærkninger.

1. **Størrelserne h, b og d kan antage talværdierne**

   0,1,...,15

   og denne begrænsning skyldes, at de, som vi ved, skal udtrykkes ved ét sedecimalt ciffer, altså i pseudoordren skal stå som

   0,1,... 9,A,... F.

   Hertil må bemærkes at

   b=0 næppe har praktisk betydning, da denne værdi medfører at intet betydende ciffer læses ud

   og at

   b>11 giver udlæsning udover maskinens kapacitet, idet AR's og FAR1's 39 binaler jo kun svarer til 11-12 cifre i 10-talsystemet. Det vil derfor normalt være uden værdi at lade mere end 11 betydende cifre udlæse, men gøres dette alligevel, vil kun de 11 første, og når $2^{39} \cdot C(AR)$ udlæses, endog kun de 10 første være sikre (dette sidste på grund af den specielle metode, der anvendes ved udlæsningen). Det er derfor i de fleste tilfælde rimeligt at begrænse b til

   $$1 \leq b \leq 11 \quad (B)$$

   og ved $2^{39} \cdot C(AR)$ til

   $$1 \leq b \leq 10 \quad (A).$$

2. Strækker det opgivne antal heltalscifre, h, ikke til ved den talværdi, der skal udlæses (er m.a.o. den pågældende størrelse større end 10^h) vil programmet hoppe til hac n_1. Det er altså en sådan indre modstrid programmet reagerer overfor. (Tallet 81673.4 bliver med h = 3 således ikke trykt som 81700, men der hoppes til n_1. Det andet ville kræve h=5, b=3 jvfr. eks. 3.).

Man kan i hac n_1 have en stopordre eller et hop til et kontrolprogram, men foruden kode eller regnefejl kan årsagen selvfølgelig også bare være en gal vurdering af de mulige regneresultater.

Som eksempel nævnes, at ændres betingelserne i eks. 1. til, at vi ikke vil have trykt 0 foran kommaet, d.v.s. vi sætter h=0, b=5, d=5, ville vi ved at inkludere -1.000.. blandt talene få hop til n_1, når dette tal skulle udlæses.

Færdigtrimmede trykprogrammer.

Til nogle bestemte typer talmaterialer, eller når det er mindre væsentligt, om udlæsningen er omhyggelig tilpasset det, der skal trykkes, og man derfor vil undgå at spekulere nærmere over cifferantal etc., kan man få en udlæsning ved et enkelt indekshop til bestemte halvceller, altså uden forudgående trimning. Det drejer sig simpelthen om, at man hopper til visse, faste trimninger i udlæsningsprogrammet.

Der er tre forskellige indhopsadresser; ved to af dem udlæses det aktuelle C(AR) og ved den ene det aktuelle C(FAR). Før indekshoppet må man som sædvanlig sørge for her at have anbragt det, der skal udlæses.

Fælles for alle tre trimninger er, at udhopsadressen n er sat til 0, men der er sørget for at udhoppet aldrig bliver aktuelt, idet h er valgt så stor, at det, der skal udlæses, altid holder sig numerisk under 10^h.

1. Indekshop til 1554 bevirker en udlæsning,
hvor trimningen er:
1. pseudoordre B801BB
2. -"- 0 A 10,
hvilket igen betyder at:
C(AR)≠ -1 udlæses
med 0 foran kommaet
og 11 decimaler.

C(AR)= -1 udlæses som
- 1.00000 00000 (kun 10 decimaler).
Vi gentager, at i koden skal kun stå:

```
        .     .  .  .
<---  r    1554 A 16
--->  r+1    .  .  .
```

2. Indekshop til 1558 bevirker en udlæsning,
hvor trimningen er:
1. pseudoordre B A01BA
2. pseudoordre 0 A 20 ,
hvilket vil sige at:
C(FAR) udlæses med en taldel og 10-potenseksponent,
taldelen har 1 (betydende) ciffer foran kommaet
11 betydende cifre
10 decimaler

Eks.:
8.51384 83442 5
- 1.31664 52127 - 1
4.21839 01405 - 3

3. Indekshop til 1562 bevirker en udlæsning,
hvor trimningen er:
1. pseudoordre B 82CC0
2. -"- 0 A 30,
hvilket vil sige at:
2^{39}·C(AR) udlæses
med maksimalt 12 heltalscifre
" 12 betydende cifre
ingen decimaler
og intet komma.

Eks. - 139

18041 52366

-50117 43986

h=b=C er valgt så stor for at sikre, at $\left|2^{39}\cdot C(AR)\right|$ altid er mindre end 10^h, hvilket netop kræver $h \geq 12$. Men som forklaret ovenfor vil man kun kunne stole på de 10 første cifre.

4. Indekshop til 1572.

Det drejer sig ved dette hop ikke om, at der hoppes til en fast trimning, men om at den trimning som man her skal lade følge efter ordren 1572 A 16, altså de sædvanlige to pseudoordrer, bliver virksom ved den trykning, der foregår umiddelbart i tilknytning til hoppet. Eks. Tallet der skal udlæses står i hec 100:

	:		:		
	r		100	A 40	
	r + 1		1572	A 16	C(100) trykkes i overensstemmelse
adresse lige	r + 2		Bs_1s_2	h bd	} trimningen
	r + 3		n	A tk	}
	:		:		

Vi understreger, at det er en betingelse at adressen (den absolutte) r + 2 er lige!

Pointen ved dette særlige hop er imidlertid ikke, at man specielt slipper for hoppet til trykning: 1550 A 16, men at det er indrettet så trimningen i r + 2 og r + 3 ikke ødelægger en tidligere trimning.

Om udnyttelsen af dette forhold henvises til side 15.29 og fremefter.

15.3. Fuldstændig beskrivelse.

t = 1,2 og 3

Tallagringer

Her skal gives en klassifikation af de forskellige talmaterialer, man kan komme ud for at skulle udlæse.

Det er nævnt, at det altid er C(AR) eller C(FAR), der danner basis for udlæsningen, og det drejer sig derfor blot om at specificere, hvordan disse størrelser skal fortolkes for at give det fuldt færdige resultat af regningerne. Svaret er selvfølgelig givet i koden, da det af den fremgår, med hvilken skalafaktor resultatet står i AR eller i FAR.

Trykprogrammet kan udlæse $10^{p_1} \circ 2^{p_2} \circ C(AR)$ og $10^{p_1} \circ 2^{p_2} \circ C(FAR)$, hvor p_1 og p_2 er heltal. Vi tænker os, at det endelige resultat a står i en celle m med skalafaktoren f, d.v.s. DASK-tallet $C(m) = f \circ a$. *) Koderen ønsker derfor at udlæse $a = f^{-1} \cdot C(m)$. Da f imidlertid ikke behøver at være af formen $10^{p_1} \circ 2^{p_2}$ (f kan være et hvilket som helst tal, f.eks. $\frac{1}{\pi 4}$), bliver det nødvendigt, inden udlæsningen, at foretage en multiplikation svarende til følgende omskrivning

$$a = \frac{f^{-1}}{10^{p_1} \circ 2^{p_2}} C(m) \circ 10^{p_1} \circ 2^{p_2} .$$

Eksponenterne p_1 og p_2 vælges, - og her er selvfølgelig mange muligheder - så brøken bliver et DASK-tal. Koderen skal så sørge for at multiplicere C(m) med denne brøk før udlæsningen, der derefter skal finde sted med faktoren $10^{p_1} \circ 2^{p_2}$.

Man kan her stille det spørgsmål, hvorfor problemets skalafaktorer ikke er valgt, så man ender med i facit at have skalafaktoren $10^{-p_1} \circ 2^{-p_2}$ i stedet for f. Selv om dette ofte kan gøres, vil der være situationer, hvor det er afgørende, at regningerne udføres med ganske bestemte skalafaktorer, som ikke uden videre kan tillempes et sådant krav, eller hvor en sådan fremgangsmåde i virkeligheden ville være besværligere end blot at lade (det ene tal) resultatet multiplicere, sådan som det er blevet vist.

Vi har som resultat af dette at alle forekommende tilfælde kan deles i:

tallagring 1, hvor
det udlæste skal have værdien C(AR) eller $10^{p_1} \circ 2^{p_2} \circ C(AR)$ t = 1.

tallagring 2, hvor
det udlæste skal have værdien C(FAR) eller $10^{p_1} \circ 2^{p_2} \circ C(FAR)$ t = 2.

Hertil kommer, at det er praktisk af tallagring 1 at udskille som et særligt tilfælde

tallagring 3, hvor
det udlæste skal have værdien $2^{39} \circ C(AR)$, eller $10^{p_1} \circ 2^{p_2} \circ (2^{39} \circ C(AR))$ t = 3.

Ved tallagring 1 og 3 skal tallet anbringes i AR før hoppet til den egentlige udlæsning. Ved tallagring 2 skal det anbringes oppakket i FAR før dette hop.

---

*) Ved flydende regning kan man gå frem på samme måde; iøvrigt vil C(FAR) vel oftest være det fuldt færdige resultat.

Udlæsning uden faktorer.

Når det ved de tre tallagringer drejer sig om at udlæse henholdsvis C(AR), C(FAR) og 2^{39}·C(AR), skal man kun iværksætte trimning og hop til det egentlige udlæsningsprogram :

```
                    .     .  .  .
til trimning <--    r    1566 A 16
                    r+1  B  s1 s2 h b d
                    r+2     n  A  tk
fra trimning -->    r+3     .  .  .
                    .       .  .  .
                    r+v     n1 I 40      tal --> AR  (f. eks.)
                    .       .  .  .
til udlæsning <--   r+q  1550 A 16
fra udlæsning -->   r+q+1   .  .  .
                    .       .  .  .
```

Trimningen i ordrerne r til r+2 kan gælde lige så mange udlæsninger (ved udhop til 1550), det skal være. Den brydes kun af en ny trimning, foretaget af koderen, men ikke ved, at man hopper til en af de faste trimninger i 1554, 1558 eller 1562. Dette forhold kan være til stor nytte, når f.eks. argumenter og udregnede tal skal trykkes ind mellem hinanden.

Bemærkning: Trimningen ødelægger C(AR).

Udlæsning med en faktor: $10^{p_1} \cdot 2^{p_2}$.

Faktoren skal indføres før hoppet til 1550, det egentlige tryk-program, og det skal gøres ved, at man anbringer eksponenterne som henholdsvis

$1024 + p_1$ i hac 1607's adressepositioner

$1024 + p_2$ i hac 1565's adressepositioner,

hvor p_1 og p_2 kan være positive eller negative.

Det kan være naturligt at tænke sig de ordrer, der skal til, anbragt umiddelbart efter trimningen, men der er intet i vejen for, at de f.eks. kan anbringes før trimningen. Vi får, når man deler tilfældene i tre:

10^{p_1}

```
  .        .    .  .
r+3  (1024+p1)  A  75
r+4       1607  A  74
  .        .    .  .
```

hvor det forudsættes, at hac 1565 har 1024 i adressepositionerne, d.v.s. at p_2 virkelig er 0.

2^{p_2}

```
  .        .    .  .
r+3  (1024+p2)  A  75
r+4       1565  A  74
  .        .    .  .
```

hvor det forudsættes, at hac 1607 har 1024 i adressepositionerne, d.v.s. at p_1 virkelig er 0.

$10^{p_1} \cdot 2^{p_2}$

```
  .        .    .  .
r+3  (1024+p1)  A  75
r+4       1607  A  74
r+5  (1024+p2)  A  75
r+6       1565  A  74
  .        .    .  .
```

For sådanne udlæsninger gælder, at en faktor vil være virksom sålænge eksponenterne står i 1607 og 1565. Altså vil en ny trimning, der ikke efterfølges (eller forudgås) af en indførelse af sine egne eksponenter, give udlæsning med faktoren fra forrige udlæsning.

Fjernelse af eksponenter foretages med ordrer som

```
1024  A  75
1607  A  74
1565  A  74 .
```

Imidlertid har man mulighed for at udlæse uden en faktor, men stadig bevare dens eksponenter i 1607 og 1565, og derved bliver man i stand til at veksle mellem udlæsninger med og uden faktor uden hver gang at skulle gennemløbe ordrer som de, der er vist ovenfor. Det gøres ved

1) at addere 8 til cifret s_1, hvis "grundværdi" iøvrigt bestemmes af helt andre forhold, som det vil blive forklaret side 15.14

2) at benytte en af de tre faste trimninger i 1554, 1558 og 1562 (hvis den iøvrigt opfylder de øjeblikkelige krav til trykningen).

Vi fremhæver, at det man herved opnår er, at eksponenten lades uforstyrret i sin halvcelle uden at være virksom, men at den efter en trimning, hvor 8 ikke er adderet til s_1 igen virker.

På side 15.29 vil blive vist, hvordan man kan kombinere udlæsninger med og uden faktorer.

h,b og d

Disse parametre har følgende betydning:

h angiver det maksimale antal heltalscifre,

b angiver det maksimale antal betydende cifre,

d angiver det maksimale antal decimaler

i talmaterialet, sådan som koderen vil have det udlæst.

Vi understreger, at dette betyder, at h, b og d refererer til det færdigt udskrevne tal, $10^{p_2} \cdot 2^{p_2} \cdot C(AR)$, $10^{p_1} \cdot 2^{p_2} \cdot C(FAR)$.

Da størrelserne skal angives ved et sedecimalt ciffer, er deres værdier begrænset til

$$0,1,\ldots\ldots\ldots,15 ,$$

der altså nedskrives som

$$0,1,\ldots\ldots\ldots,9,A,\ldots\ldots\ldots,F.$$

En modstrid mellem på den ene side b og på den anden h+d lader den af størrelserne dominere, der ved udlæsningen giver det mindste antal betydende cifre (b er det maksimale, ikke nødvendigvis det effektive antal betydende cifre).

s_1= 0,1,2,4 el. 8,9,A,C

Trykformer.

Når man har specificeret tallagringen, står man overfor at afgøre, hvorledes man vil have tallene trykt, og her er mulighederne delt i 3 trykformer, der fastsættes ved

$$s_1 = 0 \text{ eller } 1,$$
$$s_1 = 2,$$
$$\text{eller } s_1 = 4 .$$

En bestemt side af skrivemaskinens virkemåde skal først omtales.

Ved at trimme udlæsningsprogrammet fastlægges bl.a. et vist antal anslag, som alle udføres ved enhver udlæsning med denne trimning, men som fra tal til tal kan være fordelt forskelligt mellem fortegn, komma,

cifre og mellemslag. Herom nærmere senere. 1ste anslag (altså eventuelt bare mellemslag) begynder fra en vis udgangsstilling af valsen, ofte fra et tabulatorstop eller efter vognretur (ny linie). Når vi i det følgende taler om, at f.eks. komma kommer under komma, skal det forstås på den måde, at kommaet ved hver udlæsning af et tal bliver slået i samme anslag, regnet fra udgangsstillingen (og at tallene på den måde i virkeligheden godt kan komme i samme vandrette linie blot fra forskellige tabulatorstop.)

De tre trykformer karakteriseres nu på følgende måde.

Trykform 0, $s_1 = 0$ eller 1.

Tallene udlæses som almindelige decimalbrøker, ægte eller blandede, med komma under komma:

- 138.571...
81.234...
- 0.011 903...

Dette sker både med $s_1 = 0$ og $s_1 = 1$, men ved denne sidste s_1-værdi afsluttes udlæsningen med tabuleringsordren*, d.v.s. ved trykning vil valsen, når tallet er trykt, føres frem til næste tabulatorstop, står altså eventuelt parat til trykning af et nyt tal.

Trykform 2, $s_1 = 2$.

Her udlæses tallene

enten på almindelig måde som ved trykform 0

eller som taldel og en eksponent for en 10-potensfaktor, altså (taldel) $\cdot 10^{\text{eksponent}}$

Eksponenten trykkes efter taldelen, og er et heltal (positivt eller negativt) med højst 4 cifre. Eksponenten 0 trykkes ikke, men der udføres mellemslag svarende til et mellemrum, fortegn og 3 cifre (nemlig som udlæsningen af en 3-cifret eksponent forløber; mellemslaget før fortegnet sløjfes ved 4-cifrede eksponenter).

Hvilken af de to muligheder, der bringes til udførelse, afhænger af det enkelte tals værdi, hvordan skal nu forklares.

---

* på hulstrimmelen vil komme et 3-tal uden femte hul, jfr. afsnit 6.4, hvilket medfører tabulering ved senere læsning af strimmelen.

Vi tænker os, at h, b og d er valgt. Er $b \leq h + d$ vil

1. Tal, x, for hvilke

$$10^{b-d-1} \leq |x| < 10^{h}$$

blive læst ud som ved trykform 0, d.v.s. uden eksponent, og her med konstant antal betydende cifre (b).

2. Tal for hvilke $|x| \geq 10^{h}$

blive læst ud som taldel og eksponent, med
netop b betydende cifre (altså konstant antal)
netop h heltalscifre i taldelen og positiv eksponent.

3. Tal for hvilke $|x| < 10^{b-d-1}$

blive læst ud som taldel og eksponent
netop b betydende cifre
og enten netop b-d heltalscifre, nemlig når $b-d \geq 0$,
eller netop d-b nuller efter kommaet, nemlig når $b-d<0$,
og i begge tilfælde negativ eksponent.

Er $b>h+d$ vil 2 eller 3 vise hvorledes trykningen bliver, idet antallet af betydende cifre nu er konstant lig h+d og der altid vil være netop h heltalscifre i forbindelse med den dertil svarende positive eller negative eksponent. Altså her vil kun tal der i forvejen har præcis h heltalscifre trykkes uden eksponent.

Eksempel:

7001.286...
-316857.85...
27.39051..
0.06603272..
-103.341..
-0.000173462..
0.000001034822..
-92430.112..

Vælges h=4, b=5 og d=3 , får man

7001.3
-3168.6 +2
27.391
66.033 -3
-103.34
-17.346 -5
10.348 -7
-9243.0 +1

Vælges b=4, b=5 og d=7 , får man

```
 7001.3
-3168.6          +2
   27.391
    0.066033
 -103.34
   -0.0017346    -1
    0.0010348    -3
-9243.0          +1
```

Man kan sige, at det karakteristiske ved denne udlæsning er, at de numerisk største tal, der bliver trykt uden eksponent, skrives med h heltalscifre, og at de numerisk mindste tal, der bliver trykt uden eksponent, skrives med b-d heltalscifre eller med d-b nuller efter kommaet.

Det vil hyppigt være netop disse to antal, altå h og b-d, man har fastsat, når man beslutter sig for trykformen, og først senere bestemmes en værdi for b og dermed for d (eller omvendt).

Vælger man $b-d = h \geq 0$, får vi en speciel virkemåde frem, nemlig at samtlige udlæste tal og taldele har h cifre foran kommaet og d decimaler. (Iøvrigt fås det samme blot $b-d > h$ eller udtrykt anderledes $b > h + d$).

Vi vil straks diskutere brugen af trykform 2.
Ved man ret nøje, hvor resultaterne rent numerisk vil ligge, dog uden at være helt sikker på sine vurderinger, kan denne trykform være bekvem. Den kan nemlig bringes til at lade alle forudsete resultater komme på sædvanlig form, mens resultater, der er numerisk større end forudset, udlæses med maksimal heltalsdel (d.v.s. h heltals-cifre) og en positiv eksponent. I disse tilfælde ville der ved de andre trykformer komme et udhop, f.eks. til kontroludskrift. Resultater, der er numerisk mindre end forudset, vil på lignende måde blive udlæst med negativ eksponent og første betydende ciffer placeret på en bestemt plads i taldelen.

Ved man derimod intet præcist om resultaterne, eller er de overordentlig stærkt svingende (eller måske bare alle numerisk meget store eller numerisk meget små), vil man sætte $b-d = h > 0$ og på den måde få en ensartet taldel og variationen i det væsentlige lagt over i eksponenten (der kan variere fra omkring - 1330 til omkring + 1330).

Trykform 4 , $s_1 = 4$.

Tallene udlæses som ved trykform 0, d.v.s. som decimalbrøker, ægte eller blandede, men med første betydende ciffer under første betydende ciffer, altså f.eks. således:

-138.571...
-81.2342...
1300.28...
1.86235...

Vi fremhæver allerede her, blot for at det ikke skal overses, at denne trykform kræver $h = b = d + 1$, men venter iøvrigt til side 15.24 med en nærmere forklaring af spørgsmål i denne forbindelse.

Ved at addere 8 til s_1 bibeholdes trykformen, men vi opnår, som det blev nævnt side 15.13, at trykprogrammet går udenom de halvceller, hvor 10-potenseksponenten p_1 og 2-potenseksponenten p_2 er placeret, d.v.s., der udlæses uden faktoren $10^{p_1} \cdot 2^{p_2}$, som dog står klar til senere brug.

Det kan volde noget besvær at få oversigt over sammenspillet mellem h,b,d og trykform, og vi skal derfor give en detaljeret, delvis grafisk beskrivelse af, hvordan disse ting griber ind i hinanden.

Forinden vil det dog være praktisk at omtale :

n

Er det tal, der skal udlæses, numerisk større end eller lig 10^h, må der et eller andet sted i programmet være overset en ting, begået en fejl eller lignende. Der kan være tale om virkelige kodefejl eller blot om, at resultaternes ekstreme værdier er galt vurderet.

Ved trykform 0 og 4 vil programmet i sådanne tilfælde hoppe ud til halvcelle n, hvor man eventuelt kan prøve at indkredse, hvad der er sket.

Som omtalt undgår man dette udhop ved trykform 2, hvilket er fordelagtigt, når man ikke har villet bruge tid til omhyggelige vurderinger af resultaterne eller lignende (men for n må alligevel indsættes et tal, f.eks. 0).

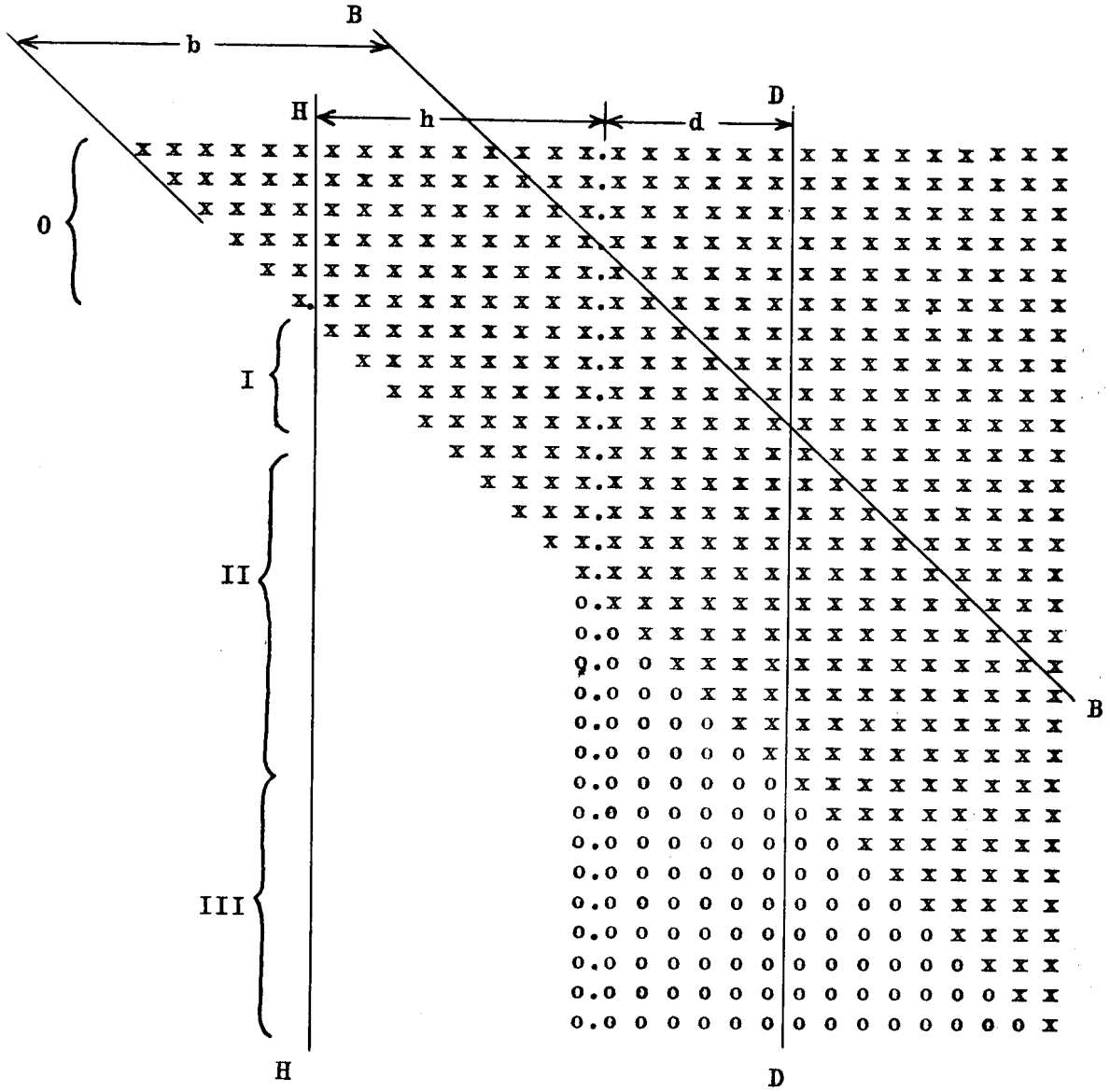
Ved sprængning af trimning kan man fra ordren i halvcelle n f.eks. administrere skrivning af asterisk og mellemslag; et 50-hop til 1761 vil derefter retablere indeksregistrene med tilbagehop til HS fra udlæseprogrammet.

Trykform 0 i forbindelse med valg af h, b, d og n.

Da h og d er begrænset til værdierne 0,1,.....,15, kan vi højst få udlæst et tal med 15 cifre foran eller 15 cifre efter kommaet.

I skemaet nedenfor vises, hvad det, taget rent numerisk, er for et værdiområde, vi på denne måde har mulighed for at udlæse.

Tallene er ordnet systematisk efter antallet af betydende cifre. Disse er vist som et x. For en ordens skyld nævner vi, at det altså højst er de 15 forreste af de betydende cifre, vi kan få udlæst.



Vi indtegner følgende linier:

den lodrette linie H, der afgrænser h cifferpladser foran kommaet

den lodrette linie D, der afgrænser d decimaler (efter kommaet)

den skrå linie B, der afgrænser de første betydende cifre.

Det vil være let i de enkelte tilfælde at placere disse linier, og det er klart, at de kan give alle oplysninger om, hvad det konkrete valg af h,b og d vil føre til.

Vi har valgt h = 9

b = 12

d = 6

og ser ved en betragtning af skemaet, at

i området 0 er heltalsdelen større end h tillader, d.v.s. vi får udhop til hac n

" " I er værdien af b for lille til, at vi kan få alle d(6) decimaler udlæst, d.v.s. vi har en udlæsning med konstant antal betydende cifre

" " II er værdien af d for lille til, at vi kan få alle b(12) betydende cifre udlæst, d.v.s. vi har en udlæsning med konstant antal decimaler.

" " III er d for lille til, at vi overhovedet får udlæst betydende cifre.

Man ser endvidere, at

i området I ville vi have fået nøjagtig den samme talmæssige udlæsning med enhver værdi d $\geq$ 6, når blot h og b fastholdes,

i " II ville vi have fået nøjagtig samme talmæssige udlæsning med b $\geq$ 12, når blot h og d fastholdes.

Bemærkninger:

1. Vælges, i modsætning til i eksemplet her, h større end b, f.eks. h = 14, vil kravet om, at det betydende antal cifre maksimalt skal være b, medføre, at tal, hørende til øverst i området I, (ændret så det svarer til, at h nu er 14), udlæses som

x x x x x x x x x x x x o o

x x x x x x x x x x x x o

der suppleres altså med nuller indtil kommaet, som ikke trykkes, men erstattes af et mellemrum (d.v.s. anslaget bevares).

2. Når d vælges lig 0, og man således kun udlæser heltal, vil ikke alene kommaet ikke trykkes, men anslaget (mellemrummet) er fjernet

3. Tallene, der udlæses, er alle afrundede; de begrænsende linier skal altså ikke tages som udtryk for en blot og bar bortskæring af cifre. (Ved tal svarende til øverste linie i område III kan vi altså trods alt få udlæst betydende cifre, nemlig 0.000001)

Trykform 2 i forbindelse med valget af h,b,d.

Selvom trykformen i virkeligheden altid arbejder på samme måde, fremtræder den, som det er forklaret, i to ret forskellige former; hvilken afgøres af om

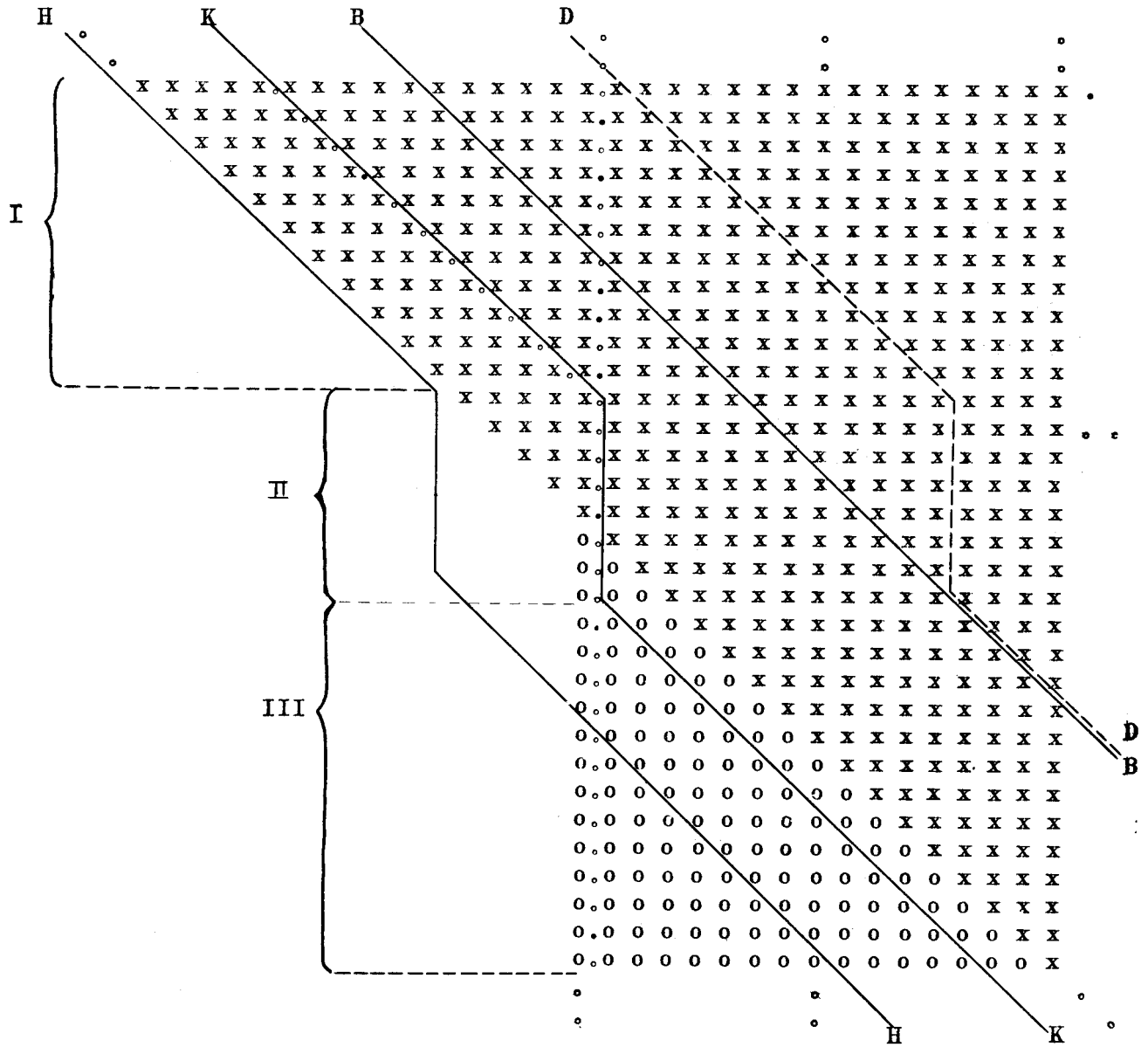
$h > b-d$, hvor man tilsigter en udskrivning af samme karakter som ved trykform 0,

eller

$h = b-d > 0$, hvor man får en udskrivning udelukkende i form af talpar med et fast antal heltalscifre.

Vi vil benytte samme skema som før, men må gøre os klart, at det talværdiområde, der kan udlæses med trykform 2, er så godt som ubegrænset, idet det går fra omkring 10^{1330} til omkring 10^{-1330}.

$h > b-d$ (eller $h + d > b$)



Vi trækker her følgende linier

en skrå linie B, som overalt afgrænser de b første betydende cifre

en knækket linie H, som følger første betydende ciffer indtil der er netop h heltalscifre eller d-b nuller efter kommaet, derefter går lodret nedad til der er netop b-d heltalscifre henholdsvis d-b nuller efter kommaet og derefter igen går på skrå i afstanden $h-(b-d) > 0$ fra første betydende ciffer.

Den knækkede linie D, er taget med; den bestemmes udfra d på en lignende måde, som H blev bestemt udfra h, og sammen med de andre linier illustrerer den, hvor i den samlede mængde anslag af heltalscifre, decimaler og mellemslag placeres (cifferpladser mellem B og D angiver mellemslag før eksponentens 5 anslag).

Vi har valgt $h = 5$, $b-d = -2$ og i forbindelse hermed

$b = 10$

$d = 12$.

Det er nu således, at

mellem de to linier H og B står de cifre, der udlæses,

dog, som forklaret ved trykform 0, først efter afrunding.

Iøvrigt er der vist en "kommalinie" K, nemlig gennem taldelenes kommaer. Ved udlæsningen er det disse kommaer, der trykkes, og de placeres under hinanden.

Det udlæste vil falde i tre grupper:

tal i område I, hvor antallet af heltalscifre er større end h, udlæses med netop h heltalscifre og b betydende cifre i taldelen samt positiv eksponent.

tal i område II udlæses som ved trykform 0 (men med mellemslag for eksponenten 0) og konstant antal betydende cifre (b); dette er udlæsningens hovedområde.

tal i område III, er numerisk for små til at blive udlæst uden eksponent og får enten b-d heltalscifre foran eller d-b nuller efter kommaet i taldelen samt negativ eksponent. Antallet af betydende cifre er b.

Udhopsadressen n bliver aldrig aktuel, men som sædvanlig må man indføre et tal på denne plads.

Bemærkninger:

1. Vælges b = h vil alle tal i område I få en taldel, der er et heltal; disse udlæses uden at kommaet trykkes, dette erstattes af et mellemrum. Er yderligere h > b, vil de h-b sidste heltalscifre være nuller.

2. Vælges d = 0, vil alle taldele (i område II: alle tal) være heltal, og her er kommaet udeladt også som anslag (vi får ikke mellemrum).

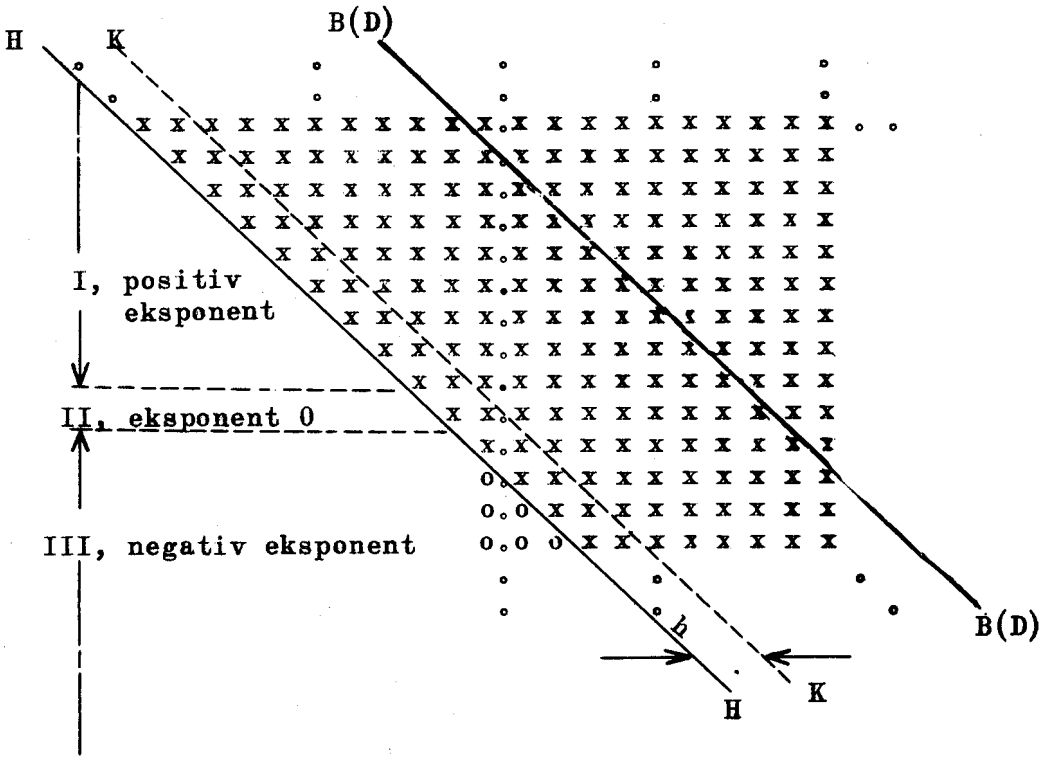
$\underline{h=b-d \geq 0}$ (d.v.s. h+d = b), iøvrigt samme udlæsning når blot $h+d \leq b$

Linierne H,B og D bliver parallelle, skrå linier (og D bliver sammenfaldende med B).

Med h = 2

b = 10, og følgelig d = 8,

ser skemaet således ud:



De udlæste cifre står mellem H og B, og det er igen kommaerne i linien K, der trykkes, og de placeres under hinanden.

Den udlæste taldel har et <u>fast antal heltalscifre</u> h og et <u>fast antal decimaler d</u>.

Selvfølgelig gælder også her, at udhopsadressen n ikke bliver aktuel

Bemærkning:

Vælges d = 0, altså h = b, betyder det, at alle tal udlæses som heltal med den tilsvarende eksponent. Her er kommaet udeladt også som anslag.

Trykform 4 i forbindelse med valget af h, b, d og n

Krav: h = b = d + 1.

Her er det talværdiområde (taget numerisk), der kan udlæses, igen begrænset og på samme måde som ved trykform 0.

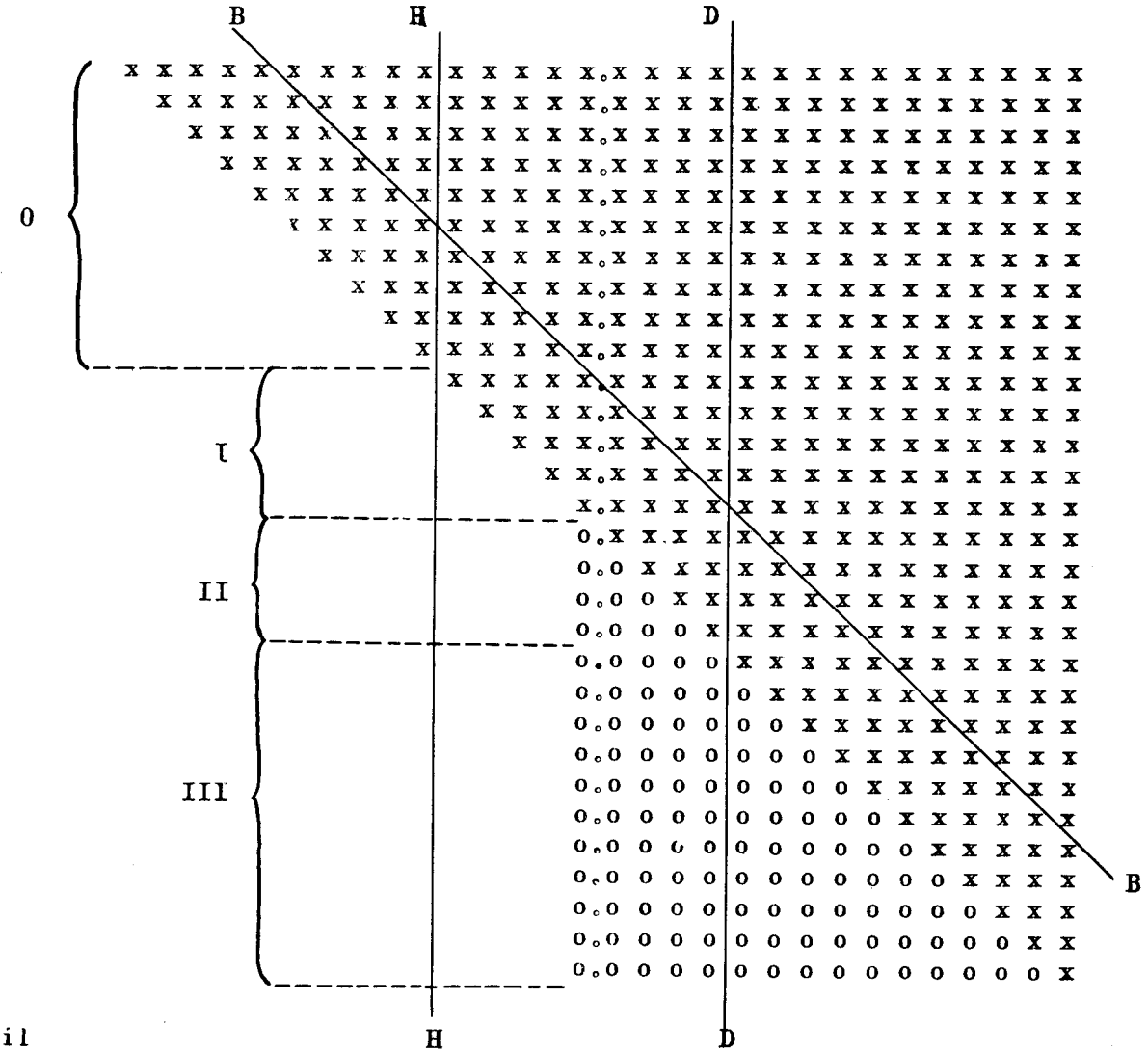
Vi trækker følgende tre linier:

en skrå linie B, der afgrænser de første betydende cifre

en lodret linie H, der afgrænser h heltalscifre foran kommaet

en lodret linie D, der afgrænser d decimaler.

Vælges h = b = 5, d = 4 bliver liniernes placering i skemaet:



Her vil

tal i område 0 give udhop til hac n

tal i område I give den udlæsning der tilsigtes, hvor første betydende ciffer virkelig kommer under første betydende ciffer

tal i område II give en udlæsning, hvor heltalsnullet kommer på første betydende ciffers plads, og derefter følger kun 4 decimaler, eks:

11825	tilhører område I
4803.1	
8.0312	
-103.01	
0.7703	tilhører område II
0.0004	

tal i område III give en udlæsning uden betydende cifre (0.0000, eventuelt 0.0001). Antallet af betydende cifre er igen overalt b

Bemærkninger:

1. Ved tal, der hører ind under første linie i område I (heltal), trykkes kommaet ikke, men erstattes af mellemrum.
2. Det vil være muligt v.h.a. parameteren s_2 at fjerne nullet foran kommaet i ægte brøker, og således få et betydende ciffer mere i disse tal, når samtidig d vælges lig h(b).
   Eks.: -103.01
   .77026
   -.00043 , hvor altså h = b = d = 5.

s_2 = 0,1,2 eller 3

Med denne parameter kan man fjerne dels fortegnsanslaget for positive tal, dels nullet foran kommaet i ægte brøker eller begge dele. D.v.s., der er ikke tale om blot at hindre, at disse tegn slås, men direkte om at spare de anslag, hvori de skulle komme.

Dette har betydning, når plads på valsen eller tid (maskintid) skal spares, og når der er sørget for, at der ikke derved kan opstå vanskeligheder ved læsningen af de trykte tal.

$s_2 = 0$ Her udlæses som normalt, d.v.s. med anslag for + tegnet og heltalsnul

$s_2 = 1$ Anslaget for + er bevaret
" " heltalsnullet er sløjfet.

Denne variant har kun betydning for trykform 4, hvor den som tidligere forklaret f.eks. giver udlæsningen

-103.01
.77026
-.00043

Ved andre trykformer opnås det samme med h = 0 (h kan ikke være mindre end 1 ved trykform 4!).

$s_2 = 2$ Anslaget for + sløjfes.
" " heltalsnullet bevares.

Denne variant bruges, når udlæsningen udelukkende gælder positive tal.

$s_2 = 3$ Anslaget for + sløjfes.
" " heltalsnullet sløjfes.

	+ tegn anslag	ikke + tegn anslag
heltalsnul	0	2
ikke heltalsnul	1	3

Eks.:

tallene 27.51332
-8.10366
-0.00727
0.91224

bliver altså trykt som

27.51332
-8.10366
-.00727
.91224

hvor trykform 0 og en passende trimning tænkes anvendt.

k=0,A,B,C,D eller E.

Når maskinen er indstillet til at perforere under udlæsningen, får man med k = 0 en strimmel, hvorfra en skrivemaskine (Flexowriter) kan udskrive på helt sædvanlig måde.

Derimod kan man ikke lade talresultaterne læse direkte ind i DASK fra denne strimmel. Dette skyldes, at +, - og de forskellige kommaer, som hører til et tal, der skal indlæses, ikke er stanset eller i hvert fald ikke stanset i den kode, der kræves til indlæsningen.

Imidlertid kan man få suppleret med det, der mangler (fortegn, komma og tal - slut-symbol), simpelthen ved at maskinen stanser både de huller skrivemaskinen kræver og de, som DASK kræver. En sådan "dobbeltstansning"vil maskinen udføre, når k sættes lig det af cifrene

A,B,C,D,eller E,

der svarer til det komma-symbol tallet, der skal udlæses, skal være forsynet med for atter at kunne læses korrekt ind.

Man må i denne forbindelse være klar over, at tal, som f.eks. 10^{P}.C(AR) selvfølgelig ikke (altid) kan indlæses som DASK-tal, men nu må behandles f.eks. som flydende tal. Lignende forhold gør sig gældende ved heltal udlæst med faktorer.

Antallet af anslag ved de forskellige trykformer.

Trykform 0 bruger 2 + h + d anslag,

fordelt på følgende måde:

1) mellemslag (fra linie H til første ciffer)
2) fortegn d.v.s. minus eller mellemslag
3) heltalscifre, henholdsvis heltalsnullet ved ægte brøker
4) komma, altid anslag nr. 2+h
5) decimaler (indtil den begrænsende linie, d.v.s. B eller D).
6) mellemslag (hvis den begrænsende linie er B, indtil linien I

Med $s_1 = 1$ spares mellemslagene under 6), og i stedet udføres tabuleringsordren.

Det vil være let at se, hvordan ovenstående skal ændres, hvis b er mindre end h.

Med d = 0 spares 4).

Trykform 2. bruger 7 + h + d anslag :

1) eventuelt mellemslag (fra linien H til første betydende ciffer).
2) fortegn
3) heltalscifre eller heltalsnullet
4) komma, altid anslag nr. 2 + h
5) d decimaler (heri eventuelt mellemslag fra linien B til D).
6) et mellemslag (dog ikke hvis eksponenten er 4-cifret)
7) eventuelle mellemslag, hvis antallet af cifre i eksponenten er < 3
8) fortegn
9) eksponenten

Med d = 0 spares 4).

Trykform 4 bruger 2 + b anslag:

1) fortegn
2) heltalscifre eller heltalsnullet
3) komma
4) decimaler

Bemærk: mellemslag forekommer ikke.

Valsen har ialt 140 anslag.

Ved at sætte $s_2 = 0$ bruger udlæsningen den plads, der svarer til de antal anslag her er anført.

Ved at sætte $s_2 = 1$ eller $s_2 = 2$ spares eventuelt plads svarende til 1 anslag, og ved at sætte $s_2 = 3$ spares eventuelt plads svarende til 2 anslag.

Nøjagtigheden under udlæsning.

Når det drejer sig om at udlæse C(AR) (tallagring 1 eller 3) og $|C(AR)| \geq \frac{1}{10}$, vil det udlæste svare til C(AR)'s 39 binaler efterfulgt af (binære) nuller.

Er $|C(AR)|$ derimod $\leq \frac{1}{10}$, sker der under udlæsningen en omregning og herunder hober afrundingsfejl sig op. Dette kan bevirke, at kun de 11 første betydende cifre svarer til det C(AR), koderen vil have udlæst; ved tallagring 3 kan endog kun de 10 første betydende cifre betragtes som helt sikre.

Af samme grund vil man ved udlæsning af C(FAR) kun kunne regne med, at de 11 første betydende cifre svarer til FAR1's 39 binaler(taldelen x').

Resumé.

Som fremgangsmåde kan foreslås:

1. tallagringen specificeres, og t-værdien nedskrives,
2. man bestemmer sig for trykformen, s_1-værdien nedskrives,
3. de numeriske forhold bestemmer h,b og d, og man kan nu overse om der skal spares plads, d.v.s. hvilken værdi man skal give s_2,
4. n vælges; ofte vil man her antagelig nøjes med at hoppe til en stopordre (som man bør have present under kørslen).
5. k vælges.

De to pseudoordrer er herefter gjort klar og tilbage bliver, om tallagringen kræver en faktor $10^{p_1} \cdot 2^{p_2}$ ved udlæsningen, således at man må indføre eksponenterne p_1 i 1607 og p_2 i 1565 ved særlige ordrer, to for hver eksponent.

Kombinationer af trimninger.

Vi vil ved et simpelt eksempel vise, hvordan man på forskellige måder kan indrette en trykning, så den veksler mellem udlæsninger, hvor der kræves en faktor, og udlæsninger, hvor der ikke kræves faktorer.

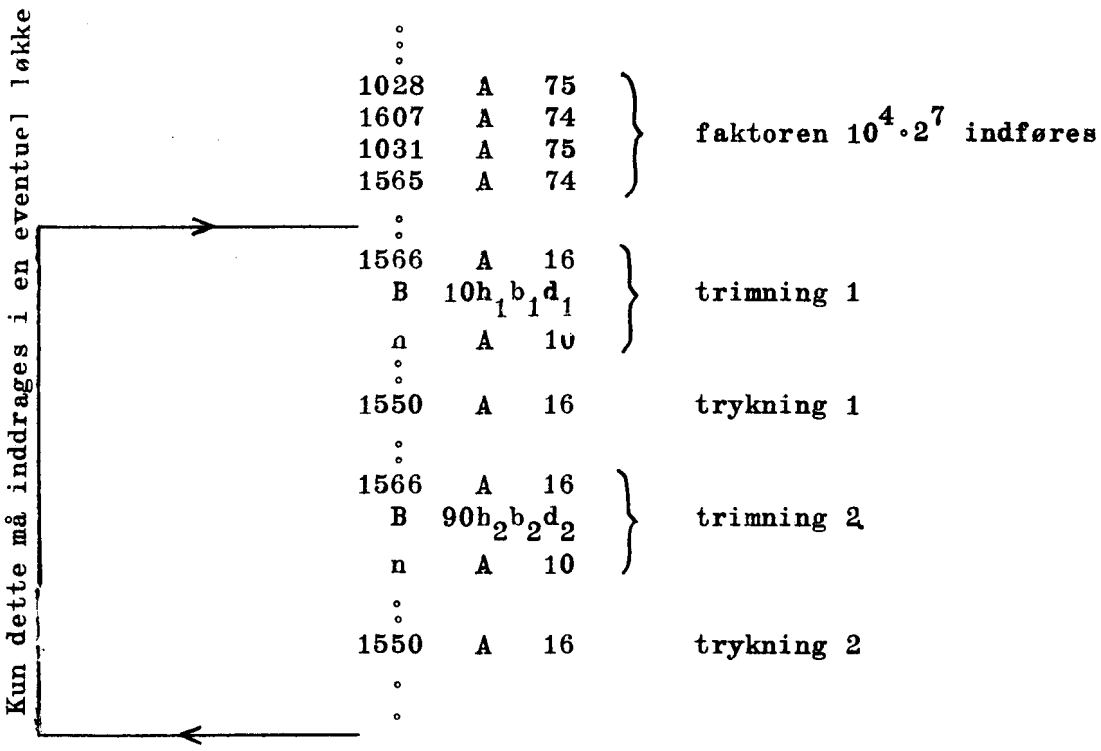
Vi tænker os, at der først skal trykkes et tal med f.eks. faktoren $10^4 \cdot 2^7$, og derefter et tal uden faktor.

1. Kun sædvanlig trimning anvendt

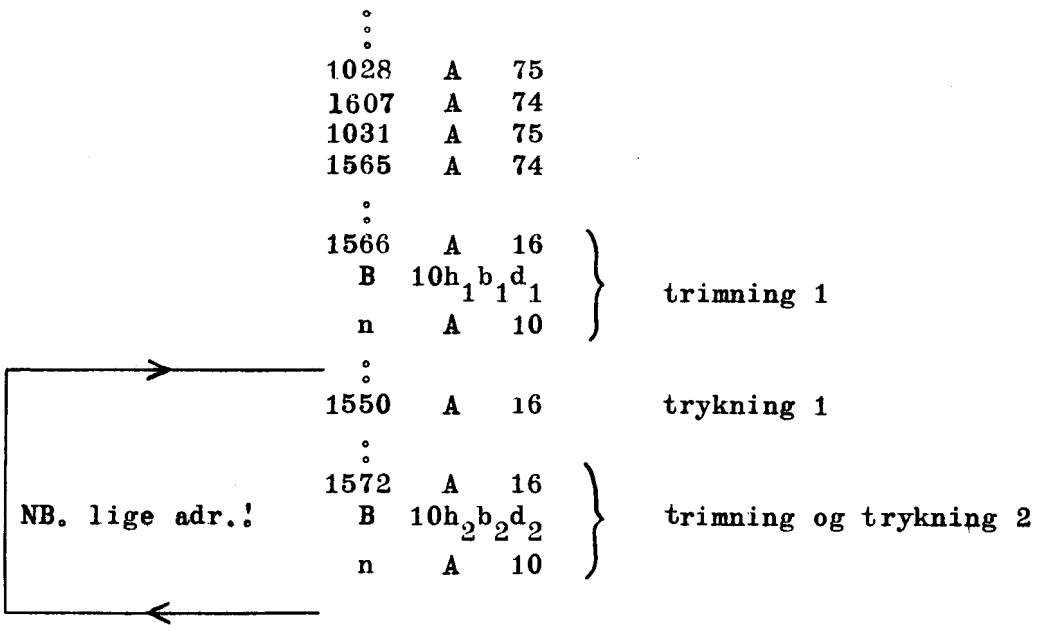
Dette må inddrages i en eventuel løkke

```
.
.
1566   A   16
B  10  h1 b1 d1      } trimning 1
  n    A   10
.
.
1028   A   75
1607   A   74
1031   A   75
1565   A   74        } faktoren 10^4 . 2^7 indføres
.
.
1550   A   16        trykning 1
.
.
1566   A   16
B  10  h2 b2 d2      } trimning 2
  n    A   10
.
.
1024   A   75
1607   A   74
1565   A   74        } faktoren 10^0 . 2^0
.
.
1550   A   16        trykning 2
.
.
```

2. Almindelige trimninger og trimning hvor en faktor overspringes.



3. Almindelig trimning i kombination med den specielle indhopsadresse 1572.



4. Tænker vi os, at tallene, der skal trykkes uden faktor, er argumenterne x_0 $x_0' + \Delta x, \ldots, x_0 + n\Delta x$ kan dette gøres bekvemt med standardtrimningen i 1562.
Lad hec 80 være reserveret til argumentet og fra start indeholde dettes begyndelsesværdi, x_0, og C(78) = Δx.
Det løbende argument opbygges da simpelthen ved ordrerne

```
78   A  40
80   A  06
```

```
:
1028   A   75   }
1607   A   74   }  faktoren 10^4 · 2^7 indføres
1031   A   75   }
1565   A   74   }
:
1566   A   16   }
  B  10h1b1d1   }  trimning 1
  n    A   10   }
:
1550   A   16      trykning 1
:
  78   A   40   }  argument x0+nΔx ændret til
  80   A   06   }        x0+(n+1)·Δx.
1562   A   16
:
```

Særlige indhop i udlæseprogrammet.

Ved udskrift på perforator ønskes ofte en kontrol af korrekt hulning. Dette fås ved før og efter udskriften at hoppe på 16-ordre til henholdsvis hac 1747 og hac 1739. Ved hoppet til 1747 perforeres 10 cm blank strimmel og en særlig kontrolsumcelle i udlæseprogrammet nulstilles. Ved hoppet til 1739 perforeres "vogn retur", stopkombination og et F, derefter udlæses 5 sedecimale cifre fra kontrolsumcellen, hvori der under udlæsningen af talmaterialets cifre er dannet disses sum modulo 2^{20} før de perforeres; endelig perforeres 10 cm blank strimmel.

Ved denne procedure kan strimmelen ved et særligt kontrolprogram indlæses i DASK til kontrolsummation hvorved eventuelle perforatorfejl kan afsløres.

KAPITEL 16

Programfejl og kontrolprogrammer.

Kodning for en automatisk regnemaskine stiller krav om næsten fuldkommen akkuratesse. For flertallet af de symboler, der gennem læsemekanismen leveres til en regnemaskine under løsningen af en opgave, gælder det, at en fejl vil bevirke, at maskinens svar bliver fejlagtige. For mange fejl vil det endda gælde, at de vil få maskinen til at løbe helt vild eller give fuldkommen falske svar.

I betragtning af den menneskelige organismes uegnethed, når det gælder stadig og fuldkommen akkuratesse, er det ikke mærkeligt, at de krav, som stilles til programmet for en automatisk regnemaskine, kun sjældent kan opfyldes ved det første forsøg. Det må betragtes som et faktum, at flertallet af de koder, som for første gang prøves på en maskine, indeholder fejl.

Yderligere viser det sig ofte vanskeligt at lokalisere fejl, selv hvor det er tydeligt, at programmet arbejder forkert.

Det må følgelig være en vigtig opgave for enhver regneinstitution at give koderne hjælp ved bekæmpelsen af fejl. Disse bestræbelser må gå ud på dels gennem erfaren vejledning at hjælpe koderne til at undgå fejlene, dels gennem særlige metoder at tillade maskinen selv at hjælpe koderne til at finde fejl med et ringe forbrug af maskinens tid. Som et bidrag til disse bestræbelser skal der i det følgende gives en række praktiske forslag, som erfaringsmæssigt kan bidrage til mere fejlfri kodning, og der skal beskrives visse metoder til fejlfinding ved hjælp af dertil udviklede programmer for DASK. Den tredie mulighed, at sætte maskinen til at udføre selve kodningen, skal derimod ikke berøres, trods de store muligheder der ligger heri.

Forslag til en arbejdsgang ved kodningen.

1. Numerisk analyse. Udgangspunktet for enhver numerisk beregning er en reduktion af de forekommende processer til dem, som maskinen kan udføre. De derhen hørende principper er for omfattende til, at der kan siges noget almindeligt derom. Kun skal opmærksomheden henledes på tre forhold.

1) I en vis forstand må det hævdes, at maskinen umiddelbart kan udføre enhver operation, som er udformet som en biblioteksse-kvens. Derfor må man straks ved planlæggelsens begyndelse rådføre sig med biblioteket for at fastslå, hvilke dele af programmet der ikke behøver at analyseres nærmere.

2) Det vil ofte være af stor betydning for den effektivitet, hvormed et større problem bliver løst med en hurtig elektronisk regne-maskine, at man fra første færd har formuleret den rigtige målsæt-ning. Ved løsning af regneproblemer ved hjælp af manuelle metoder kan man ofte udsætte beslutninger om arbejdets videre forløb til, efter at de indledende processer er i gang. Ved en maskine som DASK foregår indlæsning og udskrivning af data forholdsvis langsomt, og det er derfor afgørende, at man fra første færd stiler mod det ende-lige svar på sit problem i nøjagtig den form, det ønskes.

3) Det er ønskeligt allerede på et tidligt tidspunkt at over-veje de mulige gennemgribende kontroller, som problemet tillader. Som eksempler skal nævnes: En matrixinversion kontrolleres fuld-stændigt ved en inversion af resultatet og en sammenligning med ud-gangsmatricen. En løsning til en differentialligning kan kontrolle-res grundigt ved en gentagen (evt. sideløbende) integration med en anden integrationsmetode eller ved brug af et andet interval, eller andre skalafaktorer. De fleste tabelleringer kontrolleres effektivt gennem differensdannelse.- Det vil ofte være muligt at inkludere så-danne kontroller i selve programmet, og maskinen vil selv kunne be-stemme, hvorvidt den fornødne overensstemmelse er til stede. Dette er fordelagtigt, også fordi det sparer den tidsrøvende udskrivning, som ville være nødvendig ved en dobbelt beregning.

2. <u>Disposition af programmet</u>. Det er altid ønskeligt så vidt muligt at benytte bibliotekets sekvenser, både fordi man herved spares for det pågældende arbejde, dels og især fordi man da er sikret mod fejl i den pågældende del af programmet. Under dispositionen af pro-grammet må man derfor først og fremmest granske specifikationerne for bibliotekets sekvenser og udvælge dem, som kan bruges.

Hertil kommer, at det ofte er muligt at udforme visse dele af programmet som lukkede sekvenser. Dette må i høj grad anbefales af følgende grunde:

a) Den pågældende del vil kunne efterprøves separat, hvilket gør den eventuelle fejlsøgning lettere.

b) Den pågældende sekvens vil kunne indlemmes i biblioteket til glæde for andre brugere.

Det er klart, at hver ny særligt konstrueret sekvens vil kræve et særligt prøveprogram til kontrol af dens korrekthed. Disse underproblemer må naturligvis betragtes ganske som andre programmer og udarbejdes efter de her fremstillede synspunkter.

Når biblioteksekvenserne på dette stadium er udvalgt, vil det også være muligt at disponere over de 7 frie etikettemærker 9, A, ... F, efter de principper der er angivet i kapitel 9. I øvrigt står man sig altid ved at anbringe oplysningerne i lageret i en naturlig orden. Derfor bør man ikke på dette stadium begynde at tildele nogen programdel sin adresse. Det medfører blot, at man senere må ændre den, hvilket igen medfører en høj risiko for fejl.

3. Egentlig kodning. For hele planlægningen, men især for den egentlige kodning, gælder det, at man bør begynde med de groveste træk, og at man først bør begynde at fylde detaljer ind, når alle hovedtrækkene er fastlagt og så grundigt gennemtænkt, at det er nogenlunde sikkert, at de ikke behøver at ændres.

3a. Rutediagrammet. Dette er et vigtigt middel til at få overblik over den nøjagtige rækkefølge, hvori programmets enkelte dele skal arbejde. Det kan udformes efter de principper, der er omtalt i kap. 11, eller man kan vælge med det samme at nedskrive det i kun een dimension, for således med det samme at opnå en entydig korrespondance med ordrenes eendimensionale placering i lageret.

3b. Skitsekodning. På grundlag af det grundigt gennemtænkte rutediagram følger en første skitsemæssig kodning, uden nedskrivning af egentlige adresser, undtagen hvor der er tale om henvisninger til talsekvenser eller standardsekvenser, hvis placering er fastlagt gennem etikettemærker. Derimod kan operationsdele og index- og etikettemærker i reglen nedskrives fuldstændigt. Som vejledning og idégiver ved dette arbejde er biblioteksekvenserne hovedkilden. Foruden sine andre vigtige funktioner tjener biblioteket også det formål at gøre DASK-kodernes bedste erfaringer tilgængelige for studium.

3c. Kontrol af skitsekoden. Det må betragtes som meget vigtigt, at allerede skitsekoden underkastes en meget omhyggelig gennemgang og efterprøvning, fordi rettelser, specielt omplaceringer og indføjninger af ordrer, endnu meget let kan foretages. Det anbefales, at man gennemgår alle ordrer flere gange, og hver gang særligt hæfter opmærksomheden ved eet bestemt forhold, hellere end at man forsøger at overse alle forhold på een gang. Som eksempler kan nævnes:

a) Indholdet i AR og MR følges nøje under programmets gang, specielt med henblik på nulstillinger.

b) Konferering af hver enkelt operationsdel med ordrelistens specifikationer.

c) Gennemgang af tællere, specielt at de tæller det rette antal gange.

I det hele bør man bestræbe sig på at se programmet fra forskellige synspunkter. De fleste fejl skyldes det velkendte forhold, at man, når man een gang har reageret fejl, senere i samme situation er stærkt tilbøjelig til at begå den samme fejlreaktion.

På dette stadium bør man også forvisse sig om, at de anvendte bibliotekssekvensers specifikationer er fulgt.

3d. Fuldstændig nedskrivning af program. Skitseprogrammet vil som oftest være lidet overskueligt på grund af rettelser, og den endelige nedskrivning indledes derfor gerne med en afskrivning. Forud herfor bør man foretage en omhyggelig afmærkning af skitseprogrammet med pile og henvisninger, og ganske særligt en grundig, umiskendelig overstregning af alle fejl. Der er intet så ærgerligt som tankeløst at arbejde videre med en serie symboler, som for længst er kasseret. Afskriften foretages fortrinsvis på programark, som er forsynet med cellenumre, herved undgås optællingsfejl. Først på dette stadium indføres de fuldstændige adresser og endnu kun med blyant.

3e. Henvisninger. Som sidste skridt indføres nu krydshenvisninger efter det dobbelte bogholderis princip: Der skrives en liste over adresserne på alle de celler, hvorpå programmet opererer, og ved hver adresse skrives nu adressen på alle de ordrer, som refererer dertil. Ved variable ordrer skrives henvisningen blandt noterne. Dette vil især bidrage til, at utilsigtede dobbelte anvendelser af arb.-celler bliver opdaget. Desuden er krydshenvisningerne meget nyttige, hvis det senere, trods al omhu, skulle vise sig nødvendigt at indføje nye ordrer.- Først efter denne kontrol indføres alle adresser med blæk.

Der skal på dette sted ikke mangle en opfordring til at nedskrive alle symboler tydeligt. Den tid, der bruges hertil, er i alle tilfælde ganske ringe i sammenligning med den samlede kodetid.

4. Planlægning af prøvekørsel. Efter afslutningen af udarbejdelsen af programmet efter ovennævnte retningslinier må opmærksomheden rettes fremad mod den mulighed, at programmet ved en prøvekørsel alligevel skulle svigte. I så fald bør man nemlig være parat til på den mest effektive måde at udnytte maskinens mulighed for at hjælpe ved lokaliseringen af fejlen.

Den første forudsætning for, at dette kan ske på en velorganiseret måde, er, at man har gjort sig ganske nøje klart, hvorledes programmet bør virke både med hensyn til indlæsning og udskrift, specielt bør trykte resultater straks kunne sammenlignes med korrekte værdier.

Videre er det klogt allerede på forhånd at forberede et af de aktive fejlsøgningsprogrammer, således at det umiddelbart kan bringes i anvendelse.

5. Programetiketterne kan nu umiddelbart nedskrives.

6. Hulningen af det samlede program bør altid foretages to gange, fortrinsvis af to forskellige personer, og resultaterne sammenlignes for at fejl kan elimineres.

7. Prøvekørsel. Ved prøvekørslen vil man først prøve programmet på ganske normal vis. Som eksempler skal nævnes et par muligheder for fejlagtige programreaktioner og rimelige fremgangsmåder til at sikre data til brug ved den senere lokalisering af fejlen.

7a. Indlæsning standser før tiden. Notér på hulstrimmelen, hvor indlæsningen standsede. Ferritudskrift. Stoppet skyldes enten forkerte programetiketter eller utilsigtet indlæsning i indlæseprogrammets celler.

7b. Programmet standser efter indlæsningen eller snurrer rundt i en lukket løkke. Kontroludskrift af registre. Herefter fornyet kørsel med overvågningsprogram.

7c. Programmet opfører sig korrekt, men giver numerisk fejlagtige resultater. Kontrolprogram med numerisk overvågning.

8. Fejllokalisering. Det materiale, som er vundet ved prøvekørslen, bør gennemgås med stor omhu, hvert eneste symbol bør konfereres, selv efter at en fejl er fundet - der kunne jo være flere!

9. Fejlstatistikken. Alene et kendskab til de fejl, som hyppigst er forekommet ved andre koderes arbejde, må kunne bidrage til at reducere fejlenes antal. Det foreslås derfor, at koderne i en vis udstrækning noterer sig typerne på de fejl, som de har begået. Det vil sikkert være praktisk forsøgsvis at indføre følgende kategorier af fejl:

A) Operationscifferfejl knyttet til ordrevarianter.
B) Andre operationscifferfejl.
C) Indexmærkefejl.
D) Etikettemærkefejl.
E) Egentlige adressefejl.
F) Adressefejl indført gennem ufuldstændige rettelser på et sent tidspunkt.
G) Programetikettefejl.
H) Fejlagtig anvendelse af indlæseprogrammets konventioner.
I) Fejlagtig anvendelse af biblioteksekvenser.
J) Andre fejl, herunder fejlagtig numerisk analyse.

Det er klart, at såfremt en optælling af fejl skal blive af værdi som vejledning, må den omfatte de antal, som har optrådt i afsluttede programmer, d.v.s. fejl, som er fundet efter påbegyndelsen af den første prøvekørsel, indtil programmet har arbejdet fuldstændig korrekt.

Kontrol- og hjælpeprogrammer.

I normalleje 1 indgår et kompleks af hjælpeprogrammer til brug ved indkøring. Disse programmer er alle lagrede på tromle og kan manuelt kaldes frem på indlæseprogrammets plads, således:

1. Maskinen stoppes (kodet eller manuelt).
2. Maskinen skal stå på gang.
3. Den aktuelle ordre må ikke have formen n,B,35, n,C,55, n,D,75 eller n,D,16 + varianter.
4. En af nøglerne "Kontroludskrift enkel" eller "Kontroludskrift Etiket" betjenes, herved sker følgende:

Kontroludskrift enkel.

På skrivemaskinen udskrives:

vr. C(KR) (dec.adr.) med evt. etikettemærke, C(ASOP) (dec.adr.,I,(q)OP), * for spild, -(minus) for C(AR)<0, 0 for C(AR) = 0, 1 for C(AR) = -1, C(IRB), C(IRC), C(IRD), (som 3 adr.).

Derefter stop med hop tilbage til den aktuelle ordre med alle registre retableret, dog ødelægges evt. tidligere valg af ydre enhed. 1. kanal af indlæseprogrammet kaldt ind.

Kontroludskrift Etiket.

På skrivemaskinen skrives:

vr. C(KR) vr. E *.

Derefter kan ved tryk på en af trykknapperne AR_{0-39} vælges et kontrolprogram (KP), hvis nummer (= nummeret på den indtrykkede knap) udskrives. Eksisterer det valgte KP ikke, udføres påny vr. E *. Klar til fornyet KP-valg.

Eksisterer det valgte KP, tilføres dette derefter de fornødne informationer ved hjælp af trykknapperne AR_{0-19}.

Informationerne kan være:

1) ordrer eller dele af ordrer.
   a) n
   b) n, I
   c) n, I, (q)
   d) n, I, (q), OP

   Formerne a - c afsluttes med "19" (delinformation slut).

2) Tal.

a) B efterfulgt af 1 - 10 sed. cifre.

b) dec. tal efter sædvanlig konvention, dog uden eksponent-del og uden enkeltstående A. Maksimalt 11 cifre. "Slut A" erstattes af "19".

Første information skal være fra 1) og tal kan kun omfatte een talværdi og vil altid være sidste besked til KP.

Enhver information udskrives på skrivemaskine, efterhånden som den tilføres:

AR_{0-15} som cifrene 0 - F.

AR_{16-18} som F1, F2, F3, dog kun dersom de trykkes som 1. ciffer i 1. information, i modsat fald giver de desuden vr. E*. Klar til nyt KP-valg.

AR_{19} har specielt funktion som afslutningssymbol:

1) Efter E *: * udhop som efter "Kontroludskrift enkel".

2) Efter tal og ufuldstændige ordrer: mlr. (delinformation slut).

3) Efter fuldstændige ordrer og AR_{19}: * start KP.

AR_{20-39} som vr. E * nr. (nyt KP valgt).

Som 1. informations første cifre har cifrene A - F3 speciel funktion, idet de anvendes til at adressere:

A	:	AR
B	:	IRB
C	:	IRC
D	:	IRD
E	:	KR
F	:	MR
F1	:	FMD
F2	:	FAR
F3	:	FMR
AE	:	AR, IRB, IRC, IRD og MR
FAE	:	FMD, FAR, FMR.

Kontroludskrift kan udføres efter stop hop fra indlæsningsprogrammet, der hoppes da ud fra KP, som om man kom fra den ordre, hvortil der blev beordret hop fra indlæseprogrammet.

Som eksempler på kontrol- og hjælpeprogrammer kan nævnes:

Fremkaldelse af Normalleje 1 fra tromlen. Kontrolprogram nr. 0

Funktion: Nulstiller ferritlageret, kontrollerer checksummen af Normallejets tromlekanaler (udskrift ved fejl), kalder faste del og trykprogrammet ind, nulstiller MD, AR, MR, IRB, IRC og IRD og stopper med hop til 1987 (sedecimalt 7C3). Ved start: indlæsning.

Nødvendige information: ingen.

Sumkontrol af perforatorstrimmel. Kontrolprogram nr. 4

Funktion: Summerer cifrene på en perforatorstrimmel fremstillet med Normallejets trykprogram og sammenligner summen med den sedecimale kontrolsum der følger efter. Kontrolsummen er kendetegnet ved et forudgående "F", stop undervejs betyder derfor, at der er optrådt et F.

Hvis summen stemmer udskrives * og programmet går til nyt kontrolprogram.

Hvis summen ikke stemmer, udskrives sedecimalt først den på strimmelen hullede og derefter den ved indlæsningen dannede sum og programmet går til nyt kontrolprogram.

N.B. Perforatorstrimmelen må ikke indeholde bogstaver.

Nødvendig information: Ingen. Perforatorstrimmel skal være sat i strimmellæseren før der afgives klarsignal til kontrolprogram.

Ferritudskrift på ordreform. Kontrolprogram nr. 20

Funktion: Udskriver et opgivet område af ferritlageret på ordreform, 10 ordrer pr.linie. Hver linie begynder med adressen på 1. ordre på linien. Hvis en eller flere linier er nul udskrives de ikke, men der kommer en ekstra vognretur. Der indledes med blank strimmel og afsluttes med stopkombination, blank strimmel.

Nødvendig information: 1) ingen: programmet udskriver da ferritlageret fra 0 - 1535.

2) een adresse: programmet udskriver da ferritlageret fra 0 - 1535 med

etiketter til fornyet indlæsning og afslutter strimmelen med stop, hop til den opgivne adresse efterfulgt af checksum, der dækker området fra 0 til sidste udskrevne ordre.

3) to adresser: programmet udskriver da ferritlageret fra 1. adr. til sidste adr.

4) tre adresser: de to første adresser behandles som under 3) den sidste som under 2).

Efter udskriften går programmet til nyt kontrolprogram.
NB. Skrivemaskin-perforatoromskifteren skal normalt stå i stilling "0". De manuelle operationer kommer da på skrivemaskine, selve udskriften på perforator. Ønskes - undtagelsesvis - udskrift på skrivemaskine kan omskifteren stilles i "S + P". Der fås da stop før og efter udskriften, AR nulstilles og skrivemaskinen startes ved (evt. gentagne) tryk på +.

Udskrift af fra- og tilhops adresser med sortering: kontrolprogram nr. 28
uden sortering: kontrolprogram nr. 29

Funktion: Programmet overvåger et andet program ordre for ordre og udskriver alle - eller visse - hop der udføres, med et hop pr. linie, først frahopsadressen derefter tilhopsadressen. Hop indenfor eller fra normallejet (KR > 1535) udskrives ikke. Typografiske operationer, der udføres af det program, der overvåges udskrives som det sidste sedecimale ciffer i ordrens effektive adresse efterfulgt af *. Tal udskrives understregede.

Dersom det sidst udskrevne hop udføres flere gange før der skal udskrives et nyt, skrives antal gange det er udført umiddelbart før vr. til den næste hopudskrift (bemærk, tæller man f.eks. til 37 udføres normalt 36 hop).

Alle hopadresser $\leq$ 1535 udskrives med etikettemærke.
Dersom q = 8 eller E_q = 0 udskrives adressen dog absolut

(NB. Etikettemærket beregnes udfra etikettecellernes aktuelle indhold, og behøver derfor ikke at svare til de under indlæsningen benyttede, er man i tvivl er det altid klogt at udskrive etikettecellerne). Programmet kan undlade at udskrive hop, der foretages indenfor eller fra opgivne etikettemærkers område.

Stopordre udskrives altid (med operationsdel) hvorefter stoppet udføres (man kan da udmærket ændre indholdet i AR før fornyet start).

Program 28 lagrer de udskrevne hop og udskriver kun hoppene 1. gang de udføres. Kataloget har dog kun plads til 29 hop, når der er fuldt bliver det slettet og programmet begynder på en frisk. Program 29 udskriver hoppene hver gang de forekommer og er derfor mere tidsrøvende.

Omlægges 56-hop omskifteren under kørslen vil programmet udskrive vr. ved det første hop det når til og derefter stoppe. Ved start forlades overvågningsprogrammet og maskinen fortsætter almindeligt i programmet, der ikke længere overvåges.

Man kan da kun komme ind i overvågningsprogrammet ved at vælge dette påny.

Man kan udmærket stoppe og benytte andre kontrolprogrammer og derefter fortsætte i overvågningen.

Plads i ferritlageret. Program 28 og 29 ligger ikke som de øvrige programmer på indlæseprogrammets plads, men kan anbringes efter ønske i ferritlageret.

Program 28 kræver 60 + 256 hac.

program 29 kræver 256 hac.

Opgives ingen speciel placering (se nedenfor) anbringes program 28 automatisk fra 1220 - 1535 og program 29 fra 1280 - 1535.

Nødvendig information:

1. information: adressen på den ordre overvågningen skal begynde på (evt. E, dersom overvågningen skal begynde fra og med den aktuelle adresse, idet kontrolprogrammet vælges).

2. information: adresse, der angiver hvor overvågningsprogrammet kan placeres i ferritlageret (max. henholdsvis 1476 og 1536), adressen skal være lige. Program 28 fylder 316 hac., program 29 256 hac.

   NB. denne information kan udelades, program 28 placeres da fra 1220 - 1535, program 29 fra 1280 - 1535.

3. information (sedecimal)
   "B" efterfulgt af et eller flere sedecimale cifre der angiver de etiketmærker indenfor hvis område der ikke ønskes hopudskrift.

   NB. denne information kan udelades, der fås da udskrift indenfor alle etiketmærker.

   Når kontrolprogrammet startes udskrives vr., hvorefter overvågningen begynder,

KAPITEL 17.

Afsluttende eksempel.

De eksempler, der er medtaget i de enkelte kapitler, vil formodentlig kunne hjælpe læseren over eventuelle vanskeligheder under udarbejdelsen af en kode, idet de forskellige problemer, der melder sig i det praktiske arbejde med nedskrivningen af koden, skulle være behandlet i et eller flere eksempler. Alligevel her vi fundet det nyttigt at medtage et eksempel på udarbejdelsen af en større kode. Herved opnår vi bl.a., at de forskellige spørgsmål behandles i sammenhæng.

Vi har valgt at behandle et konkret problem, i hvilket man skal foretage visse beregninger på givne tal. Angående den matematiske metode og de anvendte formler kan f.eks. henvises til Zurmühl: Praktische Mathematik 1953, side 274.

Numerisk harmonisk analyse.

På X-aksen (fig. 1) er stykket fra 0 til 2π delt i 24 dele, hver af længden $\frac{2\pi}{24}$. Delepunkterne betegnes $x_0, x_1, \ldots, x_{23}$. Til hvert delepunkt x svarer der en kendt funktionsværdi y. De funktionsværdier, vi her vil gå ud fra, er opstillet i tabellen.
Yderligere har vi afsat nogle af funktionsværdierne på fig. 1.

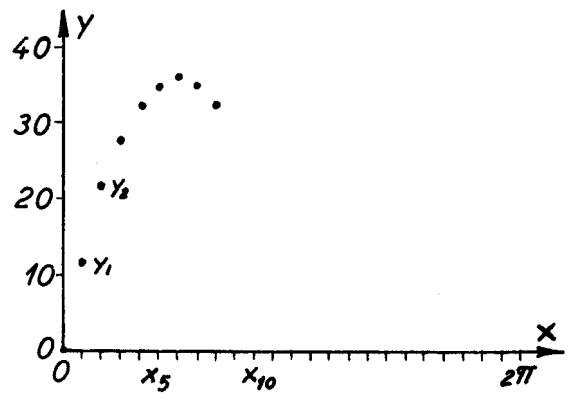


Fig. 1.

y_0	0,00	y_6	36,33	y_{12}	0,00	y_{18}	-36,33
y_1	11,53	y_7	35,08	y_{13}	-11,53	y_{19}	-35,08
y_2	20,17	y_8	32,62	y_{14}	-20,17	y_{20}	-32,62
y_3	27,94	y_9	27,94	y_{15}	-27,94	y_{21}	-27,94
y_4	32,62	y_{10}	20,17	y_{16}	-32,62	y_{22}	-20,17
y_5	35,08	y_{11}	11,53	y_{17}	-35,08	y_{23}	-11,53

Tabel 1.

Vi opstiller nu et trigonometrisk polynomium med ialt 24 led.

$$y = a_o + a_1\cos x + a_2\cos 2x + \ldots\ldots + a_{11}\cos 11x + a_{12}\cos 12x \qquad (1)$$
$$+ b_1\sin x + b_2\sin 2x + \ldots\ldots + b_{11}\sin 11x$$

I dette udtryk ønsker man at bestemme de 24 tal $a_o, a_1, \ldots a_{12}, b_1, b_2, \ldots b_{11}$, således at man, ved at erstatte x med x_o, netop får y_o; ved at erstatte x med x_1 netop får y_1 o.s.v.

Når koefficienterne er bestemt, kan (1) derefter benyttes til beregning af funktionsværdier, der svarer til vilkårlige værdier i intervallet 0 til 2π.

For koefficienterne gælder følgende almindelige formler (svarende til et vilkårligt lige antal funktionsværdier N).

$$a_o = \frac{1}{N}\sum_{j=0}^{N-1} y_j \quad ; \quad a_{\frac{N}{2}} = \frac{1}{N}\sum_{j=0}^{N-1} (-1)^j y_j$$

$$a_n = \frac{2}{N}\sum_{j=0}^{N-1} y_j \cos\left(n \cdot j \frac{2\pi}{N}\right) \quad ; \quad b_n = \frac{2}{N}\sum_{j=0}^{N-1} y_j \sin\left(n \cdot j \cdot \frac{2\pi}{N}\right) \quad ; \quad n = 1,2,\ldots, \frac{N}{2} - 1$$

Idet N = 24 får vi da følgende udtryk for de søgte koefficienter

$$(1)\quad a_o = \frac{1}{24}\sum_{j=0}^{23} y_j \qquad (2)\quad a_{12} = \frac{1}{24}\sum_{j=0}^{23} (-1)^j y_j$$

$$(3)\quad a_n = \frac{1}{12}\sum_{j=0}^{23} y_j \cos\left(n \cdot j \frac{2\pi}{24}\right) \qquad n = 1, 2, \ldots, 11. \qquad (2)$$

$$(4)\quad b_n = \frac{1}{12}\sum_{j=0}^{23} y_j \sin\left(n \cdot j \cdot \frac{2\pi}{24}\right) \qquad n = 1, 2, \ldots, 11.$$

Vi vil nu udarbejde en kode til beregning og trykning af disse koefficienter.

Indledende bemærkninger.

1) De givne talværdier (tabel 1) ligger udenfor DASK-intervallet. Ved at regne med værdierne $10^{-4} \cdot y_j$*) får man alle størrelser, herunder mellemresultater, til at ligge mellem -1 og +1. Indretter vi blot trykprogrammet til at trykke 10^4 gange de af DASK beregnede værdier, får vi netop de ønskede koefficienter.

*) der er altså anvendt skalafaktoren 10^{-4}

2) Da der foreligger en biblioteksekvens, der, for x i AR, beregner cos2πx og sin2πx, kan samtlige koefficienter beregnes uden vanskelighed. Imidlertid vil en direkte anvendelse af cos-sin sekvensen bevirke, at denne skal gennemløbes 2·11·24 gange (se formlerne (2)). Da der kun forekommer ialt 24 forskellige vinkler, hvis cos og sin yderligere gentages, kan vi nøjes med at benytte cos-sin sekvensen 7 gange, nemlig for værdierne

$0,\ 1\cdot\frac{2\pi}{24}\ ,\ 2\cdot\frac{2\pi}{24}\ \ldots\ ,\ 6\cdot\frac{2\pi}{24}$. Denne fremgangsmåde kræver til gengæld noget mere administration. Når den alligevel er foretrukket, skyldes det ønsket om at få gjort køretiden så kort som mulig, et hensyn der først får sin fulde berettigelse, når man forestiller sig, at koden skal anvendes flere gange (med nye y-værdier).

3) I betragtning af, at de fundne koefficienter normalt skal benyttes til beregning af flere funktionsværdier, foretager vi - i forbindelse med trykning - en lagring af koefficienterne.

Programmets inddeling.

Kodearbejdet bliver betydeligt nedsat gennem en opdeling af programmet. Her deler vi koden i tre dele.

HSI Beregning og lagring af de nødvendige cos-værdier.

HSII Beregning og lagring af a_0 og a_{12}. Trykningen forberedes og påbegyndes med trykning af a_0.

HSIII Beregning, lagring og trykning af $a_1, a_2, \ldots, a_{11}$ samt $b_1, b_2, \ldots, b_{11}$ Endelig trykkes a_{12}.

Vi bemærker, at trykningen i overensstemmelse med opgavens karakter indgår på naturlig måde i programmet. Der var selvfølgelig intet i vejen for at lade sekvenserne nøjes med lagringen og derefter afslutte koden med en særlig sekvens for trykning.

Etikettemærkning.

Ved nedskrivningen af koden benytter vi etiketterne på følgende måde:

	Etikettemærke	dvs. første ordre (eller tal) i:
HSI	9	0A9
beregnede cos-værdier	A	0AA
HSII	B	0AB
HSIII	C	0AC
BS sin-cos	D	0AD
Talmaterialet	E	0AE
Koefficienterne	F	0AF

Fig. 2.

De enkelte sekvenser kodes på 0A8-form.

Vi bemærker, at vi her udnytter etikettecellerne i fuld udstrækning. Da man i forvejen kender antallet af ordre i BS sin-cos og antallet af konstanter, koefficienter og cos-værdier ville besværet med at holde styr på lagringsadresserne på disse dele af koden næppe være overvældende, men ved hjælp af etikettemærkningen undgår vi helt dette problem.

Udskrift af resultaterne.

n	a	b
0	a_0	
1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
.	.	.
.	.	.
.	.	.
11	a_{11}	b_{11}
12	a_{12}	

Fig. 3.

At a_{12} først trykkes under HSIII hænger sammen med ønsket om at få trykt denne værdi til sidst, som vist på figur 3.

Koefficienterne ønskes trykt med fire decimaler.

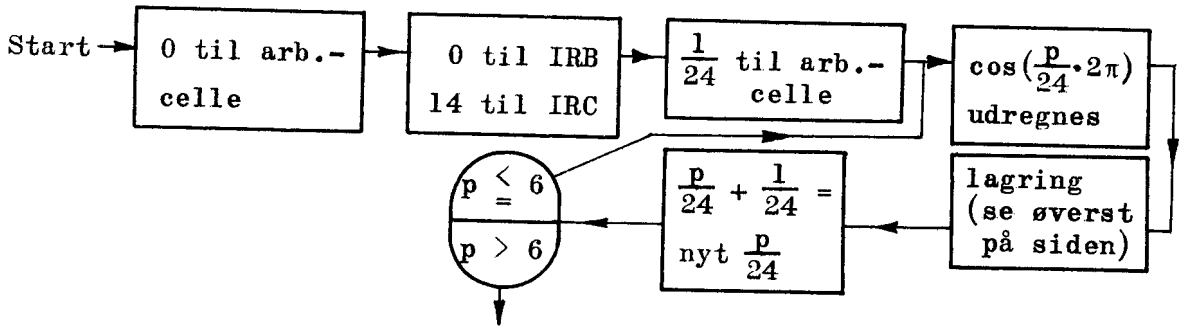
HS1.

Beregning og lagring af $\cos(0\cdot\frac{2\pi}{24}),.....,\cos(6\cdot\frac{2\pi}{24})$

$\cos(p\cdot\frac{2\pi}{24})$ p = 0,1,...,6 lagres i 2pAA og i (48 - 2p)AA

$-\cos(p\cdot\frac{2\pi}{24})$ p = 0,1,...,6 " " (24 + 2p)AA og i (24 - 2p)AA

Rutediagram.



	0(A9)	20 A8 48	(adressen 20 indføres til slut)
	1	0 A 35	
	2	14 A 55	2 gange antal omløb til IRC
	3	2039 A 60	} $\frac{1}{24}$ i MR.
	4	48 AE 2B	} $\frac{1}{24}$ i MR.
	5	0 A 07	
	6	22 A8 08	$\frac{1}{24}$ → 22 (der først indføres til slut)
18 →	7	20 A8 40	
BS ←	8	0 AD 16	
BS →	9	0 BA 08	
	10	34 CA 08	(adressen skal - første gang - være 48,
	11	0 BA 41	altså 34 da C(IRC) = 14)
	12	24 BA 08	
	13	10 CA 08	
	14	2 B 35	
	15	2046 C 55	Træk 2 fra IRC. (Ordren kunne have været
	16	22 A8 40	anbragt som ordre nr. 7).
	17	20 A8 06	
7 ←	18	7 A8 53	
HSII ←	19	0 AB 10	
	20	A	
	21	A	
	22	A	
	23	A	

HSII.

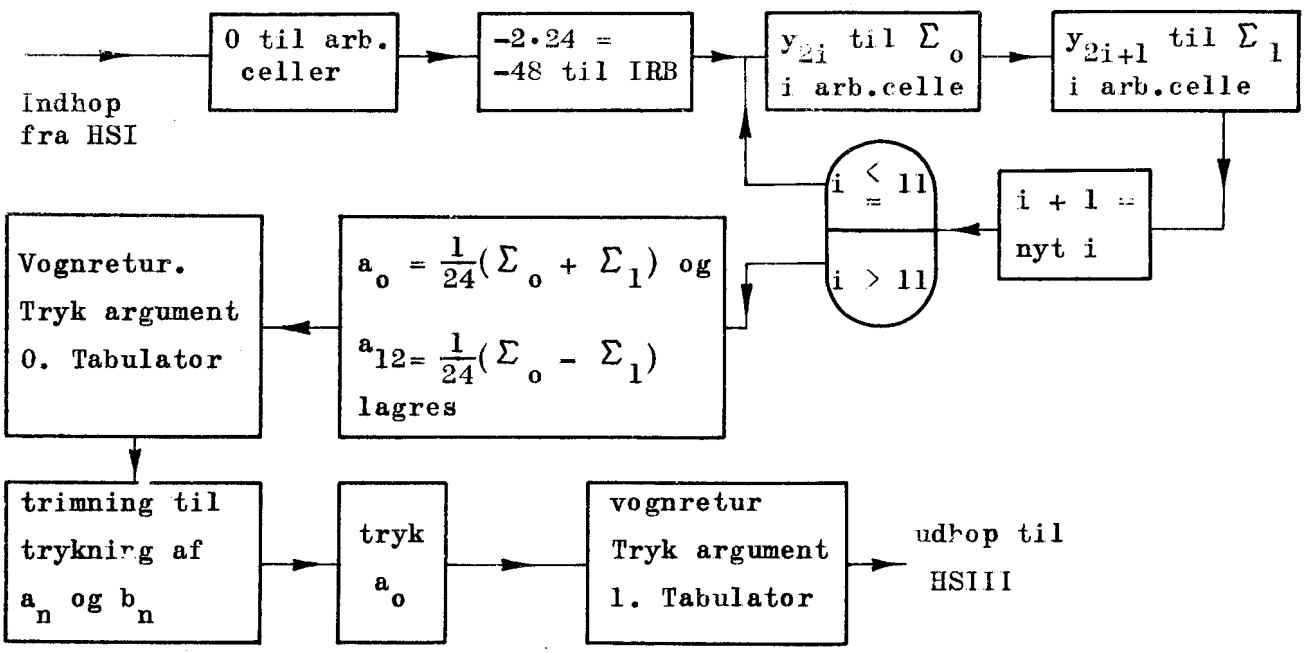
Beregning og lagring af a_o og a_{12}. Forberedelse af trykning og trykning af 1. linie (se fig. 3) samt argument 1 (i 2. linie)

Da $a_o = \frac{1}{24}\sum_{j=0}^{23} y_j$ og $a_{12} = \frac{1}{24}\sum_{j=0}^{23} (-1)^j y_j$ danner vi først

$$\sum\nolimits_o = \sum_{i=0}^{11} y_{2i} \qquad \text{og} \sum\nolimits_1 = \sum_{i=0}^{11} y_{2i+1} \quad , \text{ hvorefter}$$

$$a_o = \frac{1}{24}\left(\sum\nolimits_o + \sum\nolimits_1\right) \text{ og} \quad a_{12} = \frac{1}{24}\left(\sum\nolimits_o - \sum\nolimits_1\right).$$

Rutediagram



	0(AB)	32 A8 48	(adressen 32 indføres til slut)
	1	34 A8 48	
	2	2000 A 35	
8 →	3	48 BE 40	
	4	32 A8 06	y_{2i} til Σ_o
	5	50 3E 40	
	6	34 A8 06	y_{2i+1} til Σ_1
	7	4 B 35	
3 ←	8	3 A8 33	
	9	32 A8 04	$\Sigma_o + \Sigma_1$ → MR
	10	22 A9 0A	a_o → AR

	11	0 AF 08	a_o lagres
	12	34 A8 41	$-\Sigma_1 \rightarrow$ AR
	13	32 A8 04	$\Sigma_o - \Sigma_1 \rightarrow$ MR
	14	22 A9 0A	$a_{12} \rightarrow$ AR
	15	24 AF 08	a_{12} lagres
	16	1 A 3B	vognretur
	17	2042 A 60	0 → AR
	18	0 A 3A	trykt 0
	19	3 A 3B	tabulator
	20	1566 A 16	hop til trimning
	21	B 10264	} parametre
	22	54 AC 10	
	23	1028 A 75	} p.g.a. udlæsning med faktoren 10^4
	24	1607 A 74	
	25	0 AF 40	$a_o \rightarrow$ AR
	26	1550 A 16	hop til trykning
	27	1 A 3B	vognretur
	28	2040 A 40	1 til AR_{39}
	29	0 A 3A	tryk 1.
	30	3 A 3B	tabulator
HSIII ←	31	0 AC 10	hop til HSIII
	32	A	
	33	A	
	34	A	
	35	A	

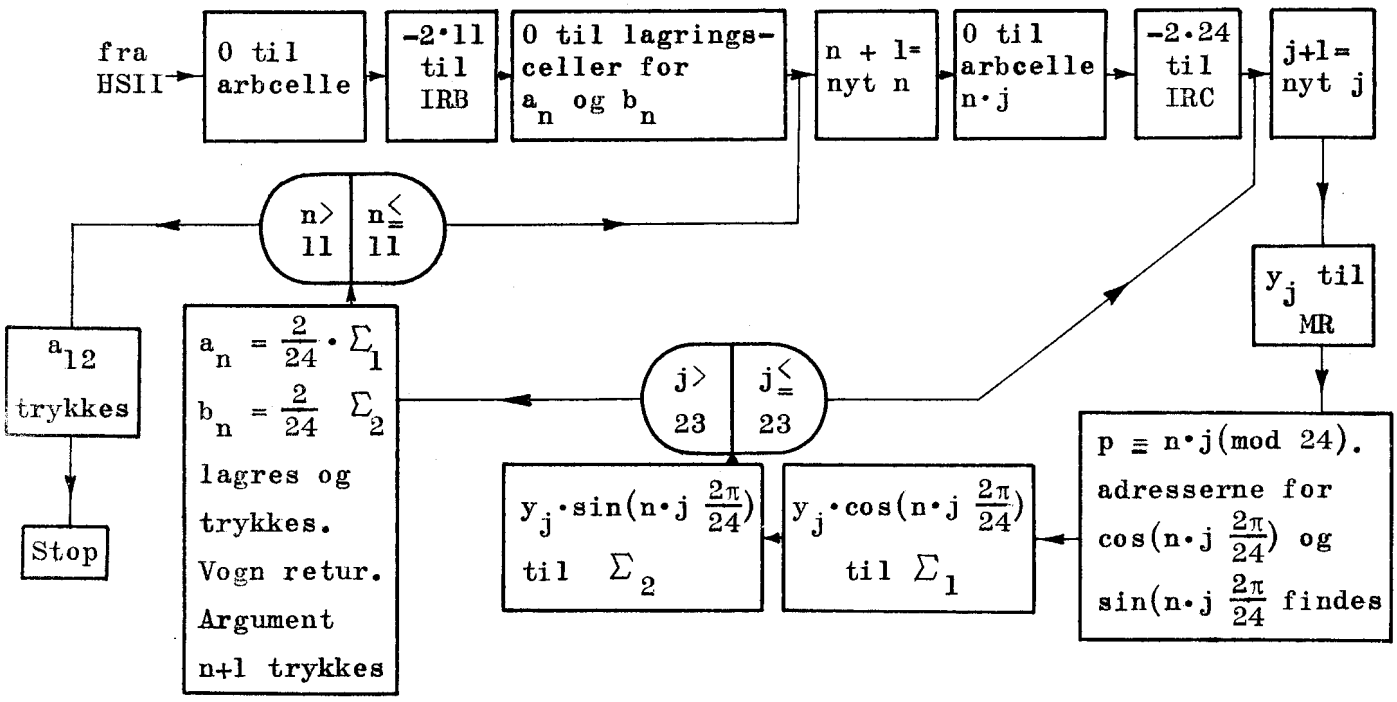
H.S. III

Beregning og lagring af $a_1 \ldots a_{11}$; $b_1 \ldots b_{11}$. Trykningen afsluttes.

$$a_n = \frac{2}{24} \sum_{j=0}^{23} y_j \cos(n \cdot j \frac{2\pi}{24}) \qquad b_n = \frac{2}{24} \sum_{j=0}^{23} y_j \sin(n \cdot j \frac{2\pi}{24})$$

Vi bestemmer p, hvor $p \equiv n \cdot j \pmod{24}$.
$\cos(nj \frac{2}{24})$ findes da i 2pAA.
$\sin(nj \frac{2}{24})$ findes i (2p - 12)AA hvis $2p - 12 \geq 0$;
ellers i (2p+36)AA.

Rutediagram



	0	58	A8	48	0 til arb.celler.
	1	2026	A	35	-22 til IRB
53 →	2	24	BF	48	0 til lagrings-
	3	48	BF	48	celler
	4	2039	A	60	2^{-11} → AR
	5	58	A8	26	2^{-11} → arb.celle
	6	59	A8	68	0 → arb.celle
	7	2000	A	55	-48 → IRC

33 →	8	48	CE	44	y_j → MR
	9	59	A8	60	den
	10	48	AE	21	rigtige
13 ←	11	13	A8	51	adresse
	12	59	A8	28	på
11 →	13	59	A8	60	$\cos(n \cdot j \cdot \frac{2\pi}{12})$
	14	1	A	0C	findes.
	15	57	A8	28	
	16	2010	A	20	
	17	18	A8	29	adressen på $\cos(n \cdot j \frac{2\pi}{24})$ i 18
	18	0	A	0A	$y_j \cos(n \cdot j \frac{2\pi}{24})$ i AR
	19	24	BF	06	lagring
	20	48	AE	61	den
	21	1	A	0D	rigtige
	22	57	A8	20	adresse
26 ←	23	26	A8	11	på $\sin(n \cdot j \cdot \frac{2\pi}{12})$
	24	48	AE	20	findes.
	25	48	AE	20	
23 →	26	2010	A	20	
	27	28	A8	29	adressen på $\sin(n \cdot j \cdot \frac{2\pi}{24})$ i 28
	28	0	A	0A	$y_j \cdot \sin(n \cdot j \cdot \frac{2\pi}{24})$ i AR
	29	48	BF	06	lagring
	30	58	A8	60	forberedelse
	31	59	A8	26	af"næste
	32	2	C	55	omgang"
8 ←	33	8	A8	53	hop
	34	22	A9	40	$\frac{1}{24}$ → AR
	35	1	A	0C	$\frac{1}{12}$ i AR
	36	2042	A	24	$\frac{1}{12}$ → MR
	37	48	BF	0A	b_n i AR
	38	48	BF	08	b_n lagres
	39	24	BF	0A	a_n i AR
	40	24	BF	08	a_n lagres
	41	1550	A	16	a_n trykkes
	42	48	BF	40	b_n i AR
	43	1550	A	16	b_n trykkes
	44	1	A	3B	vognretur

```
45     58    A8    60  ⎫  n + 1 =
46   2039    A     20  ⎬  nyt n
47     28    A     0D  ⎭  med enhed i AR39
48   2042    A     20     blind
49   1572    A     16  ⎫
50      B  90220       ⎬  tryk n
51     60    A8    30  ⎭
52      2    B     35     IRB øges med 2
53      2    A8    33     tilbagehop
54     24    AF    40     a12 til AR
55   1550    A     16     tryk a12
56     60    A8    30     stop
57      A
58      A
59      A
60   1987    A     30     stop ved sprængning af trimning.
```

Placering i lageret

Da de enkelte sekvenser fungerer uafhængigt af, hvor de er placeret i lageret, skal vi blot sørge for, at sekvenserne ikke "overdækker" hinanden. Dette krav er opfyldt ved følgende placering:

HSI	fra hac 0
HSII	fra hac 50
HSIII	fra hac 100
BS sin-cos	fra hac 170
Beregnede cos værdier	fra hac 240
Konstanterne y_0, y_1,...y_{23} samt tallet 24 (m. enhed i pos. 11)	fra hac 300
Koefficienterne a_0... a_{12}, b_1 ... b_{11}	fra hac 350

Vi bemærker, at den såkaldte "tætte" pakning af lageret ikke er tilstræbt.

Den ydre etikettering.

```
  0  E  69        0 → 2009
 50  E  6B       50 → 2011
100  E  6C      100 → 2012
170  E  6D      170 → 2013
240  E  6A      240 → 2010
300  E  6E      300 → 2014
350  E  6F      350 → 2015

  0  E  3       sæt lb.adr. = 0
  0  E  0         0 → 2008
    HSI
 50  E  3       sæt lb.adr. = 50
  0  E  0        50 → 2008
    HSII
100  E  3       Sæt lb.adr. = 100
  0  E  0       100 → 2008
    HSIII
170  E  3       Sæt lb.adr. = 170
  0  E  0       170 → 2008
    BS sin-cos
300  E  3       sæt lb.adr. = 300
```

konstanterne

```
  0  E  51
348  A  00
```

Talkoden

AA
CC001153A
CC002017A
CC002794A
CC003262A
CC003508A
CC003633A
CC003508A
CC003262A
CC002794A
CC002017A
CC001153A
AA
DC001153A
DC002017A
DC002794A
DC003262A
DC003508A
DC003633A
DC003508A
DC003262A
DC002794A
DC002017A
DC001153A
B01800

OPERATIONSLISTE FOR DASK

GRUNDFORMER

Operation	Navn	Virkning	Var. type
00	Adder	C (m) + C (AR) → AR	1
01	Subtraher	− C (m) + C (AR) → AR	1
02	Adder numerisk	| C (m) | + C (AR) → AR	1
03	Subtraher numerisk	− | C (m) | + C (AR) → AR	1
04	Adder og læs til MR	C (m) + C (AR) → AR & MR	1
05	Subtraher og læs til MR	− C (m) + C (AR) → AR & MR	1
06	Adder og læs til celle	C (m) + C (AR) → AR & m	2
07	Læs fra MR til AR	C (MR) → AR	3
08	Læs fra AR til celle	C (AR) → m	1
09	Læs fra ARadr til celle	C (ARadr) → m adr	1
0A	Multiplicer	C (m) · C (MR) → AR	4
0B	Divider	C (AR) : C (m) → MR. Divisionsrest → AR	4
0C	Skift til venstre	C (AR) skiftes m pos tv.	5
0D	Skift til højre m. tegn	C (AR) skiftes m pos th. C (AR_0) uforandret	5
0E	Normaliser	C (AR) skiftes tv., indtil C (AR_0) ≠ C (AR_1). Skiftantal → m adr	5
0F	Skift til højre u. tegn	C (AR) skiftes m pos th. 0 → AR_0	5
10	Hop	Næste ordre fås i m	6
11	Hop på fortegn	Næste ordre fås evt. i m, bet. af fortegnet af C (AR)	7
12	Hop på spild eller ej	Næste ordre fås evt. i m, bet. af spildsituation	8
13	Hop på indeks	Næste ordre fås evt. i m, bet. af C (IR)	9
14	Læs fra IR til celle	C (IR) → m adrpos	9
15	Læs adresse til IR	m → IR	9
16	Indekshop	Næste ordre fås i m. C (KR) → IRD	10
17			
18			
19	Læs fra strimmel	Sedecimale cifre læses fra str. til AR og evt. m	11
1A	Tryk ciffer	C (AR_{36-39}) trykkes eller stanses	12
1B	Tryk tegn	Tegn (afh. af m*)) trykkes eller stanses	12
1C	Vælg ydre enhed	Ydre enhed nr. m kobles til AE	13
1D	Læs fra ydre enhed	En gruppe ord læses fra ydre enhed til m, m+2 . . .	14
1E	Søg blok på bånd		
1F	Læs til ydre enhed	En gruppe ord læses fra m, m+2 . . . til ydre enhed	14

*) Operation 1B

m	Virkning
0	Blank strimmel
1	Vognretur
2	Mellemslag
3	Tabulering
4	− (minus)
5	+ (plus)
6	— (understregning)
7	. (punktum)
8	* (stjerne)
15	Stop

VARIATIONSTYPER

TYPE 1

Grf.	Ikke 0→AR	Helord
+20	før op.	Halvord
+40	0→AR	Helord
+60	før op.	Halvord

TYPE 2

06	Adder og	Helord
26	læs til celle	Halvord
46	Øg adresse	Helord
66	med 2	Halvord

TYPE 3

07 (27)	C (MR) → AR
47 (67)	C(MR) ∩ C(AR) → AR

TYPE 4

Grf.	Kort	Helord
+20	akkumulator	Halvord
+40	Lang	Helord
+60	akkumulator	Halvord

TYPE 5

Grf. (+20)	Kort akkumulator
+40 (+60)	Lang akkumulator

TYPE 6

10	Ikke 0→AR	Ikke stop før hop
30	før op.	Stop før hop
50	0→AR	Ikke stop før hop
70	før op.	Stop før hop

TYPE 7

11	Hop på +	Ikke stop før hop
31		Stop før hop
51	Hop på −	Ikke stop før hop
71		Stop før hop

TYPE 8

12	Hop på spild	Ikke stop før hop
32		Stop før hop
52	Hop på ikke-spild (Skift, hvis spild)	Ikke stop før hop
72		Stop før hop

TYPE 9

Grf.	(Ingen virkning)
+20	Vedr. IRB
+40	Vedr. IRC
+60	Vedr. IRD

TYPE 10

16	Hop, ubetinget	Ikke stop før hop
36		Stop før hop
56	Hop evt., bet.af afbr.på KB	Ikke stop før hop
76		Stop før hop

TYPE 11

19	10 sed. cifre → AR & m	Helord
39		Halvord
59 (79)	0 → AR. 1 sed. ciffer → AR_{36-39}	

TYPE 12

Grf.	Ikke 0→AR	Tryk
+20	før op.	Stans
+40	0→AR	Tryk
+60	før op.	Stans

TYPE 13

1C
(3C)
(5C)
(7C)

TYPE 14

Grf.	Helord
+20	Halvord
(+40)	(Analog grf.)
(+60)	(Analog +20)

881131