8.69

KODNING

FÖR

BESK

MATEMATIKMASKINNÄMNDENS
ARBETSGRUPP

Andra uppl. maj 1958

FÖRORD.

Denna handbok i kodning för Besk är en utvidgning av de stencilerade kompendier, som tidigare utarbetats vid Matematikmaskinnämndens Arbetsgrupp under medverkan av S. Comét, G. Kjellberg, O. Karlqvist samt några av dem, som bidragit med kapitel även denna gång. I denna upplaga har även synpunkter från utomstående, i synnerhet från programmerare och ingenjörer vid AB Åtvidabergs Industrier och SAAB, fritt utnyttjats.

Författarna vill framföra sitt tack till byråchefen G. Hävermark, Matematikmaskinnämnden, och byråchefen N. Helleberg och civilingenjören R. Nilsson vid Kungl. Vattenfallsstyrelsen för deras hjälp och initiativ vid arbetet med handboken. Vi vill också tacka ingenjör S. Lindberg, som ritat figurerna.

Stockholm den 14 november 1956.

Germund Dahlquist

FÖRORD TILL ANDRA UPPLAGAN

I föreliggande andra upplaga av "KODNING FÖR BESK" har gjorts vissa omarbetningar, som blivit nödvändiga på grund av nytillskott i programbiblioteket och ändringar av maskinens tekniska data. De senare består främst i en förkortning av operationstiderna och införandet av en ny operation för inläsning av telexremsa. Fortsatta tekniska förbättringar är planerade. Det är vår förhoppning att de gjorda omarbetningarna, som skett under medverkan av Matematikmaskinnämndens personal, också bidragit att öka framställningens tydlighet.

Stockholm den 2/5 1958

Gunnar Ehrling

Sammanställning av tekniska data om BESK f.n. (2/5 1958).

Besk är en helt binär elektronisk parallellmaskin. Dess minne består av

I. Ferritminne, kapacitet: 1024 helord (om 40 binära siffror)

II. Två trummor, kapacitet: tillsammans 8192 helord lagrade i 256 kanaler med 32 helord i varje.

Minnet användes för lagring av både program och numeriska data.

Operationstid för

addition (och de flesta andra operationer):	42 μ sek
multiplikation:	350 μ sek
division (2 op.):	686 μ sek
kanal till eller från trumman:	20-40 m sek

Talrepresentationen är helt binär och ett helord representerar ett tal x i intervallet $-1 \leq x < +1$. Negativa tal representeras med komplement. Maskinen arbetar med fix binärpunkt.

Varje instruktion innehåller 20 binära siffror. Ett helord rymmer två instruktioner av enadresstyp. Beträffande orderlistan för Besk, se "Kodning för Besk" sid. 3.6-3.10.

Inläsning sker via pappersremsa, som läses av en dielektrisk remsläsare med en hastighet av 400 rader (tecken) i sekunden. Remsan innehåller 5 kanaler. En rad på remsan kan av maskinen uppfattas antingen som ett 5-siffrigt binärt tal (telextecken) eller som ett 4-siffrigt binärt tal varvid en av kanalerna användes endast för att indikera att information finns på remsan.

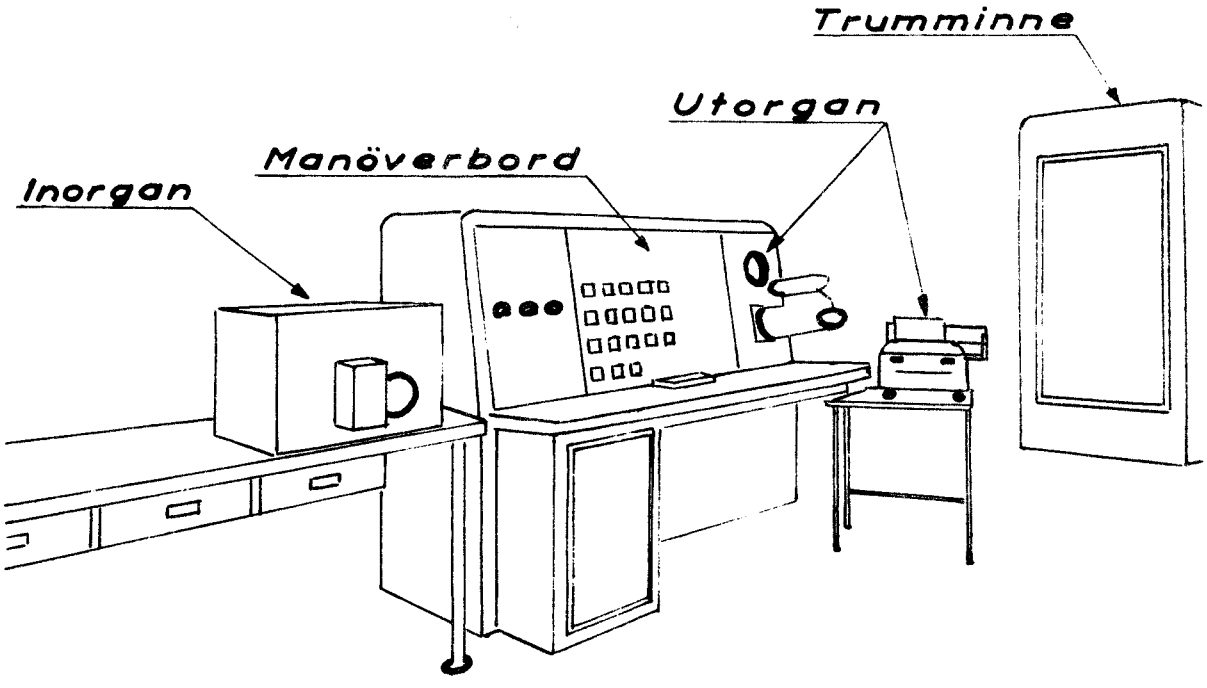
Utmatning sker dels via elektrisk skrivmaskin med c:a 15 tecken per sekund, dels via snabbstans, som f.n. stansar 145 rader per sekund på pappersremsa.

Hjälpapparatur finns för kopiering, utskrivning och stansning för hand av remsa. Vidare finns även apparatur för översättning från hålkort till remsa och vice versa samt för översättning från remsa stansad enligt telex-systemet till remsa stansad enligt det ovan nämnda speciella systemet med 4-siffriga binärtal.

INNEHÅLLSFÖRTECKNING

(Varje kapitel har sin separata paginering.)

1.	Inledning.	(Germund Dahlquist)
2.	Det binära talsystemet och representation av tal i Besk.	(Lennart Hjertman)
3.	Utförligare beskrivning av Besk och dess operationslista.	(Germund Dahlquist och Valentin Mauranen)
4.	Genomgång av operationslistan, del I.	(Germund Dahlquist)
5.	Genomgång av operationslistan, del II.	(Bengt Alenius)
6.	Genomgång av operationslistan, del III.	(Gunilla Hambraeus och Valentin Mauranen)
7.	Organisation av delprogram. Flödesscheman.	(Gunnar Ehrling)
8.	Skalfaktorer. Speciell aritmetik.	(Germund Dahlquist och Alan Edmonds)
9.	Färdigställande och inläsning av programremsor.	(Gunnar Ehrling och Ossian Holmgren)
10.	Inkörning av program.	(Lennart Hjertman och Ossian Holmgren)
11.	Biblioteket för standardkoder.	(Germund Dahlquist och Heinz-Otto Kreiss)
12.	Automatisk kodning. System FA 5.	(Gunnar Hellström och Germund Dahlquist)
13.	Utförligare teknisk beskrivning av Besk.	(Carl-Ivar Bergman, Sune Solbrand och Erik Stemme)
Appendix I.	Sakregister.	(Kerstin Arle)
Appendix II.	Tabeller.	(Kerstin Arle)


Trumminne
Utorgan
Manöverbord
Inorgan

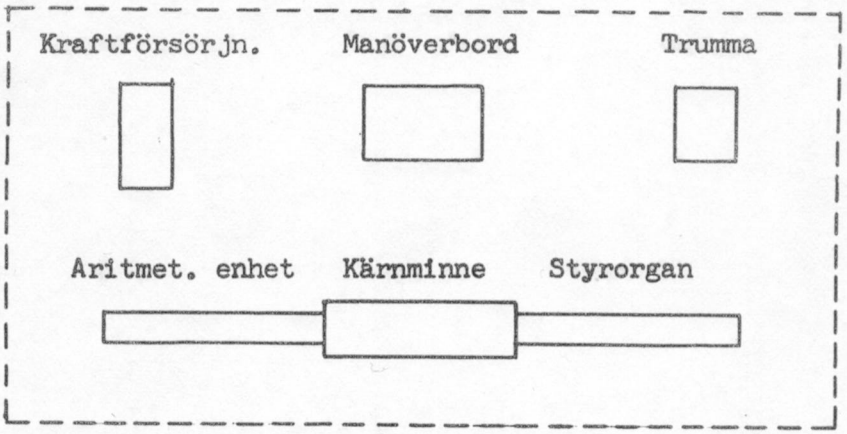


Fig. 1 Plan över Besksalen

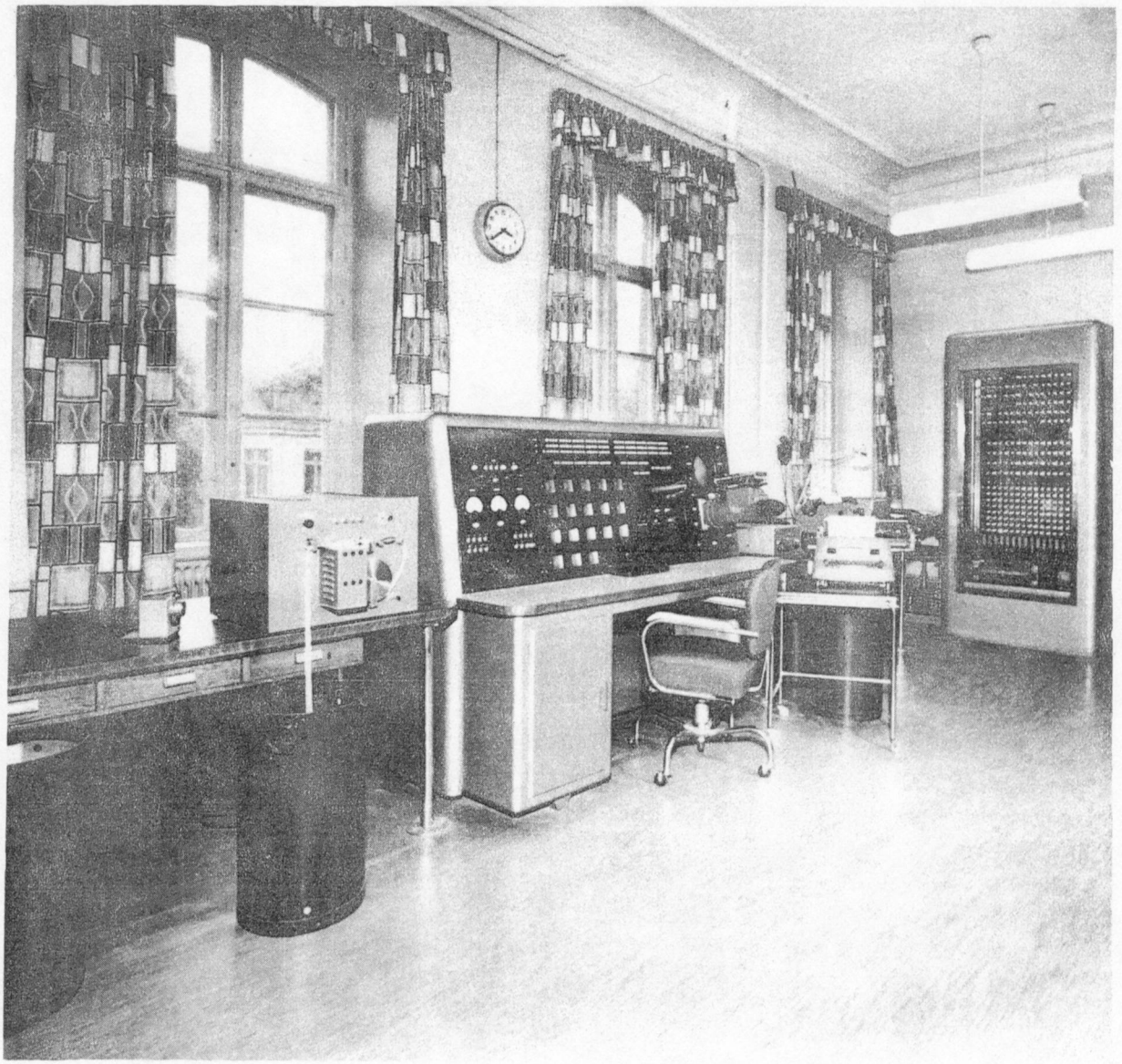


Fig. 2

1. INLEDNING

När den elektroniska siffermaskinen Besk planerades, var det många som menade, att en så snabb maskin skulle lösa alla Sveriges numeriska problem på fjorton dagar, och att den sedan endast undantagsvis skulle bli använd. Den spådomen har visat sig helt felaktig: sedan Besk togs i bruk i december 1953 har den i allt högre grad använts av industrien, försvaret och högskolorna. När detta skrivs (oktober 1956), är tiden mogen för övergång till kontinuerlig drift, och ännu är många stora projekt bara i begynnelsen. Den felaktiga spådomen berodde delvis på en underskattning av det totala antalet arbetstimmar i hela landet, som åtgår för rutinberäkningar, men framför allt förbisåg man nog, att balansen mellan experimentella och intuitiva arbetsmetoder å ena sidan och numeriska beräkningar å andra sidan skulle förändras genom att det blev möjligt att utföra omfattande beräkningar snabbare och billigare. På några områden medförde siffermaskinerna tidigt organisatoriska förändringar (t.ex. flygindustrien och meteorologien). Inom en del andra verksamhetsgrenar med omfattande beräkningsarbeten har intresset hittills varit svagare än väntat. Anledningen är nog i många fall svårigheten att se, hur den nya och främmande tekniken på lönsammaste sätt skall sättas in i ett stort problemkomplex. I sådana situationer är det dock alltid lönande att först angripa ett detaljproblem för att vinna erfarenhet och att därefter ta upp de stora frågorna.

De elektroniska siffermaskinerna är tämligen avancerade produkter av den moderna elektrotekniken, men för att självständigt använda dem och skriva egna räkneprogram behöver man inte kunna någon elektroteknik. Därför kommer den i och för sig mycket intressanta konstruktiva sidan av Besk inte att behandlas förrän i sista kapitlet. En överblick av maskinens huvuddelar och deras uppgifter är emellertid nödvändig. För att komma fram till en sådan skall vi föreställa oss ett räknebiträde R, som arbetar helt efter Besks principer. Det är då viktigt att betrakta alla arbetsmoment i räknebiträdets arbete, även t.ex. antecknandet av mellanresultat, och inte endast de egentliga aritmetiska operationerna. Det kan inte hjälpas att R och hans arbetsplats ter sig något grotesk - en organisation som är lämplig för en elektronisk maskin är inte säkert vare sig praktisk eller uthärdlig för en människa - men principiellt orimlig är inte liknelsen.

Räknebiträdet R är instängd i ett rum med en svart tavla, som är indelad i 2048 rutor, numrerade 0, 1, 2, 3, ..., numret kallas rutans adress, i varje

ruta kan han med krita skriva fem siffror. Han har också en räknesnurra för de fyra räknesätten. Han saknar förmåga till överblick och kan endast tolka mycket enkla order. Hans repertoar omfattar endast några få olika operationer, bl.a. att med räknesnurran addera det tal som står i en angiven ruta till det som förut står i räknesnurrans produktregister. En annan operation är: anteckna produktregistrets innehåll i en angiven ruta på tavlan. Operationerna på hans repertoar kan därför ges tvåsiffriga beteckningar, och därigenom kan order till R ges i en rent numerisk form. Den nämnda additionsoperationen betecknas med 10 och anteckningsoperationen betecknas med 11. Uppgiften "addera innehållet i ruta 758 till produktregistrets innehåll och anteckna resultatet i ruta 764" kan därför representeras med följande två order: 75810 76411. De tre första siffrorna i en order kallas adressdelen, de två sista kallas operationsdelen. En beräkningsgång kan beskrivas genom en längre eller kortare följd av femsiffriga order, en sådan följd kallas en kod.

En del av den svarta tavlan tas i anspråk för koden, en order i varje ruta. R har den generella anvisningen att utföra orderna i den ordning de står på tavlan, ända tills han kommer till en s.k. hopporder. Bland de operationer som R har på sin repertoar finns det nämligen också några av organisatorisk typ, bl.a. en som kallas ovillkorligt hopp: "tag nästa order från den cell, som anges i den här orderns adressdel."

R måste alltid hålla tre uppgifter i huvudet: 1. kontrollnumret d.v.s. numret på den ruta, där den aktuella ordern är skriven, 2. den aktuella orderns adressdel och 3. den aktuella orderns operationsdel. Enligt det ovan sagda ökas kontrollnumret med ett efter varje order, utom efter hopporder, där som sagt adressdelen anger nästa kontrollnummer.

R:s kontakter med omvärlden sker via två brevlådor i väggen. Den ena är för ingående material som kommer på lappar med 1 eller 10 siffror, en lapp i taget. Den andra är för utgående material, en siffra i taget. Även räkneprogrammet erhålls på detta sätt. Varje ny räkneuppgift startar med att hela tavlan suddas ren. Därefter kommer en speciell startlapp med tre siffror, om det t.ex. står 000 på lappen betyder det, både att första kontrollnumret skall vara 000, och att nästa korts innehåll (10 siffror) skall skrivas i rutorna 000 och 001, därefter utföres den order som just lästs in till 001 o.s.v., och inläsningen av koden kommer i full gång. Vi skall inte här gå in på detaljerna i inläsningen av koden, men en sak är viktig: den huvudsakliga beräkningsgången påbörjas inte förrän hela koden har skrivits upp på tavlan.

Besk är en ganska trogen analogi till R och hans utrustning, analogin definieras av följande lexikon:

R	Besk
svarta tavlan med rutor	kärnminnet (eller ferritminnet) med numrerade halvceller, vilkas innehåll kallas halvord. Minnet används både för lagring av order och tal.
räknesnurran	aritmetiska enheten med tre register för tal: ackumulator(registret) (AR), multiplikator(registret) (MR), multiplikandregistret (MD)
R:s huvud som håller reda på kontrollnummer, adressdel och operationsdel för ordern och omtolkar orderna till serier av manuella rörelser	styrorganet med kontrollregistret (KR), adressregistret (AS) och operationsregistret (OP)
lådan för ingående material	inorganet: remsläsaren som kan läsa numerisk information i form av hålkombinationer på en pappersremsa
lådan för utgående material	utorganen: a) en snabbstans varmed Besk förfärdigar pappersremsor, vars innehåll kan skrivas ut med elektriska skrivmaskiner helt utanför Besk. Remsorna kan också läsas av Besk, de utgör då ett s.k. yttre minne, b) elektrisk skrivmaskin direkt ansluten till Besk, c:a 10 gånger långsammare än snabbstansen, c) funktionsskrivare, d.ä. ett katodstrålrör där kurvor kan ritas. (Kap. 6.)

Jämför blockschemat, sid. 3.2. Analogin med R innebär också följande:

För att Besk skall lösa ett problem, måste en remsa med en kod läsas in och lagras i minnet. Inläsningen sker med hjälp av en inläsningskod, som är så konstruerad, att den först läser in sig själv och därefter läser in huvudkoden. Det finns arkivremsor för sådana inläsningskoder, som kan kopieras. Den egentliga beräkningsgången startar inte förrän hela koden blivit inläst, en halvcell för varje order. Men hur är det då möjligt att hålla maskinen sysselsatt en längre tid med ett program som ryms i det begränsade minnet ? Det beror främst på att stora beräkningsuppgifter i regel är sammansatta av ett fåtal helt olika beståndsdelar. Varje del består av ett måttligt antal operationer, och några av delarna upprepas gång på gång (ev. med mindre förändringar) med olika talmaterial.

Se t.ex. på den välkända metoden att lösa ekvationssystem av första graden med många obekanta genom att eliminera de obekanta en efter en. Detta problem är ett av de vanligaste vid Besk. Åtskilliga ekvationssystem med mer än hundra obekanta har förekommit. För att eliminera den första av de obekanta skall samma beräkningsmönster utföras för var och en av systemets koefficienter med undantag för den första ekvationens. När alla ekvationerna behandlats på det sättet, återstår ett ekvationssystem med en obekant mindre, som sedan kan behandlas på samma sätt o.s.v. och slutligen har man bara en obekant kvar. Själva aritmetiken i hela denna process kan därför framställas med en mycket kort cykel, väsentligen en multiplikation följd av en subtraktion, men man måste då lägga till några order, som ändrar adresserna i denna sekvens på det sätt, som krävs vid en regelbunden svepning av ekvationssystemets koefficientschema. Det måste alltså utföras aritmetiska operationer på adressdelarna i själva räkneprogrammet, men eftersom räkneprogrammet lagras i samma minne som talmaterialet erbjuder detta inga principiella svårigheter. Med hjälp av dylik s.k. ordermodifikation blir huvuddelen av räknearbetet utfört i en cykel med ungefär tjugo order, som genomlöps flera hundratusen gånger vid ett ekvationssystem med t.ex. hundra obekanta. Det blir i de följande kapitlen talrika konkreta exempel på detta grundbegrepp.

Ytterligare ett viktigt begrepp skall illustreras med detta exempel. Före varje ny ekvation i systemet måste några räkningar göras utöver den vanliga cykeln. Antag, att systemet har 100 obekanta, och att k koefficienter just har behandlats i en ekvation. Vad som skall ske därnäst måste därför bli olika i fallen $k \leq 100$ och $k = 100$. I förra fallet skall ett hopp göras tillbaka till cykelns början, i senare fallet skall ett hopp göras till pro-

gramdelen för förberedelse av nästa ekvation, vilken bl.a. innefattar en undersökning om det överhuvud taget finns någon "nästa ekvation" eller om man har kommit till slutet. För att möta sådana situationer har Besk på repertoaren en del operationer som kallas villkorliga hopp. En av dem, "hoppa på plus". fungerar på följande sätt:
Om ackumulatorns innehåll är negativt, ökas det aktuella kontrollnumret med ett som vanligt.
Om ackumulatorns innehåll är positivt eller noll, blir nästa kontrollnummer lika med den aktuella orderns adressdel.

Genom kombinationer av sådana villkorliga hopp kan Besk utföra komplicerade val på grundval av framräknade resultat. Begreppen ordermodifikation och villkorliga hopp är fundamentala för användningen av Besk.

Besk jämfördes ovan med det artificiella räknebiträdet R, men ingenting nämndes om tabeller,som är ett så viktigt hjälpmedel vid alla manuella beräkningar. Det är möjligt att lagra tabeller och interpolera även i Besk, så långt minnet räcker till, men eftersom minnet är begränsat och aritmetiska enheten är snabb, lönar det sig i allmänhet bättre att lagra en kod för beräkning av den önskade funktionen. Koderna för de viktigaste elementära funktionerna och en del andra ofta återkommande uppgifter har samlats i ett bibliotek för standardkoder. Kodbiblioteket är för kodaren ungefär vad tabellsamlingen är för räknebiträdet. Man kan använda en standardkod utan att känna dess konstruktion i detalj, och man behöver inte skriva av det; man ber stansbiträdet foga ihop kopior av arkivremsorna för de standardkoder man behöver tillsammans med huvudprogramremsan.

Hittills har operationshastigheterna inte angetts, hela framställningen har kretsat kring maskinens principiellt viktigare förmåga att bearbeta numerisk information helautomatiskt enligt en i maskinen lagrad kod. Men hastigheten är också viktig vid stora problem. Besk var i början av sin livstid en av de snabbaste maskinerna i världen: de flesta operationerna, bl.a. addition och subtraktion, hopporder tar ungefär 56 mikrosekunder (en mikrosekund = en miljondels sekund). Därvid är tiden att söka upp en operand i minnet och flytta den till aritmetiska enheten inberäknad. Multiplikationen är långsammare, den tar 6 ½ additionstid = 364 mikrosekunder. En division tar ungefär lika lång tid som två multiplikationer. In- och utmatning är betydligt långsammare, remsläsaren läser 400 tecken per sekund. Med tecken menas en decimal siffra eller någon av bokstäverna A, B, C, D, E, F . Snabbstansen stansar för närvarande 145 tecken per sekund, och den elektriska skrivmaskinen trycker endast 12 a 15 tecken per sekund.

Kärnminnet rymmer 2048 halvord. Dessutom har Besk ett åtta gånger så stort minne, det s.k. trumminnet, som kan betraktas som reservoar till kärnminnet. Skall information som är lagrad på trumman användas,så måste den först överföras till kärnminnet. Trumminnet är uppdelat i 256 kanaler. Varje kanal rymmer 64 halvord. Överföringen mellan trumman och kärnminnet sker en kanal i taget. Överföringen av en kanal tar mellan 20 och 40 millisekunder, alltså ungefär mellan 350 och 700 additionstider.

Efter denna översikt av Besks huvuddelar skall vi betrakta stadierna i bearbetningen av ett problem innan det kan komma till maskinen. Gränserna mellan de olika stadierna blir i praktiken oftast flytande, i synnerhet om det är samma person som utför flera av arbetsmomenten.

Först kommer analysen. Den innefattar avgränsning och matematisk formulering av problemet, val av numeriska metoder, undersökningar rörande noggrannheten o.dyl.

Sedan följer programmeringen. Då studeras beräkningarnas tidsföljd och logiska sammanhang i ord eller bild (flödesschema). Programmeraren avgör också i stora drag, hur minnet skall disponeras, hur storheterna skall representeras i maskinen - skall det t.ex. vara konstanta skalfaktorer eller flytande räkning (kap. 8). Ofta måste då mindre förändringar i utgångsformlerna göras, t.ex. ombyte av variabler. Vidare bestäms hur ingångsvärden och resultat skall ges, hur beräkningarna skall checkas, vilka standardkoder som skall användas m.m. En del av programmeringsarbetet har direkta motsvarigheter även vid planering av manuella beräkningar.

Därefter följer kodningen. Kodaren bryter ned beräkninsgången till dess elementära beståndsdelar och uttrycker den som en teckenföljd, som kan stansas på hålremsa och överföras till maskinen.

En programremsa är vanligen inte riktig första gången den sätts in i maskinen. Där kan finnas t.ex. logiska fel, matematiska fel, skrivfel, stansfel. Ett viktigt arbetsmoment är därför inkörningen. En betydelsefull del av programmerarens arbete är att planera inkörningen,så att han - utan att förbruka mycket av maskinens och sin egen tid - får upplysningar som är tillräckliga för att ringa in felen eller för att bevisa att programmet är rätt,så att reguljär körning kan påbörjas.

Stansningen av programremsan hör i regel inte till de mest betungande momenten i förberedelsearbetet till ett problem - den görs egentligen bara en gång för varje problem. Däremot är stansningen av numeriska data, som varierar från tillfälle till tillfälle när ett program används, ofta tidskrävande. Vid projekt med stora datamängder bör man därför se om det är möjligt att mekanisera den detaljen, t.ex. med hjälp av en automatisk översättare från hålkort till remsa. Det finns också i marknaden översättare från analogirepresentation till remsa, som kan kopplas till mätapparatur, och som direkt ger mätresultaten som tecken på en remsa.

Intressanta perspektiv öppnas av sådan apparatur i synnerhet i förening med Telex och med sådana rems- eller hålkortstyrda verktygsmaskiner m.m., som är under utveckling. Man kan därför använda en fjärrstyrd siffermaskin som en länk i en helt eller delvis automatiserad arbetsprocess. En amerikansk rapport berättar, att man redan nu har gjort det vid vindtunnelexperiment. Den tillsatsutrustning, som fordras för sådana användningar av Besk, hör väl delvis framtiden till, men det är nog idé att redan nu tänka på sådana möjligheter jämsides med att maskinen används för överskådligare detaljproblem.

2. DET BINÄRA TALSYSTEMET OCH REPRESENTATION AV TAL I BESK.

2.1. Binär och sedecimal representation. Besk, liksom många andra elektroniska siffermaskiner, arbetar enligt det binära talsystemet; Besk är en binär siffermaskin. Orsaken till detta är att maskinens logiska och tekniska uppbyggnad blir enklare i detta system, och räkneoperationerna kunna utföras avsevärt snabbare än om maskinen arbetar decimalt. Visserligen måste decimala data översättas (konverteras) i maskinen vid inmatning och utmatning, men räknetiden för konverteringen blir obetydlig i jämförelse med totala inläsnings- eller utmatningstiden.

I det decimala talsystemet är basen 10, och varje tal representeras med de arabiska siffrorna 0, 1, 2, ... 9, vilket betyder att t.ex.

$$1956 = 1\cdot10^3 + 9\cdot10^2 + 5\cdot10^1 + 6\cdot10^0$$

$$e = 2{,}718\ldots = 2\cdot10^0 + 7\cdot10^{-1} + 1\cdot10^{-2} + 8\cdot10^{-3} + \ldots$$

Ett brutet tal består av heltalsdel och bråkdel. Före siffrorna i bråkdelen (decimalerna) utsättes decimalkomma.

Valet av basen 10 är antagligen motiverat av antalet fingrar hos människan. Varje heltal $B > 1$ kan användas som bas i ett talsystem, och sifferrepresentationen för ett positivt tal i det "B-ala" systemet bestämmes genom en utveckling:

$$x = a_r B^r + a_{r-1} B^{r-1} + \ldots + a_1 B^1 + a_0 B^0 + a_{-1} B^{-1} + \ldots,$$

där a_r, a_{r-1}, ... a_1, a_0, a_{-1}, ... är heltal, $0 \leq a_i < B$, vilka vart och ett skall representeras med ett siffertecken. System med B = 60, 20, 12 etc. har använts.

I det binära talsystemet är B = 2 och endast siffrorna 0 och 1 används. Exempel: Talet 47 (decimal representation markeras hädanefter genom understrykning, när det är risk för missförstånd) skrives:

$$\underline{47} = 1\cdot2^5 + 0\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 101111$$

$$\underline{0{,}1} = 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + \ldots =$$

$= 0{,}0001100110011\ldots$ (periodiskt bråk).

En binär siffra kallas även bit. Före bråkdelen av ett tal utsättes binärkomma. För representation av stora tal erfordras ungefär $^2\log 10 =$ $= 3{,}32$ gånger så många bitar som decimala siffror. 10 bitar motsvarar ungefär 3 decimaler. Då det är besvärligt att räkna med binära tal på grund av det stora antalet siffror, använder man system där grupper av bitar representeras med ett siffertecken. I Besks in- och utmatningssystem representeras grupper om fyra bitar

med ett tecken, nämligen en av siffrorna 0 - 9 eller en bokstav A - F. Detta föranleder användning av det sedecimala systemet, vars bas är $2^4 = 16$.

De första heltalen i ovannämnda olika system betecknas

decimalt	binärt	sedecimalt
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
17	10001	11
18	10010	12
.	.	.
.	.	.
.	.	.

Exempel: 1956 = $7 \cdot 16^2 + 10 \cdot 16^1 + 4 \cdot 16^0$

sedecimalt:	7	A	4
binärt:	111	1010	0100

2.2. Konversion från decimalt till binärt system och vice versa. Konversion av data utföres i allmänhet medelst standardkoder i Besk, men kodaren kan ha nytta av följande regler för konversion vid handräkning eller med bordsräknemaskin.

Konversion av decimalt heltal till binärt heltal utföres på följande sätt:
Dividera talet med 16. Resten är sista sedecimala siffran
" kvoten " " " " näst sista " "
o.s.v., tills kvoten blir 0. Konvertera resultatet till binär form.

Exempel: Konvertera 1956 till binär form:

1956 : 16 rest: 4
kvot: 122 : 16 rest: 10
" 7 : 16 rest: 7

∴ 1956 = 7A4 = 0111 1010 0100

Konversion av decimalbråk till binärbråk utföres på följande sätt. Konvertera först eventuell heltalsdel för sig enligt föregående avsnitt. Multiplicera bråkdelen med 16. Heltalsdelen av produkten är den första sedecimala siffran. Multiplicera produktens bråkdel med 16. Den nya heltalsdelen är den andra sedecimala siffran. O.s.v. tills bråkdelen blir 0 eller tills tillräckligt antal siffror erhållits. Konvertera det erhållna sedecimalbråket till binär form. (Obs. Sedecimalbråket motsvarar ej den sedecimala beskrepresentationen för decimalbråket, jfr sid.2.9)

Exempel: Konvertera 0,1 till binär form.

16·0,1 = 1,6
16·0,6 = 9,6
16·0,6 = 9,6 o.s.v. (periodiskt bråk)
∴ 0,1 = 0,199A = 0,0001 1001 1001 101 avrundat (avrundning sker genom höjning av den sista siffran, om den första strukna siffran är ≥ 8).

Konversion av binärt heltal till decimalt heltal utföres på följande sätt: Uppdela talet i grupper om fyra bitar utgående från sista biten och konvertera till sedecimal form.
Multiplicera den första sedecimala siffran med 16 och addera den andra siffran. Multiplicera resultatet med 16 och addera den tredje siffran o.s.v., tills den sista siffran adderats.

Exempel: Konvertera 11 1110 1000 = 3E8 sedecimalt till decimal form.

3·16 + 14 = 62
62·16 + 8 = 1000
∴ 1111101000 = 3E8 = 1000

Konversion av binärbråk till decimalbråk kan utföras på följande sätt. Uppdela binärtalet i grupper om fyra bitar utgående från binärkommat och konvertera till sedecimal form. Konvertera sedecimaltalet som ett heltal enligt föregående avsnitt, och dividera resultatet med 2^n, där n är antalet "binaler" (bitar efter binärkommat). (Jfr även sid. 2.12)

Exempel: Konvertera 0,1101 0001 1010 = 0,D1A till decimalbråk.

$13 \cdot 16 + 1 = 209$

$209 \cdot 16 + 10 = 3354$; $2^{12} = 4096$

$3354 : 4096 = 1677 : 2048 = 0,8189$

Obs! Avrundning. 12 bitar motsvarar 4 decimaler.

$\therefore$ 0,1101 0001 1010 = 0,8189

Konversion av heltal utföres lätt med hjälp av C.E. Fröberg: "Hexadecimal Conversion Tables", Gleerup, Lund 1957; tabell I, sid. 5-9.
En kortare tabell finnes i Appendix II till detta kompendium.
Övningsuppgifter: Konvertera enligt regeln på sid. 2.2 följande decimala heltal till sedecimal form och jämför med Fröbergs tabell: 300; 1023; 1234. Konvertera enligt regeln på sid. 2.3 följande sedecimala heltal till decimal form och jämför med Fröbergs tabell: 300; 7FF; 1267.

2.3. Binär och sedecimal aritmetik. Räkningarna utföres i binära och sedecimala systemen efter samma schema som i decimala systemet.

Binär addition utföres enligt följande additionstabell: 0 + 0 = 0; 0 + 1 = 1 + 0 = 1; 1 + 1 = 10.

Exempel:

decimalt	binärt	sedecimalt
1	1 1 11 11	11
214	1101 0110	D6
+ 90	+ 101 1010	+ 5A
304	1 0011 0000	130

Översta sifferraden är minnessiffror (carry). Vid binär addition erhålles carry, när två ettor adderas, vid sedecimal addition erhålles carry, när summan av två siffror är ≥ 16.
Övningsuppgifter (sedecimal addition): 2C8 + 7F = ? 7A4 + 4CE = ?
Halvcellerna i minnet numreras sedecimalt från 000, 001 till 7FF. En matris av ordningen n antages uppfylla $2n^2$ st. halvceller fr.o.m. halvcell nr 1B0. Vilket är numret för den sista halvcellen i matrisen om n = 12? (Svar 2CF).

Subtraktion kan utföras antingen som subtraktion av absolutbelopp eller som addition av komplement. I decimala systemet är komplementet med avseende på 10^n (10^n-komplementet) till ett positivt tal $x < 10^n$ lika med

10^n - x och erhålles siffermässigt på följande sätt: Talet utfylles med nollor i början så att n heltalssiffror erhålles. Därefter subtraheras varje siffra från 9 (9-komplementet), och en etta adderas i sista positionen. Om man låter de negativa talen representeras av komplementet till motsvarande positiva tal, så ersättes operationen subtraktion med komplementbildning och addition, varvid eventuell carry från första positionen försummas. I binära systemet är på s.s. 2^n-komplementet till ett pos. tal $x < 2^n$ lika med 2^n - x och erhålles siffermässigt genom att talet utfylles med nollor i början till n heltalsbitar, varefter varje siffra subtraheras från 1 (1-komplementet), och en etta adderas i sista positionen. Samma resultat erhålles genom att taga 1-komplementet av varje siffra fram till men exklusive den sista ettan. I sedecimala systemet erhålles på motsvarande sätt 16-komplement.

Exempel:

	decimalt	binärt	sedecimalt
Tal	090	0101 1010	5A
Kompl.	910 (10^3-kompl.)	1010 0110 (2^8-kompl.)	A6 (16^2-kompl.)

Subtraktion med komplement:

decimalt		binärt	sedecimalt
utan kompl.	med kompl.	11	
214	214	1101 0110	D6
- 90	+ 910	+ 1010 0110	+ A6
124	124	0111 1100	7C

Binär multiplikation utföres enligt följande multiplikationstabell:
$0 \cdot 0 = 0$; $0 \cdot 1 = 1 \cdot 0 = 0$; $1 \cdot 1 = 1$

Exempel:

decimalt	binärt
6	110
x 5	x 101
30	110
	000
	110
	11110

2.4. Representation av tal i Besk. Innehållet i en halvcell i Besks minne är ett halvord (hao), vilket består av 20 st. binära siffror. Innehållet i ett register i aritmetiska enheten är ett helord (heo) om 40 bitar. Heo och hao kunna överföras mellan minnet och aritmetiska enheten. Bitarna i ett heo eller hao kunna representera Beskorder eller numeriska talvärden eller annan information enligt olika konventioner vid kodningen. Rent formellt är ett heo ett 40-siffrigt binärt tal med ungefär samma informationsvärde som ett 12-siffrigt decimalt tal. De aritmetiska operationerna i Besk utföras på och ger till resultat heo om 40 bitar, d.v.s. maskinens sifferkapacitet är 40 binära positioner.

De 40 binära positionerna i ett beskord numreras från vänster till höger (decimalt) från 0 till 39. Utanför maskinen representeras ofta ett binärt beskord av ett sedecimalt beskord, varvid de binära positionerna 0-3 motsvaras av en sedecimal siffra, positionerna 4-7 av nästa sedecimala siffra o.s.v., så att beskordet representeras av tio sedecimala siffror. Vi får alltså följande schema för ett godtyckligt valt beskord.

vänsterhalvord

position nr	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
binärt beskord	0	1	1	0	1	0	1	0	0	0	1	1	1	1	1	0	1	1	1	1
sedecimalt beskord	6				A				3				E				F			

högerhalvord

position nr	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
binärt beskord	1	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	1	1	0	1
sedecimalt beskord	8				4				B				0				D			

Detta skrives som sedecimalt beskord 6A3EF 84B0D.
Ett beskord kan tolkas som numeriskt talvärde på olika sätt.

2.4.1. Beskordet kan tolkas som positivt heltal med enhetssiffran i pos 39. Om b_i är siffervärdet (0 eller 1) av den binära siffran i position i, så är då beskordets talvärde

$$n = b_0 \cdot 2^{39} + b_1 \cdot 2^{38} + b_2 \cdot 2^{37} + \ldots + b_{38} \cdot 2 + b_{39} = \sum_{i=0}^{39} b_i \cdot 2^{39-i}$$

Det minsta talet är vid denna tolkning talet noll, som representeras av ett beskord, som består av 40 nollor. Det största talet representeras av beskordet 1111 1111 ... 1111 (40 binära ettor) = FFFFF FFFFF (sedecimalt) = $2^{39} + 2^{38} + \ldots + 2 + 1 = 2^{40} - 1$ = 1 099 511 627 775 (decimalt) $\approx 10^{12}$.
Alltså gäller $0 \leq n \leq 2^{40} - 1$.

Exempel.

	vänsterhalvord					högerhalvord				
pos.nr.	0	3 4	7	11	15	19	23	27	31	35	39
beskord	0000	0000	0000	0000	0000		0000	0000	0011	1110	1000

= 00000 003E8 (sedecimalt) representerar heltalet 1000 (decimalt).

2.4.2. Beskordet kan tolkas som positivt heltal med enhetssiffran i position k ($0 \leq k \leq 39$). Positionerna till höger om position k lämnas utan avseende eller antages vara noll.
Beskordets talvärde blir då

$$n_k = \sum_{i=0}^{k} b_i \cdot 2^{k-i}; \quad 0 \leq n_k \leq 2^{k+1} - 1.$$

Exempel 1. k = 11 (tolkning av adressdel i vänsterhalvord, jfr sid. 3.3):

pos.nr.	0	3	7	11	15	19	23	27	31	35	39
beskord	0011	1110	1000		0000	0000	0000	0000	0000	0000	0000

= 3E800 00000 (sedecimalt) representerar talet 1000.

I detta fall är $0 \leq n_{11} \leq 2^{12} - 1 = 4095$.

Exempel 2. k = 31 (tolkning av adressdel i högerhalvord).

pos.nr.	0	3	7	11	15	19	23	27	31	35	39
beskord	0000	0000	0000	0000	0000	0000	0011	1110	1000	0000	0000

= 00000 3E800 (sedecimalt) representerar talet 1000.

2.4.3. Beskordet kan tolkas som ett positivt binärbråk med binärkommat efter pos. 0. Den k-te positionen får då positionsvärdet 2^{-k} och beskordets talvärde blir

$$y = \sum_{i=0}^{39} b_i \cdot 2^{-i} = n \cdot 2^{-39}; \quad 0 \leq y \leq 2 - 2^{-39}.$$

Exempel.

pos.nr	0	3	7	...	39
beskord	1,	101	0000	...	0000

= D0000 00000 (sedecimalt) = $1 + \frac{1}{2} + \frac{1}{8} = 1\frac{5}{8} = 1{,}625$

Märk, att, på grund av binärkommats placering, det sedecimala beskordet motsvarar sedecimalbråket (med 10 sedecimaler) för halva det tal, som beskordet representerar:

0,D0000 00000 (sedecimalbråk) = $\frac{13}{16} = \frac{1}{2} \cdot 1\frac{5}{8}$.

2.4.4. Beskordet kan tolkas som ett positivt eller negativt binärbråk med absolutbeloppet ≤ 1. Det är denna tolkning av beskordet som ett s.k. besktal, som ligger till grund för de i Besk inbyggda operationerna för de fyra enkla räknesätten. Vid denna tolkning av beskordet som ett besktal tänker man sig binärpunkten placerad mellan pos. 0 och pos. 1. Om pos. 0 innehåller en nolla, tolkas ordet som ett positivt tal; om pos. 0 innehåller en etta tolkas ordet som 2-komplementet till ett negativt tal (se sid. 2.5). Pos. 0 är alltså teckenposition.
Den k-te positionen får då positionsvärdet 2^{-k} utom pos. 0, som får positionsvärdet -1.
Beskordets talvärde blir

$$x = -b_0 + \sum_{i=1}^{39} b_i \cdot 2^{-i}.$$

Man kan även skriva

$$x = -2b_0 + \sum_{i=0}^{39} b_i \cdot 2^{-i} = -2b_0 + y \text{ (jfr punkt 2.4.3.)}.$$

Om $b_0 = 0$, så är $x = y$; $0 \leq x \leq 1-2^{-39}$

Om $b_0 = 1$, så är $x = y-2$; $-1 \leq x \leq -2^{-39}$

Allmänt gäller för ett besktal $-1 \leq x \leq 1-2^{-39}$

Märk, att -1 är ett besktal, men ej +1.

Exempel 1.

pos.nr.	0 3 7 ... 39
beskord	0,110 0000 ... 0000

= 60000 00000 (sedecimalt) = $\frac{1}{2} + \frac{1}{4} = \frac{3}{4}$.

Exempel 2.

pos.nr.	0 3 7 ... 39
beskord	1,010 0000 ... 0000

= A0000 00000 (sedecimalt) = $-1 + 0 + \frac{1}{4} = -\frac{3}{4}$ (komplementet till föregående beskord).

Komplementet bildas genom att man inverterar alla bitar, d.v.s. byter nollor mot ettor och ettor mot nollor, varefter en etta adderas i pos. 39. Samma resultat erhålles, om alla bitar inverteras från och med pos. 0 fram till men exklusive den sista ettan. Komplementet till ett sedecimalt beskord bildas genom att man bildar F-komplementet till varje siffra fram till men exklusive den sista siffra, som är skild från noll. Till denna bildas komplementet med avs. på 10 (sedecimalt) = 16. I ett sedecimalt beskord, som representerar ett positivt besktal, är den första siffran < 8; i ett sedecimalt beskord, som representerar ett negativt besktal är den första siffran ≥ 8. Binärkommat inbakas i den första sedecimala siffran. Märk, att, på grund av binärkommats placering, det sedecimala beskordet för ett positivt tal motsvarar sedecimalbråket för halva det tal, som beskordet representerar.

Exempel 3.

	decimalt	binärt beskord	sedecimalt beskord
tal	$\frac{5}{16} =$	0,010 1000 0000 ... 0000	28000 00000
kompl.	$-\frac{5}{16} =$	1,101 1000 0000 ... 0000	D8000 00000
		40 bitar	

Sedecimalbråket 0,28000 00000 = $\frac{2}{16} + \frac{8}{256} = \frac{5}{32} = \frac{1}{2} \cdot \frac{5}{16}$.

Exempel 4.

	decimalt	binärt beskord	sedecimalt beskord
tal	2^{-39} =	0,000 0000 ... 0001	00000 00001
kompl.	-2^{-39} =	1,111 1111 ... 1111	FFFFF FFFFF

2^{-39} = 0,000 000 000 001 818 989 40 (avrundat till 20 decimaler).

Exempel 5.

	decimalt	binärt beskord	sedecimalt beskord
tal	$1-2^{-39}$ =	0,111 1111 ... 1111	7FFFF FFFFF
kompl.	$-1+2^{-39}$ =	1,000 0000 ... 0001	80000 00001

$1-2^{-39}$ = 0,999 999 999 998 181 010 60 (avrundat till 20 decimaler) ≈ 1.

Reglerna för konversion av besktal behandlas i avsnitt 2.5.

2.4.5. Beskordet kan tolkas som ett positivt eller negativt heltal med enhetssiffran i pos. 39 och pos. 0 som teckenposition. Om pos. 0 innehåller en nolla tolkas beskordet som ett positivt heltal; om pos. 0 innehåller en etta tolkas beskordet som 2^{40}-komplementet till ett negativt heltal. Beskordets talvärde blir då

$$m = -b_0 \cdot 2^{39} + \sum_{i=1}^{39} b_i \cdot 2^{39-i} = x \cdot 2^{39}$$ (jfr punkt 2.4.4).

$$-2^{39} \leq m \leq 2^{39} - 1.$$

Exempel 1.

	decimalt	binärt beskord	sedecimalt beskord
tal	10 =	0000 0000 ... 0000 1010 =	00000 0000A
kompl.	-10 =	1111 1111 ... 1111 0110 =	FFFFF FFFF6

Exempel 2.

Beskordet 1000 0000 ... 0000 (40 bitar) = 80000 00000 representerar talet -2^{39} = -549 755 813 888 (decimalt) $\approx -5 \cdot 10^{11}$ (det minsta negativa talet).

Exempel 3.

Beskordet 0111 1111 ... 1111 (40 bitar) = 7FFFF FFFFF (sedecimalt) representerar talet $2^{39} - 1$ = 549 755 813 887 (det största positiva talet).

2.4.6. Beskordet kan tolkas som ett positivt eller negativt binärbråk med enhetssiffran i pos. k och pos. 0 som teckenposition. Denna tolkning kallas besktal med binär skalfaktor 2^k (se även kap. 8.1). Beskordets talvärde blir då

$$z = -b_0 \cdot 2^k + \sum_{i=1}^{39} b_i \cdot 2^{k-i} = x \cdot 2^k \text{ (jfr punkt 2.4.4).}$$

$$-2^k \leq z_k \leq 2^k - 2^{k-39}.$$

Exempel. k = 11.
Beskordet 0000 0011 1101, 0101 0000 ... 0000 (40 bitar) = 03D,50 00000 (sedecimalt) representerar talet 61,3125. I detta fall gäller $-2048 \leq z_{11} \leq 2048 - 2^{-28}$.

2.4.7.
Beträffande beskordens tolkning vid flytande räkning och vid räkning med dubbel precision hänvisas till kap. 8.

2.5. Decimal-sedecimal konversion av besktal.

Konversion av ett besktal givet i decimal form till sedecimalt beskord kan lätt utföras med hjälp av Fröbergs tabeller (jfr sid. 2.4), tabell II, sid. 10-17, enligt anvisning på sid. 3. I tabell IV återfinnes dessutom beskorden för vissa allmänna bråk.

Konversion av ett sedecimalt beskord till decimalbråk kan utföras med tabell V, sid. 24-26. Vid användning av tabell II för konversion av decimalbråk till beskord kan man kontrollera resultatet genom att konvertera tillbaka med tabell V.

Regeln för konversion av besktal från decimal form till sedecimalt beskord vid handräkning är följande (jfr regeln för konversion till sedecimalbråk på sid. 2.3!).
Multiplicera talets absolutvärde med 8. Heltalsdelen av produkten är den första sedecimala siffran i beskordet. Multiplicera produktens bråkdel med 16. Den nya heltalsdelen är den andra sedecimala siffran. Multiplicera den nya bråkdelen med 16, o.s.v. tills 10 sedecimala siffror erhållits.
Bilda 16-komplementet, om talet är negativt.

Exempel 1. Konvertera 0,1 till sedecimalt beskord.

0,1 · 8 = 0,8
0,8 · 16 = 12,8
0,8 · 16 = 12,8 o.s.v. (periodiskt bråk)
∴ 0,1 = beskord 0CCCC CCCCD.

Exempel 2. Vilket är beskordet för $-\frac{1}{7}$?

$\frac{1}{7}\cdot 8 = 1\frac{1}{7}$; $\frac{1}{7}\cdot 16 = 2\frac{2}{7}$; $\frac{2}{7}\cdot 16 = 4\frac{4}{7}$; $\frac{4}{7}\cdot 16 = 9\frac{1}{7}$; o.s.v.

$\therefore \frac{1}{7}$ = beskord 12492 49249.

$-\frac{1}{7}$ = beskord EDB6D B6DB7.

Konversion från sedecimalt beskord till decimalbråk kan utföras för hand på följande sätt. Om d decimaler önskas, avrunda beskordet till d sedecimaler (ev. d-1 sedecimaler, se anm.). Bilda komplementet, om beskordet är negativt (d.v.s. första siffran ≥ 8). Dividera den sista sedecimalen med 16, avrunda kvoten till en decimal. Sätt näst sista sedecimalen som heltalsdel till kvoten, dividera med 16, avrunda till två decimaler. Sätt den tredje sedecimalen som heltalsdel, dividera med 16, avrunda till tre decimaler. O.s.v. tills den första sedecimalen medtagits, varvid man sista gången dividerar med 8 och avrundar resultatet till d decimaler.

Anm. Om $d > 5$, d.v.s. fler än fem decimaler önskas, kan man avrunda beskordet till (d-1) sedecimaler och avrunda kvoten vid den första divisionen till två decimaler, vid den andra divisionen till tre decimaler o.s.v.

Exempel. Konvertera beskordet 023BE 8D44A ($= 180/\pi$) till decimalbråk med fem decimaler.
Beskordet avrundas till fem sedecimaler: 023BF.

F	15:16 = .9
B	11.9:16 = .74
3	3.74:16 = .234
2	2.234:16 = .1396
0	0.1396:8 = .01745

Svar: 023BE 8D44A $\approx$ 0,01745.

Övningsuppgifter. Konvertera med hjälp av Fröbergs tabell eller medelst handräkning följande decimala tal till sedecimalt beskord: $\frac{9}{16}$; -0,625; 0,8; 0,16667 (medtag 5 signifikanta sedecimaler); $\pi/4$ = 0,785 398 (medtag 5 sedecimaler).

Konvertera med Fröbergs tabell eller genom handräkning följande sedecimala beskord till decimalbråk:
D8000 00000; 145F3 10000 (= $\frac{1}{2\pi}$, avrunda till 7 decimaler).
2AAAA AAAAB = x (periodiskt bråk). Anvisning: $\frac{1}{2}$x = 0,2AAA..(sedecimalbråk); 8x = 2,AAAA...; 8x - $\frac{1}{2}$x = 2,8000... = $2\frac{1}{2}$; x = $2\frac{1}{2}:7\frac{1}{2} = \frac{1}{3}$.

Standardkoderna för konversion i Besk från decimal till binär form och vice versa (vid in- och utmatning av decimala data) arbetar enligt liknande regler, men räkningarna utföras där i det binära systemet i stället för i det decimala.

2.6. Aritmetiska operationer i Besk.

En aritmetisk operation i Besk ger riktigt resultat, under förutsättning att operanderna och resultatet tolkas som besktal och att det riktiga resultatet av operationen är ett besktal. Vid addition och subtraktion bortfaller eventuell carry från pos. 0. Subtraktion utföres som addition av komplementtal. I vissa operationer ingår absoluta beloppet av en operand, d.v.s. komplementet tages av operanden, om den är negativ. Vid multiplikation med avrundning och vid division avrundas resultatet automatiskt till 40 bitar (se sid. 4.2 resp. 4.5). Vid division $\frac{a}{b}$ måste |a| vara $\leq$ |b|, för att riktigt resultat skall erhållas.

Exempel 1. Addition.

	beskord
carry	(1)1111
Tal	14C15 00000
+ kompl.	+ EB3EB 00000
= 0	00000 00000

Exempel 2. Subtraktion. 0,1 - 0,8 = -0,7.

0,1	= 0CCCC CCCCD	
-0,8	= +99999 9999A	kompl. av 66666 66666
-0,7	= A6666 66667	" " 59999 9999A

Exempel 3. Bildning av $|a - b|$ där

a	= 14A31 00000		14A31 00000
$-b$	= -14C15 00000	kompl.	+ EB3EB 00000
$a - b$	= FFE1C 00000		FFE1C 00000
$\|a - b\|$	= 001E4 00000		

Exempel 4. Multiplikation (se vidare sid. 4.2).

	binärt	sedecimalt beskord
$\frac{1}{2} \times \frac{3}{4} = \frac{3}{8}$;	0,1000...	40000 00000
	x 0,1100...	x 60000 00000
Resultat i Besk: $\frac{3}{8}$ =	0,0110...	30000 00000

Exempel 5. Division (se vidare sid. 4.5).

$-\frac{1}{2} : \frac{3}{4} = -\frac{2}{3}$;	1,100...	C0000 00000
	: 0,110...	: 60000 00000
Resultat i Besk: $-\frac{2}{3}$ =	1,010101...	AAAAA AAAAB

En addition eller subtraktion blir riktigt utförd i Besk, endast om det riktiga resultatet av operationen är ett besktal. Om det riktiga resultatet är ≥ 1 eller < -1, erhålles i Besk som resultat ett besktal x, som skiljer sig från det riktiga resultatet y med en term, som är en heltalsmultipel av 2, d.v.s. x är kongruent med y modulo 2. I detta fall har Besks sifferkapacitet överskridits, man har fått spill, och spillindikation erhålles (se sid. 3.1 och 4.1). Om resultatet av flera additioner eller subtraktioner är ett besktal, erhålles detta riktigt, även om spill inträffat under mellanstegen. Spill kan även erhållas vid vissa andra operationer i Besk.

Exempel 1.

$-\frac{1}{2} - \frac{3}{4} = -1\frac{1}{4}$; binärt:

```
   (1)
    1,10
+   1,01
   0,11  = 3/4 = - 1 1/4 + 2 (spill)
```

Exempel 2.

$\frac{5}{8} + \frac{3}{8} = +1$;

```
    0,101
+   0,011
   1,000 = -1 = +1 - 2 (spill)
```

Eftersom +1 ej är besktal, erhålles spill i det sista exemplet. Om det riktiga resultatet av en operation är +1, erhålles i allmänhet som resultat besktalet -1 med spill. Exempel: komplementet av -1 blir i Besk -1 med spill, absolutbeloppet av -1 blir -1 med spill, oavrundad multiplikation (-1)(-1) ger -1 med spill. Om $|a| < 1$ erhålles däremot riktigt resultat av följande operationer: $-1 \cdot a = \frac{a}{-1} = -a$. Operationen $\frac{-1}{a}$ ger givetvis fel resultat, $\frac{a}{a}$ blir $= 1-2^{-39}$. Orden 0 (idel nollor) och -1 (en etta i pos. 0, följd av idel nollor) är = sina egna komplement. Man kan undersöka om ett ord är 0 eller om ett ord är -1 på följande sätt. Ordet 0 är det enda, vars absolutvärde med ombytt tecken har positiv teckensiffra i Besk; ordet -1 är det enda, vars absolutvärde har negativ teckensiffra i Besk. Operationerna i Besks aritmetiska enhet ger direkt riktiga resultat, om helorden tolkas som besktal enligt ovan. Vissa operationer kan även förklaras som om Besk arbetade med heltal i pos. 39, d.v.s. besktalet $x = n \cdot 2^{-39}$ representerar då heltalet n. Vid addition och subtraktion spelar det ingen roll, var man tänker sig binärpunkten placerad, bara den har samma position i alla termerna. Om helorden tänkes representera tal med binärpunkten placerad efter någon annan position än pos. 0, måste varje multiplikation och division kombineras med skiftorder.

Detta kallas räkning med binära skalfaktorer, se sid. 8.1. Räkning med flytande binärpunkt och räkning med dubbel precision behandlas i kap 8.

3. UTFÖRLIGARE BESKRIVNING AV BESK OCH DESS OPERATIONSLISTA.

En del av den grundläggande terminologin infördes i kapitlen 1 och 2. Här skall några kompletteringar ges.

3.1. Kärnminnet består av 2048 halvceller, som är numrerade sedecimalt från 000, 001, ... till 7FE, 7FF. (Observera att 7FF = 2047). Dessa nummer kallas adresser. En cells adress finns inte lagrad i cellen utan är endast definierad genom logiska arrangemang i de informationsvägar, som leder till och från cellen. En vänsterhalvcell (vhac) har en jämn adress, och en högerhalvcell (hhac) har en udda adress. En helcell består av en vhac tillsammans med den hhac, som har närmast högre adress. En hec har samma adress som sin vhac.
Exempel: hec20A består av vhac20A och hhac20B.

Innehållet i en vänsterhalvcell kallas vänsterhalvord (vhao). På liknande sätt definieras högerhalvord (hhao), helord (heo) och halvord (hao). Innehållet i t.ex. hhac20B betecknas hhao20B. Tidigare användes även beteckningen C(20B), som är en förkortning av det engelska uttrycket "the contents of 20B". Adressen till en cell där storheten (ordet) x lagras betecknas W(x). Vid resonemang av generell natur används bokstaven W för att beteckna adressdelen i den betraktade ordern, och i stället för heoW eller haoW skrivs kortare w.

3.2. Aritmetiska enheten innehåller tre register, som vardera rymmer ett helord: multiplikandregistret (MD), ackumulatorregistret (AR) och multiplikatorregistret (MR). De helord som lagras i dessa register betecknas med md, ar resp. mr. Positionerna i MD betecknas MD0, MD1, ..., MD39. Vänsterhalvan och högerhalvan av MD betecknas MDV resp. MDH. Vänsterhalvordet och högerhalvordet i MD betecknas mdv resp. mdh. Motsvarande gäller för AR och MR.

AR har två extra positioner AR00 och AR40, vilkas uppgift i de flesta fall är av komplicerad teknisk natur. Deras viktigaste användningar skall förklaras nedan i samband med de olika operationerna. Här skall endast nämnas att om siffrorna i AR00 och AR0 är olika, säges maskinen indicera spill. Spillindikationen (si) betyder i samband med vissa operationer (bl.a. vid addition och subtraktion men icke vid exempelvis division), att det matematiskt riktiga resultatet av en räkning ligger utanför Besks sifferkapacitet. Bland Besks operationer finns ett villkorligt hopp som styrs av si. Tidigare si ligger kvar under sådana operationer, som aldrig kan förändra ar.

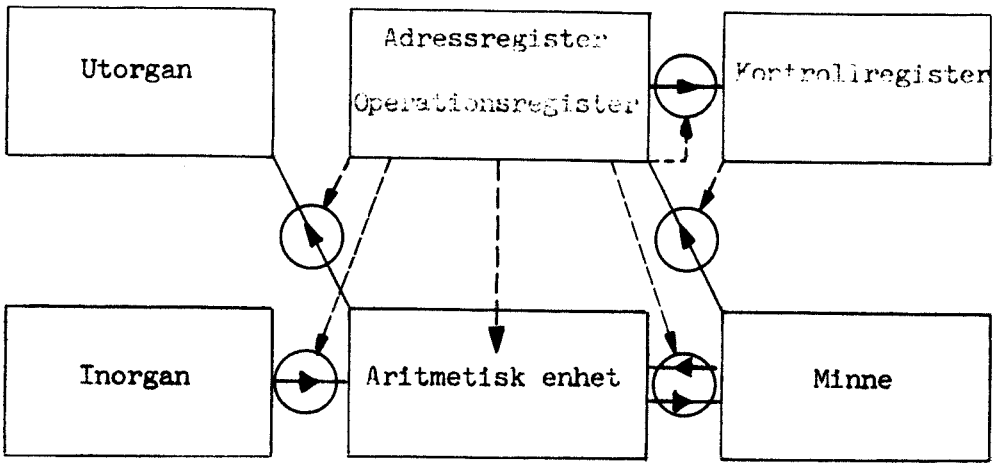


Fig.3 Förenklat blockschema för Besk
En cirkel betyder att överföringen styrs av det register som den streckade pilen utgår från.

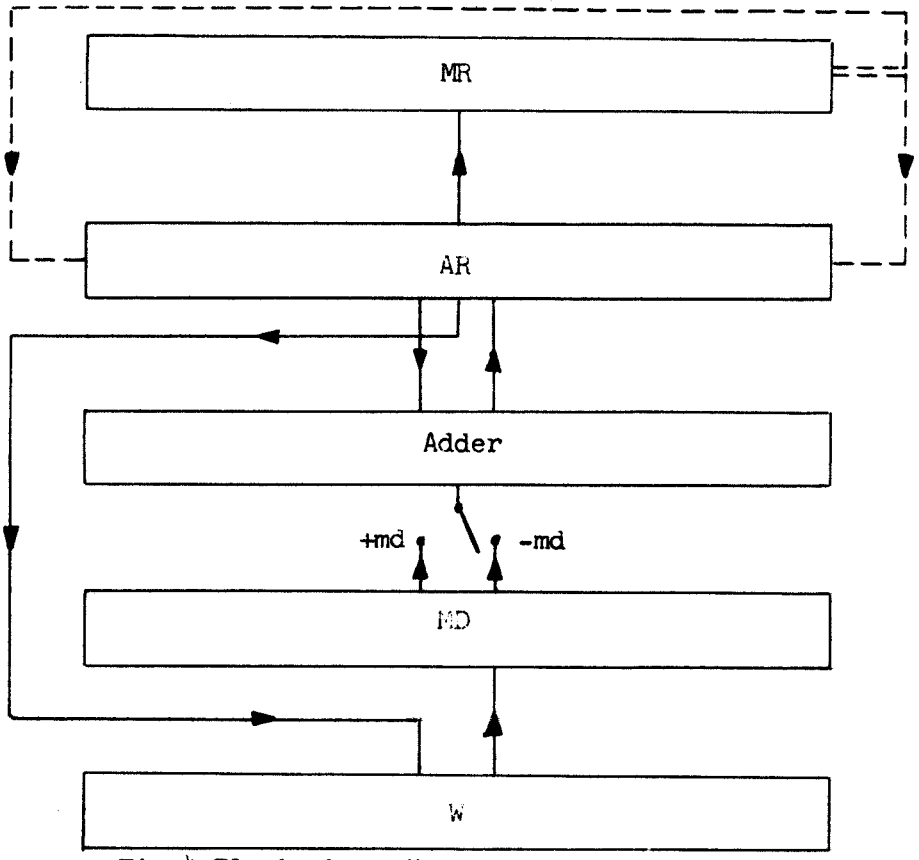


Fig.4 Blockschema över aritmetiska enheten.
I den undre bilden betyder en heldragen linje parallellöverföring, en streckad serieöverföring.

Ett helord eller ett halvord, som hämtas från en cell till aritmetiska enheten, kommer först i oförändrat skick till MD. Om det är ett vhao som hämtas, så blir högerhalvan av MD fylld med nollor. Om det är ett hhao som hämtas, så blir vänsterhalvan av MD fylld med nollor. Inuti aritmetiska enheten behandlas sedan ett ord alltid som ett helord. Därför syftar beteckningarna md, ar och mr alltid på helord. Detta är viktigt att minnas vid operationer med halvord, ty det innebär, att ett hhao som hämtats från W blir behandlat som ett positivt tal. Aritmetiska enheten känner nämligen ingen annan teckensiffra än helordets teckensiffra (position 0), som enligt vad som nyss sades blir nollställd, när ett hhao hämtas. Innan man blivit van kodare, bör man därför undvika att använda hhac för att lagra storheter som kan vara negativa.

Mellan MD och AR finns en adder, som kan bilda md + ar, -md +ar, |md| + ar eller -|md| + ar beroende på vad som påkallas av ordern. Resultatet av operationen kommer till AR. Ett ord som skall till eller från MR måste alltid passera AR (se blockschema). Ett ord som skall flyttas från aritmetiska enheten till W går direkt från AR. Därvid kan helord eller halvord överföras. Ett vhao kan endast sättas i en vhae, ett hhao endast i en hhac.

Man skiljer mellan parallellöverföring och serieöverföring mellan celler (register) och register. En parallellöverföring är snabb, alla siffror flyttas på samma gång. Vidare är den sändande cellen (registret) ostörd, d.v.s. det överförda ordet finns fortfarande kvar där. En serieöverföring är långsam (7,7 additionstider), siffrorna flyttas en efter en, och det avsändande registret töms på sitt innehåll. Endast överföringen från MR till AR[+)] är av serietyp, alla andra överföringar (även den från AR till MR) är parallella. Se fig. 4.

3.3. Styrorganet innehåller bl.a. fyra register, nämligen kontrollregistret (KR), adressregistret (AS), operationsregistret (OP) och en binärräknare (BR). Innehållen i KR, AS och OP betecknas kr, as resp. op. Innehållet i KR kallas kontrollnumret.

När en kod lagras i minnet, tar varje order en halvcell i anspråk - en vhac och en hhac är i detta sammanhang likaberättigade. En order skrivs alltså med fem sedecimala siffror. De tre första siffrorna kallas orderns adressdel, de två sista kallas dess operationsdel - precis som för "räknebiträdet R" i kapitel 1.

[+)] och överföringarna mellan trumminnet och AR

När en order skall utföras, anger kr den adress från vilken den aktuella ordern skall hämtas till styrorganet. Denna orders adressdel går då till AS och operationsdelen till OP. kr ökas normalt med ett för varje order som utförs. Undantag är hoppoperationerna (OA, OC, OE och därav härledda operationsformer). Det är viktigt att hålla isär begreppen kr och as. I de flesta fall anger as från vilken adress ett tal skall hämtas till aritmetiska enheten eller den adress där resultatet skall lagras.

Exempel: Antag att kr = 348 och att vhao348 = 15810. Då går 158 till AS och 10 till OP. 10 är beteckningen för addition, så att ordern innebär att vhao158 + ar bildas och placeras i AR.

AS bryr sig inte om den första binära positionen i adressdelen. Om det står 800 i adressdelen, tolkar AS det som 000. Ordern i ovanstående exempel hade lika gärna kunnat skrivas 95810. Kärnminnet kan därför betraktas som cykliskt, vilket kan vara till nytta för kodaren ibland.

Den första binära positionen i operationsdelen har en speciell funktion. Om det står en etta där, och om en omkopplare vid manöverbordet står i ett visst läge, får man en kontrollutskrift, som närmare beskrivs i kap. 10. I övrigt utförs ordern på samma sätt, som om det hade stått en nolla. Vi skall därför vid diskussionen av operationslistan anta att första binära siffran i operationsdelen är noll, d.v.s. att operationsdelen är mindre än 80 (sedecimalt).

3.4. Allmänt om Besks operationer. Varje operation är sammansatt av mindre deloperationer, s.k. mikrooperationer. I vilken ordning dessa utförs framgår av mikrooperationsschemat, kap. 13. Här skall endast nämnas att bl.a. överföring av ordern från minnet till styrorganet och eventuell nollställning av AR äger rum före eventuellt stopp eller kontrollutskrift. Däremot sker hämtning av tal från W till MD, lagring av resultat samt ändring av kr efter eventuellt stopp eller kontrollutskrift. Ett undantag utgör operationen med beteckningen 06 och dess härledda former. När man trycker på någon av manöverbordets startknappar, utförs endast senare delen av den order som är uppsatt i styrorganet. Start- och stoppknapparnas funktion beskrivs närmare i kap. 10.

Beteckningarna i Besks operationslista är valda efter en viss regel, från vilken det emellertid finns några undantag. Man talar om grundform och härledda former för en operation. Första sedecimala siffran i grundformen är 0

eller 1. För grundformen finns ingen mnemoteknisk princip, men regeln för de härledda formerna framgår av följande schema.

Skrivsätt för op	I början av operationen blir AR	Adressdelen syftar på
Grundform	Icke nollställd	halvcell
Grundform + 20	" "	helcell
Grundform + 40	Nollställd	halvcell
Grundform + 60	"	helcell

Regeln kan också uttryckas så, att andra binära positionen i operationsdelen avgör om AR skall nollställas, medan tredje binära positionen väljer mellan hel- och halvcell.

Exempel: 10 är grundformen för additionsoperationen. 30, 50 och 70 är de härledda formerna.

Order	Verkan
15810	vhaol58 + ar till AR
15830	heol58 + ar till AR
15850	vhaol58 till AR, ty AR nollställs i början av operationen
15870	heol58 till AR

Ordern 15970 uppfattas av Besk som en motsägelse - helcell med udda adress - och maskinen stannar. En udda adressdel i förbindelse med "grundform + 20" eller "grundform + 60" i operationsdelen är alltid en stoppkombination. Om man efter stoppet trycker på startknappen utförs ordern som en halvcellsoperation d.v.s. operationsdelen minskas med 20. I det anförda exemplet utförs alltså ordern 15950.

Vissa operationer - bl.a. 01, 04, 05 - är interna operationer i aritmetiska enheten. De utförs därför på helord även i grundformen. Adressdelen är i de fallen antingen irrelevant, eller också har den en annan betydelse än normalt (jfr operationerna 04, 05). I de fallen rekommenderas grundformen, ty då riskerar man inte att oavsiktligt råka ut för stoppkombinationen, som även gäller i de fall, då orderns adressdel ej syftar på någon cell i minnet.

Undantag mot regeln för de härledda operationsformerna är följande:

Vissa operationer är utan intresse om de inleds med att AR nollställs, andra saknar intresse om AR inte nollställs. I några av dessa fall - operationerna 04, 05, 06, 0E, 15, 19 - betyder därför "grundform + 40" och "grundform + 60"

intressantare varianter. Ett särfall är operationen 74 sid. 3.8 där endast den härledda formen 74 kan användas.

I nästa avsnitt sammanställs Besks operationslista. I de tre följande kapitlen kommenteras och exemplifieras de olika operationerna.

3.5. Besks operationslista.[+)]

Teckenförklaringar:

Kolumn "as" (adressdel):
- W — adressdelen syftar på helcellen (eller halvcellen) W i kärnminnet.
- irr — adressdelen är irrelevant.
- k — adressdelen betyder antalet skiftsteg (sedecimalt)($k \geq 0$).

Kolumn "op" (operationsdel):
- ( ) — operationsformen har huvudsakligen intresse vid stoppkombination.
- / / — operationsformen har veterligen aldrig använts.

Kolumnerna "w, ar, mr":
- x — innehållen i W, AR eller MR resp. kan förändras vid operationen.
- o — " " " " " " " är noll efter operationen.

Kolumn "Verkningar":
- w, ar, mr betyder innehållet i hec (eller hac) W, ackumulatorregistret AR resp. multiplikatorregistret MR före operationen.

Kolumn "Tid":
- Additionstiden (c:a 42 mikrosekunder) är tidsenhet.

Kolumn "Anm.":
- u — innebär att operationen är ett undantag från regeln för de härledda formerna.
- r — betyder att de operationsformer som icke upptagits i listan härledas enligt regeln men saknar intresse.
- Siffran i anmärkningskolumnen anger sidnummer, där operationen kommenteras.

Operation	as	op	w	ar	mr	Verkningar	Tid	Anm.
Addera	W	10 30		x x		w + ar till AR.	1	4.1
		50 70		x x		w till AR.		
Subtrahera	W	0B 2B		x x		- w + ar till AR.	1	4.1
		4B 6B		x x		-w till AR.		
Addera absolut	W	0D 2D		x x		|w| + ar till AR.	1	5.1
		4D 6D		x x		|w| till AR.		
Subtrahera absolut	W	0F 2F		x x		-|w| + ar till AR.	1	5.1
		4F 6F		x x		-|w| till AR.		
Addera och sätt i MR	W	08 28		x x	x x	w + ar till AR och till MR. Om w = 0; ar till MR och kvar i AR.	1	4.1
		48 68		x x	x x	w till AR och till MR.		
Subtrahera och sätt i MR	W	09 29		x x	x x	-w + ar till AR och till MR. Om w = 0; ar till MR och kvar i AR.	1	4.1
		49 69		x x	x x	-w till AR och till MR.		

+) Läsaren rekommenderas att vid första genomgången endast studera huvuddragen.

Operation	as	op	w	ar	mr	Verkningar	Tid	Anm.
Lagra i cell	W	11 31	x x			ar till W och kvar i AR.	1	4.2
		51 71	o o	o o		AR och W nollställs.		
Lagra i adressdel	W	07 27	x x			I W:s adressdel (-delar) sätts innehållet i AR:s motsvarande adressdel (-delar). Övriga pos. i W ostörda.	1	5.4
		47 67	x x	o o		Hela AR nollställs, W:s adressdel (-delar) nollställs. Övriga pos. i W ostörda.		
Addera och lagra i cell	W	06 26	x x	x x		w + ar till AR och till W.	$1\frac{2}{3}$	5.5 u
Öka adressdel	W	46	x	x		w + 0020000200 till AR; w + 00200 till W.		
		66	x	x		w + 0020000200 till AR och dito till W.		
Multiplicera utan avrundning	W	02 22		x x	x x	w·mr + 2^{-39}·ar till AR jämte MR. Teckensiffran och de 39 första bitarna av resultatet kommer i AR, de 39 återstående bitarna i pos. 1-39 i MR. Pos. MR0 blir 0.	$8\frac{1}{3}$	4.2
		42 62		x x	x x	w·mr till AR jämte MR. F.ö. som vid 02 och 22.		
Multiplicera med avrundning	W	/03/ /23/		x x	x x	Se kap. 4.	$8\frac{1}{3}$	4.2
		43 63		x x	x x	Teckensiffran och de 39 första bitarna av w·mr + 2^{-40} till AR. Se kap. 4.		
Överför mr till AR	irr	01		x	o	mr skiftas in i AR. 0 till MR.	$7\frac{2}{3}$	4.2 r
Dividera	W	12		x	x	ar/w bakvänt till MR. Pos. MR0 blir 1.	$8\frac{1}{3}$	4.5 r
		32		x	x	Divisionens rest i AR. Se kap. 4.		
Överför mr till AR bakvänt	irr	13		x	o	mr skiftas baklänges in i AR	$7\frac{2}{3}$	4.5 r
Extrahera	W	00 20		x x	o o	Resultatet kommer i AR och har ettor i de positioner där både w + ar och mr hade ettor; nollor i alla andra positioner. 0 till MR.	$7\frac{2}{3}$	5.8
		40 60		x x	o o	Resultatet kommer i AR och har ettor i de positioner där både w och mr hade ettor; nollor i alla andra positioner. 0 till MR.		
Skifta höger	k	04 /24/		x x		Om k $\leq$ 03F, skiftas ar åt höger k steg. I AR0 inmatas efter varje skiftsteg en kopia av den ursprungliga teckensiffran.	$1\frac{2}{3}$- $-11\frac{2}{3}$	5.1 u
Skifta höger utan teckenreproduktion	k	44 /64/		x x		Om k $\leq$ 03F, skiftas ar åt höger k steg. AR0 deltager i skiftet, men efter varje skiftsteg inmatas en nolla i AR0.		

Operation	as	op	w	ar	mr	Verkningar	Tid	Anm.
Skifta vänster	k	05 /25/		x x		Om k $\leq$ 03F, skiftas ar åt vänster k steg. I AR39 inmatas en nolla efter varje skiftsteg.	$1\frac{2}{3}$ - $-11\frac{2}{3}$	5.1 u
Skifta vänster med AR40	k	45 /65/		x x		Om k $\leq$ 03F, skiftas ar åt vänster k steg. AR40 deltager i skiftet, men efter varje skiftsteg inmatas en nolla i AR40.		
Normalisera	irr	15 /35/		x x	x x	ar skiftas åt vänster till dess innehållen i AR0 och AR1 blir olika (om ar $\neq$ 0). MR blir först nollställt, sedan insätts där antalet utförda skiftsteg med enheten i pos. 31. Om ar = 0, görs 63 skiftsteg. I AR39 matas alltid nollor.	$1\frac{2}{3}$ - $-11\frac{2}{3}$	5.3 u
		55 /75/		x x	x x	Som ovan, men AR40 deltager i skiftet. Nollor inmatas sedan i AR40.		
Hoppa	W	0C (2C)				Nästa kr = W (Hoppa till W).	1	4.6
		4C (6C)		o o		AR nollställs och nästa kr=W.		
Hoppa vid plus eller noll	W	0E (2E)				Nästa kr = W om ar $\geq$ 0; om ar $<$ 0, ökas kr som vanligt med ett.	1	4.6 u
Hoppa vid minus	W	4E (6E)				Nästa kr = W om ar $<$ 0; om ar $\geq$ 0, ökas kr som vanligt med ett. AR nollställs icke.	1	
Hoppa vid spillindikation	W	0A				Nästa kr = W om spillindikation finns i AR; annars ökas kr som vanligt med ett.	1	4.6 r
Läs helord från remsa	W	19 39	x x	x x		Ett helord läses från pappersremsa till AR. Därifrån sändes ar som hao eller heo till W. Det inlästa heo står även kvar i AR.	c:a 600	6.1 u
Läs 4-kanalsrad från remsa	W	59 79	x x	x x		AR nollställes och en rad med kanal 5 stansad läses från pappersremsa till AR pos. 36-39. Kanal 1 till AR36, kanal 4 till AR39. Sedan sändes ar till W som hao eller heo. Den inlästa raden står även kvar i AR. Se kap. 6 sid. 6.1.	c:a 60	6.1
Läs 5-kanalsrad från remsa	W	74	x	x		AR nollställes och en rad med någon kanal stansad läses till AR pos. 35-39. Kanal 1 till AR35, kanal 5 till AR39. Sedan går ar till W som heo. Stopp om W udda.	c:a 60	6.2a u

Operation	as	op	w	ar	mr	Verkningar	Tid	Anm.
Tryck eller stansa siffra	irr	1C				Om skrivmaskinen är inkopplad, trycks ar:s fyra sista bitar som en sedecimal siffra. Om stansen är inkopplad, stansas nämnda siffra på pappersremsa. Även kanal 5 på remsan stansas.	c:a 2000 resp. 160	6.4 r
		5C		o		AR nollställs och siffra 0 trycks i skrivmaskin eller stansas på remsa. F.ö. som 1C.		
Tryck eller stansa tecken	irr	1D				Om skrivmaskinen är inkopplad, framkallar ar:s fyra sista bitar ett typografiskt tecken eller en typografisk operation. Om stansen är inkopplad, stansas de nämnda bitarna i kanal 1-4 på remsan. Remsans kanal 5 stansas ej. De sedecimala siffrorna motsvarar typografiskt: 0 mellanslag, 1 vagnretur med radframmatning, 2 punkt (.), 3 tabulator, 4 minustecken (-), 5 plustecken (+), 6 understrykning (_); sändes före den siffra som skall tryckas, 8 asterisk (*), F stoppkombination på remsan vid utskrivning (framkallar ingen operation vid utmatningsskrivmaskin).	variabel resp. 160	6.5 r
		5D		o		AR nollställs och mellanslag utföres eller stansas på remsa. F.ö. som i 1D.		
Skriv på trumma	W	1F 3F 5F 7F		o o o o		1F och 5F, se kap. 6. 7F har samma verkan som 3F: Helorden W, W + 002, W + 004, ..., W + K - 002 går via AR in på den trumkanal, vars nummer man i en tidigare order placerat i MR med enheten i pos. 31. Obs! trumkanalens nummer tages som ett jämnt tal. Värdet på K är i regel 40 (d.v.s. 64 halvord = 32 helord), helkanal. Även 3/4, 1/2 och 1/4 kanal kan man använda efter en manuell omkoppling. Efter operation är ar = 0.	480--960	6.6

Operation	as	op	w	ar	mr	Verkningar	Tid	Anm.
Läs från trumma	W	1B 3B /5B/ /7B/	x x x x	x x x x		5B resp. 7B har samma verkan som 1B resp. 3B: Helorden i den trumkanal, vars nummer man i en tidigare order placerat i MR på samma sätt som i 3F ovan, går till AR och därifrån till halv- resp. helcellerna W, W + 002, W + 004, ..., W + K - 002; betr. K se 3F. Trummans innehåll förblir ostört. Det sist överförda helordet står efteråt också kvar i AR.	480--960	6.6
Sänd till funktionsskrivare (F)	000 /002 004 006 008 /00A 00C 00E	18 18/ 18 18 18 18/ 18 18				ar till F:s x-reg. Skriv ej. -"- -"- -"- Skriv.Punkt. -"- " Cirkel. ar till F:s y-reg. Skriv ej. -"- -"- -"- Skriv.Punkt. -"- " Cirkel. ar betyder här teckensiffran tagen separat samt en följd om åtta successiva bitar. Följden kan börja i någon av pos. 1, 2, 3, 4, 5, 6, 7 i ar beroende på läget hos omkopplaren "Position X" resp. omkopplaren "Position Y" på manöverbordet.	c:a 1 2/3 -9	6.7 r

Operationerna 14, 16, 17, 1A, 1E och de därav härledda operationerna (utom 74 sid. 3.8) är inte definierade. Om maskinen beordras göra en sådan operation, kan dock innehållet i vissa register eller celler förändras eller maskinen stannar. Det är planerat att ta dessa operationsdelar i anspråk vid utbyggnader av maskinen.

Permanenta konstanter.

Det förutsättes i denna framställning, om annat inte uttryckligen sagts, att även följande s.k. permanenta konstanter har blivit inlästa på följande platser i kärnminnet:

hac	Konstant	Användning
000	00200	2 i adressposition
001	00200	" " "
002	00100	1 " "
003	00100	" " "
004	80000	vhao av "-1" } $-1 + 2^{-39}$
005	00001	1 i pos. 39 }
006	00000	vhao av 0
007	00839	primitiv inläsning.

OBS! Dessa konstanter är inte inbyggda i Besk, men de läsas in av alla MNA:s standardprogram för inläsning av koder.

4. GENOMGÅNG AV OPERATIONSLISTAN, DEL I.

Operationslistan skall nu gås igenom närmare och dess användning skall exemplifieras. Kompletterande teknisk information ges i kap. 13. I det här kapitlet skall tretton av grundformerna och deras härledda former behandlas. De kodexempel som följer kan lämpligen betraktas som delar av större koder. Order som skall utföras efter varandra skrivs under varandra. Frågeställningen är: hur skall en kod se ut i minnet för att maskinen skall göra de önskade beräkningarna? Frågan hur man får in koder och tal i minnet tas upp först i kapitel 6. De permanenta konstanterna förutsätts vara inlästa. Se listan i slutet av kapitel 3.

Operation:	Grundform:	Verkningar:
Addera	W 10	w + ar till AR
Subtrahera	W 0B	-w + ar till AR
Addera, och sätt i MR	W 08	w + ar till MR och även till AR
Subtrahera, och sätt i MR	W 09	-w + ar till MR och även till AR

De härledda formerna tolkas enligt regeln i alla fyra fallen. Om operationens matematiskt riktiga resultat r uppfyller olikheten $-1 \leqq r < 1$, så räknar Besk rätt. Ingen si (spillindikation) finns då efter operationen. Om däremot det riktiga resultatet ligger utanför Besks kapacitet, d.v.s. om $r \geqq 1$ eller om $r < -1$, så ger Besk resultatet r-2 resp. r+2 och då blir det si efter operationen. Om det är si eller ej före operationen spelar ingen roll för tillståndet efter. Tidigare si utplånas till och med vid addition av talet noll. Vid addition och subtraktion av flera tal blir resultatet alltid riktigt modulo 2, jfr kap. 2.

Exempel 4.1: Antag att AR innehåller ordet 35D02 32111. (Ordet är helt godtyckligt valt.) Vad blir ar och mr efter utförande av orderna

a) 00608 b) 00628 c) 00648 d) 00668.

Svar: Enligt listan över permanenta konstanter är heo006 = 00000 00839.

a) ar = mr = 35D02 32111
b) ar = mr = 35D02 3294A
c) ar = mr = 00000 00000
d) ar = mr = 00000 00839

Observera att i aritmetiska enheten behandlas all information som helord. (Se stycket 3.2).

Exempel 4.2: Antag att de permanenta konstanterna är inlästa. Visa att ordern 00608 då betyder att ar sätts i MR. Vad innebär 00648?

Operation:	Grundform:	Verkningar:
Lagra i cell	W 11	ar till W

De härledda formerna tolkas enligt regeln. si kan inte uppstå vid denna operation. Tidigare si kvarstår vid operationsformerna 11 och 31. Däremot försvinner eventuell tidigare si vid operationsformerna 51 och 71.

Exempel 4.3: x, y, z är lagrade i hec100, 102 resp. 104. Bilda x + y - z och lagra det i hec180.

Lösning:

	kommentar
10070	x till AR
10230	x + y till AR
1042B	x + y - z till AR
18031	x + y - z till hec180

Exempel 4.4: x står i AR. Sätt -x i AR.

Lösning: Två order fordras. Det är lämpligt att först flytta x till en cell, t.ex. hec100.

10031	x till hec100
1006B	-x till AR

(Vad hade resultatet blivit om andra ordern hade skrivits 1002B?)

Operation:	Grundform:	Verkningar:
Överför mr till AR	irr 01	mr till AR 0 till MR
Multiplicera utan avrundning	W 02	se nedan!
Multiplicera med avrundning	W 03	

De härledda formerna tolkas i de två senaste fallen enligt regeln. För operationen 01 har alla de härledda formerna samma innebörd, och dessutom är adressdelen irrelevant. Rekommendation: skriv alltid 00001.

Den matematiskt riktiga produkten av två tal, som vardera har teckensiffra och ytterligare 39 binära siffror, är ett tal med teckensiffra och ytterligare 78 binära siffror. När Besk bildar produkten mr·w med operationen 62 (eller 42), kommer främre halvan, d.v.s. teckensiffran och de 39 första siffrorna i AR, medan bakre halvan, d.v.s. de 39 följande siffrorna fyller

positionerna MR 1, MR 2, ..., MR 39, som alltså får positionsvärdena 2^{-40}, 2^{-41}, ..., 2^{-78}. I position MR 0 sätts alltid en nolla. Om man är tillfreds med att få produkten med tecken och 39 binära siffror med korrekt avrundning, så är operationen 63 (eller 43) den riktiga. Den ger nämligen i AR den främre halvan av mr·w + 2^{-40}. Observera, att de former som inleds med nollställning av AR skall användas vid båda typerna av multiplikation. Spillindikationens funktion vid multiplikation är ännu inte helt utredd, men omfattande experiment ger vid handen, att si uppstår vid operationsformerna 42, 62, 43, 63 endast när det riktiga resultatet är +1.

Multiplikation utförs i Besk enligt en binär motsvarighet till det vanliga sättet att utföra multiplikation för hand. Vid operation 02 och dess olika former överförs först w till MD. Därefter följer en procedur som upprepas 39 gånger: Om mr39 = 1 bildas md+ar och sätts i AR, förskjutet ett steg åt höger. Om mr39=0 skiftas ar ett steg åt höger. Samtidigt skiftas (i båda fallen) mr ett steg åt höger, varvid den sista siffran faller bort. ar40 kopieras i MR0. (mr och ar betyder här det aktuella innehållet i MR resp. AR, icke innehållet vid operationens början. md ändras ej under operationen). Efter 39:e varvet skiftas mr ett steg till höger, varvid 0 sätts i MR0. Om den ursprungliga multiplikatorn är negativ, subtraheras dessutom md från ar.

Operation 03 och dess olika former utförs på liknande sätt. Enda skillnaden är att 1 sätts i AR1 utan att övriga positioner störs. Denna insättning görs <u>före</u> första varvet men <u>efter</u> den nollställning av AR, som operationsformerna 43 och 63 inleds med.

Om AR inte nollställs i början, så ger operationerna 22 och 02 resultatet mr·w + ar·2^{-39}. Teckensiffran och produktens främre halva kommer till AR, den bakre halvan kommer till MR och i MR0 sätts en nolla - på samma sätt som vid operationen 62. Formerna 22 och 02 är användbara bl.a. vid heltalsräkning, inläsning av decimala tal (kap. 6) och räkning med dubbel noggrannhet (kap. 8). Formerna 23 och 03, vilkas resultat kan härledas ur beskrivningen ovan, har veterligen aldrig använts.

<u>Exempel 4.5</u>: Ordern 00442 betyder att -mr kommer till AR. Om mr = 80000 00000, så kommer 80000 00000 till AR.

Exempel 4.6: x, y är lagrade i hec100, 102. Sätt -xy i hec180 med tecken och 39 korrekt avrundade binära siffror.

Lösning:
```
10069    -x till MR
10263    -xy till AR, avrundat
18031    -xy även till hec180
```

Exempel 4.7: x och y är lagrade i hec100, 102. Sätt tecken och de 39 första siffrorna av xy i hec180, de 39 sista siffrorna i hec182 med nolla i pos. 0.

Lösning:
```
10068
10262
18031
00001
18231
```
(Studera order för order hur ar och mr varierar.)

Exempel 4.8: x, y, z är lagrade i hec100, 102, 104. Lagra xyz i hec106. Avrundade multiplikationens noggrannhet är tillräcklig.

Lösning:
```
10068
10263
00608
10463
10631
```

Exempel 4.9: x_1, x_2, x_3 är lagrade i hec100, 102, 104.
y_1, y_2, y_3 är lagrade i hec110, 112, 114.
Bilda $x_1 y_1 + x_2 y_2 + x_3 y_3$ och lagra i hec120. Koden antages börja med kr = 200.

Lösning: Vi antecknar även kr för varje order

kr	as op	
200	10068	x_1 till MR
201	11063	x_1y_1 till AR
202	12031	x_1y_1 även till hec120
203	10268	x_2 till MR
204	11263	x_2y_2 till AR
205	12030	$x_1y_1 + x_2y_2$ till AR
206	12031	dito även till hec120
207	10468	x_3 till MR
208	11463	x_3y_3 till AR
209	12030	$x_1y_1 + x_2y_2 + x_3y_3$ till AR
20A	12031	dito även till hec120

Observera att koden har en viss regelbundenhet med "perioden" 4, som skulle fortsättas om produktsumman hade flera termer.

Operation:	Grundform:	Verkningar:
Dividera	W 12	ar/w bakvänt i MR m.m.
Överför mr till AR bakvänt	irr 13	mr baklänges till AR

De härledda formerna tolkas enligt regeln. För operationen 13 har alla de härledda formerna samma innebörd, och dessutom är adressdelen irrelevant. Rekommendation: skriv alltid 00013.

Exempel 4.10: W(a) = 100, W(b) = 102. Antag att $|a| \leq |b|$. Bilda a/b i AR.

Lösning:
10070 a till AR
10232 a/b baklänges till MR. Observera helordsformen 32.
00013 a/b rättvänt i AR

Exempel 4.11: Operation 13 vänder på den binära representationen av mr, icke den sedecimala. Om mr = 30D40 07208, så blir ar = 104E0 02B0C efter ordern 00013.

Divisionsoperationen 12 och dess olika former tillgår i Besk på följande sätt. Dividenden måste i en tidigare operation ha placerats i AR. Av divisionsoperationen blir divisorn placerad i MD. Därefter inträffar följande: Om ar och md har samma tecken, skiftas ar ett steg åt vänster, varefter md subtraheras från ar, och 1 sätts i MR0. Om ar och md har olika tecken, skiftas ar ett steg åt vänster, varefter md adderas till ar, och 0 sätts i MR0. Därefter skiftas mr ett steg åt höger (i båda alternativen). Detta händelseförlopp upprepas 39 gånger. Efter sista gången sätts 1 i MR0 (för avrundning), och slutligen ersätts siffran mr 39 med sitt komplement.

Antag att a/b skall beräknas, och att $|a| \leq |b|$. För att tolka Besks division är det lämpligt att införa heltalen

$$A = a \cdot 2^{39}, \quad B = b \cdot 2^{39}$$

Låt X vara resultatet av en exakt utförd division 2^{39} A/B. Låt Q vara det närmaste udda talet till X och definiera R genom villkoret

$$2^{39}A = BQ + R \quad (|R| \leq |B|).$$

Efter divisionsoperationen (12, 32) står $r = R \cdot 2^{-39}$ i AR och $q = Q \cdot 2^{-39}$ står bakvänt i MR. q kallas beskkvoten, och r kallas beskresten. Beskresten kan vara negativ. Observera att det sagda inte ger en entydig bestämning av Q och R i det speciella fall, då den exakta divisionen $X = 2^{39}A/B$ går jämnt ut och som resultat har ett jämnt tal. Om $|A| < |B|$ visar då en närmare analys att Q = X + 1, om B>0 och Q = X - 1, om B<0. Ett viktigt specialfall är

a = 0. Beskkvoten är då alltså inte noll utan 2^{-39} eller -2^{-39} beroende på divisorns tecken. Divisionen a/a ger alltid beskkvoten 7FFFF FFFFF. Om $a \neq 0$ och $a \neq -1$, ger divisionen -a/a beskkvoten 80000 00001.

Om $|a| > |b|$ blir Besks division felaktig även modulo 2. Det resultat som erhålls kan beskrivas på följande sätt. Inför beteckningen

$$x'' = \begin{cases} 1-x & , \text{ om } x > 0 \\ -1-x & , \text{ om } x < 0 \end{cases}$$

Om $|a| > |b|$ så är $|a''| < |b''|$. Antag att Besk vid divisionen a"/b" skulle ge kvoten $\overline{q}$ och resten $\overline{r}$. Resultatet av divisionen a/b kan då skrivas

$$q = \overline{q} \quad , \quad r = (\overline{r})''$$

Spillindikationen tycks inte följa några användbara regler vid operationerna 12 och 13. Speciellt kan den inte användas för att upptäcka fallet $|a''| > |b''|$, efter det att divisionen utförts.

Operation:	Form:	Verkningar:
Hoppa	W 0C	W → kr
Hoppa på plus eller noll	W 0E	W → kr, om ar ≧ 0. Annars kr+1 → kr
Hoppa på minus	W 4E	W → kr, om ar < 0. Annars kr+1 → kr
Hoppa på spill	W 0A	W → kr, om det är spillindikation. Annars kr+1 → kr.

Operationen 0E är ett undantag från regeln, medan de härledda formerna av 0C och 0A skall tolkas enligt regeln. Operationerna 0E, 4E och 0A kallas villkorliga hopp. Spillindikation kan icke uppstå eller utplånas av de operationsformer som aldrig kan ändra ar, t.ex. 07, 27, 0C, 0E, 4E, 11, 31, 1C, 1D.+)

Exempel 4.12: a > 0, b > 0, är lagrade i helceller. Utför a/b, om b ≧ a. Stanna maskinen om b < a. Koden skall börja på order n.

Lösning:

kr	as	op	
n	W(b)	70	b till AR
n+1	W(a)	2B	b-a till AR
n+2	n+4	0E	Om b ≧ a: hoppa till n+4. Om b < a: gå vidare till n+3
n+3	001	20	stoppkombination, om b < a
n+4	W(a)	70	a till AR
n+5	W(b)	32	a/b bakvänt i MR
n+6	000	13	a/b rättvänt i AR

+) Jfr avsnitt 3.2 samt beskrivningarna av de olika operationerna.

Exempel 4.13: Beräkna $\frac{1}{2} x^9$. Hoppa därefter till 240. x är lagrad i hec100. Koden skall börja på 200. Man kan koda detta explicit med nio multiplikationer, men det är också möjligt att göra beräkningarna i en cykel med användning av rekursionsformeln

$$y_0 = \tfrac{1}{2}, \quad y_{i+1} = x y_i \quad (i = 0,1,2, \ldots, 8).$$

Då är $y_9 = \frac{1}{2} x^9$.

200	21151	Varvräkningsindex i nollställs
201	20C50	$\frac{1}{2}$ till AR
202	20E31	$\frac{1}{2} \rightarrow$ y
→203	10068	x till MR
204	20E63	xy till AR
205	20E31	xy ersätter y
206	00550	2^{-39} till AR
207	21110	i ökas med 1
208	21111	
209	20D0B	i - 9 till AR
20A	2034E	i = antalet genomlöpta varv. Om i<9, hoppa tillbaka
←20B	2400C	till 203. Om i = 9 hoppa till 240
20C	40000	Konstanten $\frac{1}{2}$
20D	00009	Konstanten $9 \cdot 2^{-39}$
20E	00000	y (ändras under räkningarna)
20F	00000	
210	00000	
211	00000	$i \cdot 2^{-39}$ (ändras under räkningarna)

I kapitel 5 skall det visas att varvräkningen kan utföras på smidigare sätt. Observera att koden med obetydliga ändringar kan användas på allmänna uttryck av formen ax^n.

Blandade exempel: Låt koderna börja vid kr = 200.

Exempel 4.14: x, y, z, u är lagrade i hec100, 102, 104, 106.

a) Bilda $(x + y)(z - u)$ i hec10E

b) Bilda $\dfrac{x}{y + z} + u$ i hec110

c) Bilda $y x^2 + z x + u$ i hec10E

Exempel 4.15: x, y är lagrade i hec100, 102. $x > 0$, $y > 0$. Placera det större av talen x och y i hec104. Behandla sedan uppgiften utan att antaga, att x och y är positiva. Obs! Det kan då inträffa att $|x - y| \geqq 1$.

Exempel 4.16: x, y är lagrade i hec100, 102. Låt x och y byta plats

a) utan några inskränkande krav

b) utan att använda några andra celler än hec100 och 102.

Exempel 4.17: Sekvensen[+)] nedan till vänster är en del av ett stort program. Den avser beräkning av x^3y, där x och y är lagrade i hec100, 102. Mellan ordern 213 och 214 är emellertid en order 00608 bortglömd. Man vill inte numrera om i hela programmet, vilket skulle bli nödvändigt, om man satte in ordern på dess rätta plats. Man kan då rätta med en parentes på det sätt som beskrivs till höger. (Rättelser av den här typen kan göras antingen i koden före stansningen eller sedan koden stansats på remsa. Jfr rättelseremsa, ex. 6.1.). Antag att cellerna från 480 och framåt är outnyttjade.

```
...
212   10268        ...
213   10063        213   4800C
214   10063        ...
215   00608        480   10063
216   10063        481   00608   Parentes
...                482   2140C
```

Exempel 4.18: x, y, z är lagrade i hec100, 102, 104. Följande sekvens, som avser beräkning av xz/y och lagring av resultatet i hec106, innehåller ett fel. Finn felet och rätta det med en parentes. Cellerna från 180 och framåt kan utnyttjas.

```
...
114   10070
115   10232
116   00608
117   10463
118   10631
...
```

+) Ordet sekvens användes ibland synonymt med (mindre) kod.

5. OPERATIONSLISTA, DEL II.

Operation:	Grundform:	Verkningar:
Addera absolut	W OD	|w| + ar till AR
Subtrahera absolut	W OF	- |w| + ar till AR

Vad beträffar de härledda formerna och spill gäller detsamma som sagts om operationerna addera och subtrahera. Operationerna addera (resp. subtrahera) absolut och sätt i MR finnes ej vid Besk. Hur detta utföres se följande exempel.

Exempel 5.1: Antag att heol00 = x och heol02 = y. Sätt |x| - |y| i MR.
Svar: 200 1006D |x| till AR
201 1022F |x| - |y| till AR
202 00608 |x| - |y| till MR och kvar i AR

Exempel 5.2: Om heol00 och heol02 har samma tecken, sätt produkten av heol00 och heol02 i hecl04, om de har olika tecken, sätt |heol00-heol02| i hecl04.

Obs. Ordet 0 är det enda, vars negativa absolutbelopp är positivt, och ordet -1 det enda, vars positiva absolutbelopp är negativt.

Operation:	Beteckning:	Verkningar:
Skifta höger	k 04	ar skiftas (flyttas) åt höger k steg. I ARO inmatas kopior av den ursprungliga tecken siffran
Skifta höger, utan tecken	k 44	Samma som skifta höger, men i ARO inmatas en nolla efter varje skift steg
Skifta vänster	k 05	ar skiftar åt vänster k steg. I AR 39 inmatas nollor, d.v.s. AR 40 deltar ej i skiftet
Skifta vänster, med AR40	k 45	Samma som skifta vänster, men AR40 deltar i skiftet. Nollor inmatas i AR40.

k är ej en adress (skiftet berör endast ar och ej någon cell i W) och skall skrivas sedecimalt med enheten i pos. 11 resp. 31. Om $k \geq 64$ skiftas k"steg, där $k" \equiv k \bmod 64$, $0 \leq k" < 64$. De härledda formerna tolkas ej enligt regeln. 44 och 45 nollställer ej AR. Varianterna 24, 64, 25 och 65 användes vanligen ej. De har samma verkningar som resp. 04, 44, 05 och 45, men maskinen stannar om k är udda. Observera att ett skift k steg åt höger motsvarar en multiplikation med 2^{-k} och ett skift k steg åt vänster en med 2^{k}, förutsatt att AR:s kapacitet ej överskrides. Vid vänsterskift går ar0 över till AR00 i varje steg. Si kan uppstå, eftersom definitionen på si är att ar00 ≠ ar0 vid slutet av operationen. Det är inte säkert, att man får si vid vänsterskift mer än ett steg, även om AR:s kapacitet överskridits. Exempel: ar är i början av operationen 01100... (= 3/4). Efter ordern 00105 är ar = 11000... och ar00 = 0. Si finnes. Däremot är efter 00205 ar = 10000... och ar00 = 1. Si finnes ej, trots att AR:s kapacitet har överskridits. Vid högerskift går ar39 över till AR40. Vid alla operationsformer, som kan förändra ar, måste man räkna med att ar40 kan förstöras, varför operationsformen 45 har mycket begränsad användning. Si får man även efter högerskift av ett negativt tal. Orderna 00004 och 00005 användes som blindorder, d.v.s. order som ingenting gör. De förstör ej si, vilket däremot addition av 0 till AR gör.

Exempel 5.3: Multiplicera heol00 med 10 (används vid översättning av tal från binär till decimal form).

Maskinen kan inte lagra 10 eller placera det i multiplikatorn, då det ligger utanför intervallet $-1 \leq x < 1$. Multiplikation med 10 kan emellertid utföras genom att först multiplicera med 10/16 och sedan skifta resultatet 4 steg vänster (multiplikation med 16). Kalla heol00 för x.

```
Svar:    200 50000    konstanten 10/16
Start→   201 20048    10/16 till AR och till MR
         202 10063    10/16 x avrundat till AR
         203 00405    10x till AR
         204 10031    10x till 100
```

Exempel 5.4: x, y, z är lagrade i hac100, 101 och 102. Lagra teckensiffran och de 12 första binära siffrorna av vardera x, y och z i positionerna 1-13, 14-26, 27-39 resp. i heclFE. I pos. 0 skall teckensiffran för x stå.

Svar:

```
200 10050
201 01B04      skifta 27 steg höger
202 01A05      "  26  "  vänster
203 1FE31
204 10150
205 00744      skifta 7 steg höger (00704 skulle gå lika bra)
206 00D05      "  13  "  vänster
207 1FE30
208 1FE31
209 10250
20A 01B44      skifta 27 steg höger utan tecken
20B 1FE30
20C 1FE31
```

Exempel 5.5: vhaol00 = x, hhaol01 = y, vhaol02 = z, hhaol03 = u.
a) Bilda xy+zu i vhaolFE. Resultatet skall vara korrekt avrundat till 19 bitar.
b) Bilda $|x - y| - |z - u|$ i hhaclFF.

Svar: a) Tag hec008 till arbetscell. Lagra 2^{-20}, d.v.s. sedecimalt 80000 i hhacl05.

```
200 10150
201 01405
202 00608      y till MRv
203 10042      xy, hela produkten i AR
204 00831
205 10350
206 01405
207 00608      u till MRv
208 10242      zu, hela produkten i AR
209 00830      xy + zu med 38 siffror i AR
20A 10510      avrundning till 19 siffror
20B 1FE11      xy + zu, avrundat, i 1FE
```

Operation:	Beteckning:	Verkningar:
Normalisera	irr 15	utan AR40
		se nedan f.ö.
	irr 55	med AR40

ar skiftas till dess innehållen i AR0 och AR1 är olika, förutsatt att $ar \neq 0$. Efter operationen är då antingen $\frac{1}{2} \leq ar < 1$ eller $-1 \leq ar < -\frac{1}{2}$. AR40 deltar i skiftet vid 55 men ej vid 15. MR nollställes först, sedan insättes där an-

talet utförda skiftsteg med enheten i pos. 31. Om ar = 0 göres 63 skiftsteg. De härledda formerna tolkas ej enligt regeln. Varianten 35 har samma verkningar som 15 och varianten 75 samma som 55. De användes vanligen ej, eftersom maskinen stannar, om adressdelen är ett udda tal.

Exempel 5.6: x är lagrat i hec100, y i hec102. Bilda xy i formen $z \cdot 2^p$, z i hec31E, p i hhac321 med enheten i pos. 31. z skall vara normaliserat.

Svar:

```
200 10070
201 00015
202 00831    x normaliserat till 008
203 00442
204 32111    minus normaliseringsexponenten till 321
205 10270
206 00015
207 00A31    y normaliserat till 00A
208 00442
209 32110
20A 32111    minus normaliseringsexponenten adderad till hao321
20B 00868
20C 00A63    z till AR
20D 00015
20E 31E31    z normaliserat till 31E
20F 00442
210 32110
211 32111    p till 321
```

Koden kan göras snabbare, om man tar hänsyn till att produkten av två normaliserade tal är absolut icke mindre än 1/4. Utför detta.

Exempel 5.7: Samma som 5.6 men bilda $\frac{x}{y}$ i stället.

Operation:	Grundform:	Verkningar:
Lagra i adressdel	W 07	I W:s adresspositioner sätts innehållet i AR:s motsvarande positioner

De härledda formerna tolkas enligt regeln. Adresspositionerna är vid 07 och 47 pos. 0-11 eller 20-31, vid 27 och 67 pos. 0-11 och 20-31. Övriga pos. i W lämnas ostörda. Vid 07 och 27 ändras ar ej. Vid 47 och 67 nollställes hela AR först, och följaktligen nollställs även W:s adresspositioner.

Exempel 5.8: hao100 = W 00. Nollställ hecW.

Svar:
```
200 10050    W·2^-11 till AR
201 20207    W till hac202 adresspos.
202 00071
```

Order 201 ändrar hac202 till W71, så att när kr hunnit till 202, nollställes hecW och inte hec000.

Exempel 5.9: hao101 = W 00. Nollställ hecW.

Operation:	Beteckning:	Verkningar:
Addera och lagra i cell	W 06	w + ar till
	W 26	AR och till W
Öka adressdel	W 46	w + 0020000200 till AR
		w + 00200 till W
	W 66	w + 0020000200 till AR
		och dito till W

De härledda formerna tolkas ej enligt regeln. 06 och 46 är halvords-, 26 och 66 helordsoperationer. Operationerna 46 och 66 är väsentliga vid ordermodifikation (jfr kap. 1). Observera att heo 00200 00200 adderas till AR både vid 46 och 66. Operationen 06 utför i en order vad som kan utföras i två med 10 och 11.

Exempel 5.10: Förkorta exempel 5.6 genom att använda 06-ordern.

Svar:
```
200 10070
201 00015
202 00831
203 00442
204 32111
205 10270
206 00015
207 00A31
208 00442
209 32106    ersätter 32110 och 32111
20A 00868
20B 00A63
20C 00015
20D 31E31
20E 00442
20F 32106
```

Exempel 5.11: Gör sammalunda med ex. 5.4.

En följd av order kan vara avsedd att utföras ett upprepat antal gånger. (Ett program innehåller i själva verket alltid sådana följder. Att göra ett program, som endast innehåller order, som skall utföras en gång var, är slöseri med tid och pengar.) En sådan följd kan innehålla en grupp av order, som modifierar andra order i samma följd. Varje gång orderföljden åtlydes, kommer en ny och skild följd av beräkningar att utföras. På detta sätt är det möjligt att använda en cykel, som upprepas, för att utföra beräkningar, som vid första påseendet ej synes vara lämpade för en sådan behandling. Fördelen med att göra detta är, att programmen kan konstrueras med mycket färre order än annars skulle vara nödvändigt och därför fordrar mindre lagringsplats. En annan fördel är, att man kan använda samma program för likartade beräkningar, som endast skiljer sig i antalet gånger cykeln upprepas. Ett program för t.ex. beräkning av värdet av ett 9-gradspolynom kan med små ändringar användas för ett polynom av godtyckligt gradtal. Låt oss först betrakta ett enkelt exempel utfört med rak kodning utan någon cykel.

Exempel 5.12: heol00 = a_0, heol02 = a_1, heol04 = a_2, heol06 = a_3; heo008 = x. Beräkna $f(x) = a_0x^3 + a_1x^2 + a_1x^2 + a_2x + a_3$ och lagra resultatet i 184. Spill under räkningarna antages ej inträffa.

Svar: Skriv f(x) som $((a_0x + a_1)x + a_2)x + a_3$ (Horners schema).

200	10068	a_0 till AR och till MR
201	00863	a_0x till AR
202	10228	$a_0x + a_1$ till AR och till MR
203	00863	$(a_0x + a_1)x$ till AR
204	10428	$(a_0x + a_1)x + a_2$ till AR och till MR
205	00863	$(a_0x + a_1)x + a_2)x$ till AR
206	10628	$(a_0x + a_1)x + a_2)x + a_3$ till AR (och till MR)
207	18431	" " " till 184

Man ser, att vid beräkningen av ett polynoms värde får man en följd av orderpar

00863
$W(a_n)$28

som alla har samma form och endast skiljer sig genom olika $W(a_n)$. Man kan utnyttja detta genom att låta maskinen utföra dessa båda order ett upprepat antal gånger och låta maskinen själv ändra (modifiera) $W(a_n)$ mellan varje upprepning. Efter varje modifiering låter man maskinen testa, om den hunnit till den sista koefficienten. Hur detta göres se följande exempel.

Exempel 5.13: heo100 = a_0, ... heo112 = a_9; heo008 = x. Beräkna f(x) = $a_0x^9 + a_1x^8 + \ldots + a_8x + a_9$ och lagra resultatet i hec184. hao180 = 10000, hao182 = 11300. Dessa hao upplyser, var koefficientföljden börjar resp. slutar. Spill under räkningarna antas ej inträffa. Hoppa till 300 när beräkningen är klar.

Svar: Använd Horners schema. Detta är liktydigt med att använda rekursionsformeln

$$y_n = y_{n-1}\cdot x + a_n \text{ med } y_{-1} = 0, y_9 = f(x)$$

	200 18050	10000 till AR
	201 20407	100 till hac204:s adresspos.
	202 00648	0 till MR, d.v.s. $y_{-1} = 0$
→	203 00863	$y_{n-1}\cdot x$ till AR
	204 FFF28	$y_{n-1}\cdot x + a_n$ till MR ($y_n \rightarrow y_{n-1}$)
	205 20446	$W(a_{n+1}) \rightarrow W(a_n)$
	206 1820B	$W(a_n)$ - 113 till AR
←	207 2034E	Hoppa om $W(a_n) \leq 112$
	208 00001	mr till AR
	209 18431	$y_9 = f(x)$ till 184
←	20A 3000C	Uthopp

Anm. → utläses "ersätter".

FFF brukar användas för adresser, som modifieras under räkningarnas gång. Det har bl.a. den fördelen, att maskinen stannar, om modifieringen genom något misstag ej utföres. Adressen i ordern 204 blir under de successiva varven 100, 102, 104, ... 112. När efter den sista beräkningen adressen ökas till 114 blir differensen $W(a_n)$ - 113 = 114 - 113 positiv, och vid ordern 207 utföres ej hoppet tillbaka till 203, utan nästa order blir 208. Motiveringen för att låta maskinen sätta adressen 100 i hac204 i stället för att direkt i programmet skriva 10028 är, att programmet är att betrakta som ett delprogram av ett större program, där det användes flera gånger. När man återvänder till det en andra gång kommer hao204 att vara 11428 och den rätta begynnelseadressen 100 måste alltså återställas.

Exempel 5.14: Beräkna $a_0b_0 + a_1b_1 + \ldots + a_{20}b_{20}$ och lagra resultatet i hec30E. Hoppa sedan till 380. a_i är lagrat i hec(100 + 2i), b_i i hec(200 + 2i).

Svar:

```
300 10000   100 = a_i:s begynnelseadress
301 20000   200 = b_i:s        "
302 12900   129 = a_i:s slutadress + 1
303 30070   beg. för a_i och b_i till AR
304 30627   sätt begynnelseadresser i hec306
305 30E71   0 → S
306 FFF68   a_i till MR
307 FFF63   a_i b_i till AR
308 30E26   S + a_i b_i → S
309 30666   Öka adresserna i hec306 med 2
30A 3020B   W(a_i) - 129
30B 3064E   Hoppa om W(a_i) ≤ 128
30C 3800C   Uthopp
30D 00000
30E 00000   Lagring av summan ( S )
30F 00000
```

Jfr med exempel 4.9 i kapitlet 4. Ibland är det lämpligt att ha ett speciellt heltal för varvräkningen i cykliska program. Jfr ex. 6.5.

Exempel 5.15: Nollställ helcellerna 100 - 1FE.

Exempel 5.16: Skifta vart och ett av heol00, heol02,... heolFE så mycket åt vänster, som det största talet tillåter, och lagra antalet skift i hec380.

Operation:	Grundform:	Verkningar:
Extrahera	W 00	Resultatet kommer i AR och har ettor i de positioner där både w+ar och mr hade ettor; nollor i alla andra positioner. 0 till MR.

De härledda formerna tolkas enligt regeln, så att vid formerna 40 och 60 jämföres w och mr. Operationen kan användas för att extrahera siffrorna i vissa bestämda positioner i en cell i minnet.

Exempel 5.17: Talen x, y, z är 13-siffriga binära tal lagrade respektive i positionerna 1-13, 14-26, 27-39 i heclFE. Första siffran i vardera talet är teckensiffra. Stoppa maskinen om alla tre talen är positiva.

Svar: Antag att helordet 40020 01000, som har ettor i positionerna 1, 14 och 27, för övrigt nollor, är lagrat i hec100. Använd 008 som arbetscell.

```
200 1FE68    x, y, z till MR
201 10060    extraktion av innehållet i pos. 1, 14 och 27
202 00831
203 0086F    ar blir positiv endast om heo008 = 0. (Jfr avsnitt 2.4.)
204 2064E    fortsätt om heo008 ≠ 0
205 00130    stopp om heo008 = 0
```

Lägg märke till metoden att testa om heo008 är noll.

6. OPERATIONSLISTAN, DEL III.

Fig. 5 sid. 9.9 visar utseendet på den remsa som används av BESK för in- och utmatning av information. På figuren finns en rad små hål något ovanför mitten på remsan. Dessa hål används som styrhål vid kopiering och utskrivning av remsa i stansutrustningen och saknar informationsvärde vid inläsning i Besk. Bortsett från dessa hål kan 5 hål stansas i bredd på remsan. Om ett hål finns stansat i nedre kanten säges det befinna sig i kanal 5. Läget av kanal 1, 2, 3 och 4 framgår av figuren. Den information som dessa 5 hål representerar kallas en rad. Beroende på vilken inläsningsoperation som användes kan Besk antingen

1) hoppa över alla rader där 5:te kanalen inte är stansad och tolka övriga rader som sedecimala siffror mellan 0 och F på så sätt att kanal 1 till 4 tolkas som binära positioner; kanal 4: enhetsposition, kanal 3: 2-talsposition o.s.v. Ett hål motsvarar den binära siffran 1,avsaknad av hål den binära siffran 0. Stansning av sedecimala (decimala) siffror enligt detta system sker med hjälp av stansutrustning som finns beskriven i kap. 9 sid. 9.11-12. Även Besk kan stansa remsa (se operation 1C och 1D nedan och sid 9.11).
2) Läsa in varje rad som inte är blank (ostansad) som ett 5-siffrigt binärt heltal där kanal 5 motsvarar enhetspositionen. Man får på detta sätt möjlighet att läsa in remsa stansad enligt telexsystemet.

För inläsning enligt punkt 1) finns följande operationer:

Operation:	Form:	Verkningar:
Läs helord från remsa	W 19 W 39	Ett helord (10 rader med kanal 5 stansad) läses från remsa till AR. Därifrån sändes ar som hao eller heo till W. Det inlästa heo står även kvar i AR.
Läs 4-kanalsrad från remsa	W 59 W 79	AR nollställes och en rad med kanal 5 stansad läses från remsa till AR pos 36-39. Kanal 4 till pos 39, kanal 3 till pos 38 o.s.v. Sedan sändes ar till W som hao eller heo. Den inlästa raden står kvar i AR. (Om W är vhac inläses 0 till W.)

Vid helordsinläsning spolas remsa fram ända tills 10 rader med kanal 5 stansad har påträffats. Den första påträffade raden går till AR pos. 0-3 nästa till pos. 4-7 o.s.v. Vid läsning av 4-kanalsrad spolas remsa fram tills en rad med kanal 5 stansad har påträffats.

För inläsning enligt punkt 2) finns följande operation:

Operation:	Form:	Verkningar:
Läs 5-kanalsrad från remsa	W 74	AR nollställes och en rad med något hål stansad läses till AR pos 35-39. Kanal 5 till pos 39, kanal 4 till pos 38 o.s.v. Sedan går ar till hec W.

Remsa spolas automatiskt fram tills en rad med något hål stansat har påträffats. Obs. att W betecknar helcell. Om W udda inträffar stopp.

Telextecken på remsa:

kanal	1	2	3	4	5		
	.	.	.	0	.	vagnretur	
	.	0	.	.	.	ny rad	
	.	.	0	.	.	mellanslag	
	0	0	0	0	0	bokstavsskift	
	0	0	.	0	0	sifferskift	
	0	0	.	.	.	A	-
	0	.	.	0	0	B	?
	.	0	0	0	.	C	:
	0	.	.	0	.	D	Vem där
	0	.	.	.	.	E	3
	0	.	0	0	.	F	Å
	.	0	.	0	0	G	Ä
	.	.	0	.	0	H	Ö
	.	0	0	.	.	I	8
	0	0	.	0	.	J	Klocka
	0	0	0	0	.	K	(
	.	0	.	.	0	L	)
	.	.	0	0	0	M	.
	.	.	0	0	.	N	,
	.	.	.	0	0	O	9
	.	0	0	.	0	P	0
	0	0	0	.	0	Q	1
	.	0	.	0	.	R	4
	0	.	0	.	.	S	apostrof
	.	.	.	.	0	T	5
	0	0	0	.	.	U	7
	.	0	0	0	0	V	=
	0	0	.	.	0	W	2
	0	.	0	0	0	X	/
	0	.	0	.	0	Y	6
	0	.	.	.	0	Z	+

Det hittills vanligaste systemet för inläsning av remsa är inläsning med operationerna W 39, W 19 och W 59 (W 79). I det följande kommer vi att begränsa oss till att endast illustrera denna typ av inläsning. Emellertid finns i programbiblioteket sekvenser för inläsning och stansning av decimala tal enligt telexsystemet.

Exempel 6.1: Inläsning av halvord med hjälp av den permanenta konstanten 00839 i hac007. Innehållet i hac101 skall ändras till 1300C och hac208 till 21A71.

Man stansar remsan:

```
10119
0070C
00000
1300C
20819
0072C
21A71
00000
```

Sätt manuellt 007 i AS och tryck på "start med hopp".

	KR	ASOP	AR	Minnet	
start →	007	00839	10119 0070C	10119 0070C	till hec008
	008	10119	00000 1300C	1300C	till hac101
	009	0070C	00000 1300C		
	007	00839	20819 0072C	20819 0072C	till hec008
	008	20819	21A71 00000	21A71	till hac208
	009	0072C	21A71 00000		

Maskinen stannar på ordern 0072C, som är en stopporder (udda adress + helordsoperation).

Man kan alltså enligt ovanstående exempel få inläst ett program till ferritminnet från pappersremsa under förutsättning att från början i minnet står den permanenta konstanten (ordern) 00839 i hac 007. Detta sätt att läsa in till minnet kallas primitiv inläsning. Metoden är emellertid obekväm för inläsning av längre program. I följande exempel 6.2 visas hur en inläsningssekvens för program fungerar. Inläsningssekvensen antas redan stå i minnet plats 007 - 00E då inläsningen skall ske. I exempel 6.3 visas hur man får in inläsningssekvensen i minnet. För närmare upplysningar om remsa, stansapparatur och inläsning av program se kap. 9.

Exempel 6.2: Inläsning av sedecimal sekvens med hjälp av inläsningssekvens A40 som förutsättes ligga i maskinen.

Sekvensen 100 20070
101 20230
102 20431
103 00000

skall läsas in i minnet.

Man stansar remsan:

10039	Etikett som anger
10300	plats i minnet
20070	sekvens
20230	
20431	
00000	
0072C	slutetikett
00000	

A40 Kod.

007	00839
008	00000
009	00000
00A	00846
00B	01444
00C	8090B
00D	0084E
00E	0070C
00F	00000

Om man efter start i 007 följer koden order för order erhålles följande schema:

KR	ASOP	AR	Förändringar i minnet
007	00839	10039 10300	10039 10300 till hec 008
008	10039	20070 20230	20070 20230 till hec 100
009	10300	oväsentligt	
00A	00846	10239 00200	10239 till hac 008
00B	01444	00000 10239	
00C	8090B	FFFFF FFF39	Obs. att hac 809 = hac 009
00D	0084E	FFFFF FFF39	Obs. att ar negativt
008	10239	20431 00000	20431 00000 till hec 102
009	10300	oväsentligt	
00A	00846	10439 00200	10439 till hac 008
00B	01444	00000 10439	
00C	8090B	00000 00139	
00D	0084E	00000 00139	Obs. att ar positivt
00E	0070C	00000 00139	
007	00839	0072C 00000	0072C 00000 till hec 008
008	0072C	0072C 00000	stopp

Maskinen stannar på ordern 0072C och sekvensen är då inläst till plats 100-103.

Exempel 6.3: A40 självinläsande remsa (utan nollställning av ferritminnet): Om man vill läsa in sekvensen i exempel 6.2 utan att förutsätta att A40 redan finns inläst stansar (kopierar) man följande helord på remsa:

början	
00146	8090B
00239	0084E
0000E	0070C
00E19	00001
0001D	00200
00110	00200
0001C	00100
0005D	00100
00039	80000
00700	00001
00846	00000
01444	00839
	slut

Efter dessa helord stansas sekvensen enligt exempel 6.2 med etikett före och efter.
Först lägger man remsan i remsläsaren. Därefter trycker man på knappen 0 → MS (eller KASOP) på kontrollbordet. Man får då 000 till KR och 00000 till ASOP. Sedan trycker man på knappen "start från remsa" vilket medför att OP sättes = 39 och maskinen startar. I ASOP kommer då först att stå ordern 00039 vilket medför inläsning av första ordet på remsan 00146 00239 till hec 000. KR innehöll först 000 men ökas nu med 1 och nästa gång utföres ordern 00239 som står i 001 efter inläsningen. Nästa ord från remsan hamnar alltså i hec 002 o.s.v. Med ett visst mått av tålamod kan man följa vad som sker i fortsättningen. När det första avsnittet av remsan är inläst står sekvens A40 + de permanenta konstanterna i minnet. Under inläsningen har ett A skrivits ut på skrivmaskin. Sedan sker automatiskt inläsning enligt exempel 6.2 av resten av remsan där sekvensen är stansad.
Den vanliga versionen av A40 nollställer ferritminnet före inläsningen (se kap. 9).

Exempel 6.4: Decimal inläsning av ett positivt tal. Efter sista siffran stansar man ett D, som betecknar plustecken.

```
100   00648   0->MR
101   00659   rad från remsa till pos. 36, 37, 38, 39 i AR, som
102   1050B     som först nollställts; ar_v = 0 till hac006
103   1070E   test om siffra eller D
104   10B10   den senast inlästa siffran i AR
105   10B02   10·2^-39·mr + 2^-39·ar, se op.listan och nedan
106   10106
107   00001   mr till AR
108   10031   lagra talet i hec10C
109   0072C   stopp
10A   00000
10B   0000A   10·2^-39
10C   00000
10D   00000
```

Om n:te siffran betecknas x_n, så bildas heltalet y_n av de n första siffrorna enligt formeln

$$y_n = y_{n-1} \cdot 10 + x_n$$

och lägges med enheten i pos. 39 i MR. Det slutgiltiga resultatet av översättningen måste alltså skiftas in i AR. För allmännare decimala inläsningssekvenser, se kap. 9. Ett D stansat efter talet betecknar plustecken, ett B minustecken. Stansade F överhoppas. En felaktigt stansad siffra kan således "strykas" genom att förvandlas till F. Detta är möjligt, eftersom tecknet F har hål i alla kanaler på remsan.

Operation:	Form:	Verkningar:
Tryck eller stansa siffra	1 C	Om skrivmaskinen är inkopplad, trycks ar:s fyra sista bitar som en sedecimal siffra. Om stansen är inkopplad, stansas nämnda siffra på pappersremsa. Även kanal 5 på remsan stansas. Beträffande remsans utseende, se kap. 9.3.
	5 C	AR nollställes och siffran 0 tryckes i skrivmaskin eller stansas på remsa. F.ö. som 1 C.

Tryck eller stansa tecken	1 D	Om skrivmaskinen är inkopplad, framkallar ar:s fyra sista bitar ett typografiskt tecken eller en typografisk operation. Om stansen är inkopplad, stansas AR pos. 36-39 i kanal 1-4 på remsan. Remsans kanal 5 stansas ej. Se sid. 9.9!
	5 D	AR nollställes och mellanslag utföres eller stansas på remsa.

De härledda formerna tolkas enligt regeln. För såväl 1 C som 1 D gäller att endast ar är av betydelse. Adressdelen är i bägge fallen irrelevant. Beträffande de sedecimala siffrornas betydelse typografiskt vid 1D-ordern, se operationslistan! Andra hålmarkeringar än där angivna kan genom 1D-ordern stansas på remsa men framkallar ingen typografisk operation i skrivmaskin.

Med hjälp av operationerna 1C och 1D tillsammans har man möjlighet att stansa vilken hålkombination som helst på remsa. Man kan alltså stansa information enligt telexsystemet om man så önskar.

Exempel 6.5: Decimal tryckning av ett positivt tal < 1 utan avrundning med 6 decimaler. x förutsättes ligga i MR.

```
108  00550   2^-39 till AR
109  0001D   vagnretur
10A  00A51   i = 0  (antalet tryckta siffror)
10B  11342   mult. med 10 · 2^-39
10C  0001C   tryck decimal siffra
10D  00A46   (i + 1).2^-10 till 00A
10E  1140B   minus 6.2^-10
10F  10B4E   hopp tillbaka om i mindre än 6
110  ???0C   till fortsättning av programmet
111  00000   oanvänt
```

112 00000
113 0000A
114 00C00 antalet decimaler·2^{-10}
115 00004

Varvräkningen kan också utföras så,att man i 00A lagrar $-6 \cdot 2^{-10}$ och ökar med 2^{-10} för varje tryckt decimal,tills hac00A = 0. Detta ger en order mer i förberedelserna till cykeln men en order mindre i själva cykeln.

Operation:	Form:	Verkningar:
Skriv på trumma	1 F	Halvorden W, W+002, W+004,... W+K-002 går successivt till MD, vars andra halva utfylles med nollor, varpå helorden via AR går in på den trumkanal, vars nummer man i en tidigare order placerat i MR med enheten i pos. 31.
	3 F	Helorden W, W+002... W+K-002 går via AR in på den trumkanal, vars nummer man i en tidigare order placerat i MR med enheten i pos. 31.
Läs från trumma	1 B 3 B	Helorden i den trumkanal, vars nummer man i en tidigare order placerat i MR med enheten i pos. 31, går successivt till AR och därifrån till halv- resp. helcellerna W, W+002, W+004,... W+K-002. Se op.listan.

Operationerna 1 F resp. 3 F har i praktiken exakt samma verkan som op. 5 F resp. 7 F. Detsamma gäller för 1 B (= 5 B) och 3 B (= 7 B). Operationerna 5 F, 7 F, 5 B och 7 B inledes med en överflödig nollställning av AR. (Jfr mikrooperationsschemat, kap. 13.)

Trumminnets 256 kanaler är numrerade 000, 002, ... 1FE. Vid överföringar till och från trumma avkänns endast positionerna MR23 - MR30, de övriga positionerna i MR är irrelevanta. Så t.ex. om det står 20304 i MRH läses (skrives) kanal 002, och om det står 7FE11 i MRH så läses (skrives) kanal 1FE.

Exempel 6.6: Låt innehållet i kanalerna 008 och 012 byta plats på trumman. Två kanalers utrymme får användas och förstöras i kärnminnet men ingen ytterligare kanal på trumman.

```
100  10948
101  2003B
102  10B48
103  2403B
104  2003F    observera att trumkanalnumret ligger kvar i MR efter
105  10948    läsning och skrivning
106  2403F
107  0072C    stopp
108  00000
109  00800
10A  00000
10B  01200
```

Exempel 6.7: Låt innehållet i kanalerna 008 och 012 byta plats på trumman utan att förstöra mer än en trumkanals utrymme och utan att använda någon annan kanal på trumman än 008 och 012. Det är tillåtet att använda mer utrymme i kärnminnet, förutsatt att man återställer det ursprungliga innehållet.

Exempel 6.8: Gör en sekvens som söker reda på det n:te ordet i kanal k och lägger detta i AR. Överväg olika möjligheter att förse koden med den behövliga informationen (n, k).

Operation:	Form:	Verkningar:
Sänd till funk-	00018	ar till F:s x-register. Skriv ej.
tionsskrivare	/00218/	-"- -"- -"-
	00418	-"- Skriv. Punkt.
	00618	-"- -"- Cirkel.
	00818	ar till F:s y-register. Skriv ej.
	/00A18/	-"- -"- -"-
	00C18	-"- Skriv. Punkt.
	00E18	-"- -"- Cirkel.

ar betyder här teckensiffran tagen separat samt en följd av åtta successiva bitar. Följden kan börja i någon av pos. 1, 2, 3, 4, 5, 6, 7 i ar beroende på läget hos omkopplaren "Position X" resp. omkopplaren "Position Y" på manöverbordet. För att 18-operationen skall fungera, måste F vara tillslagen (manuellt).

Kurvorna bestå av diskreta punkter, som erhålles genom att sända värdepar x, y eller y, x till F. Ett givet x-värde kvarstår i x-registret obegränsad tid, tills ett nytt x-värde sändes dit oberoende av y-registret. Motsvarande gäller för y-registret. Man kan därför rita flera kurvor samtidigt med samma abskissavärde men med olika ordinatavärden för olika kurvor (eller vice versa), vilket gör ritandet snabbare.

Felet i de koordinater, som sänds till F, ligger således mellan 0 och -2^{-8}. Funktionsskrivarens bildfält är försett med ett punktraster motsvarande punkterna x, y = 0, $\pm 1/4$, $\pm 1/2$, $\pm 3/4$, ± 1. Eftersom detta raster är förskjutet till vänster och nedåt en sträcka, som motsvarar 2^{-9} i F-registren, blir felet i koordinaterna hos en punkt på bildskärmen relativt rastret absolut högst 2^{-9}, alltså approximativt 2 promille. Till detta teoretiska fel kommer tekniska avlänkningsfel i elektronstrålen (när rastret och punkten fotograferas vid olika tidpunkter); dessa fel är ofta större än förstnämnda. Därtill kommer en onoggrannhet på grund av bildfläckens storlek, i synnerhet vid större intensitet. Vid skrivning av små cirklar i stället för punkter minskas skärpan. Cirkeldiametern kan varieras manuellt.

F-operationen 18 innebär, att maskinen stannar för ett ögonblick för utmatning av ar till F. F avkänner dessutom i ordern de tre första bitarna i adressdelens sista sedecimala siffra, och en viss variant utföres med ar. Övriga pos. i orderns as-del irrelevanta.

Om man har kodat operationen 18, och F är frånkopplad, blir det alltid ett s.k. "mörkt stopp" i maskinen (se kap. 10). Man kan dock kortsluta tillslagen F manuellt genom vridning av ratten "cirkel" till vänster. Då tjänstgör 18-operationen som blindoperation (en additionstid). Obs! strömstöten vid till- och frånslagning av F påverkar kärnminnet.

Vridning av ovannämnda omkopplare innebär en förstoring av x- resp. y-koordinaten med en 2-potens. En alltför stark förstoring kan stycka bilden; det blir ett slags spill, dock utan spillindikation och kallas här för "översvämning i F". Även verkligt spill på grund av kodfel styckar bilden.

Exempel 6.9: ar = FCCCC..., binärt 1,111110011001l0..., decimalt $\approx -1/40$. Om

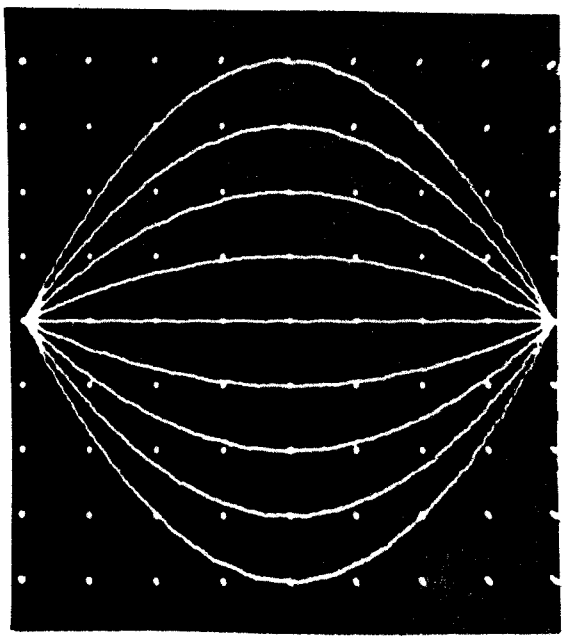
omkopplaren "Position X" eller "Y" står i pos. 6, sänds 1,00110011 ≈ -32/40 (dec.) till F. Om den står i pos. 7, sänds 1,01100110 ≈ -24/40 i stället för -64/40. I senare fallet fick man översvämning.

Vid kodande för funktionsskrivare bör man ordna det så, att kurvorna kan omtagas efter behag för fotograferingens skull.

Exempel 6.10: Rita kurvor över funktionen $a(x^2 - 1)$; $-1 \leqq x \leqq +1$ med nio olika a-värden; en kurva åt gången. Finaste punkttäthet i x-led. "Position X" och "Y" = 1. $\Delta x = 2^{-8}$ = 0080000000; W(a) = 11C, W(x) = 11E; a-värdena ligger på en remsa och tänkes numrerade n = 0, 2, ... 10 (sed.); n blir lagrad i adresspos. i vhac11A; $n_{max} + 1$ = 011 i adresspos. vhac100. Startorder 101.

```
      100  01100  gräns.
start 101  11A47  sätt n = 0.
      102  11C39  a från remsa.
      103  00470  x = -1 + 2^-39.
      104  11E31  dito till W(x).
      105  11E68  x till MR,
      106  00018  x till F, skriv ej.
      107  11E63  x^2 i AR,
      108  00408  x^2 - 1 i MR,
      109  11C63  y = a(x^2 - 1) i AR,
      10A  00C18  y till F. Skriv.
      10B  11870  Δx i AR,
      10C  11E26  nytt x,
      10D  10F0A  är x > +1 ?
      10E  1050C  annars räkna ny punkt.
      10F  0046B

      110  00018  skriv [x = +1 - 2^-39
      111  00C58  separat [y = 0 (obs.58-op.).
      112  10370  stopp (se nedan).
      113  11A46  öka n med 2,
      114  1000B  n - gräns,
      115  1024E  nytt a,om n<11(sed.);
      116  1012C  annars SLUT-stopp.
      117  00000  används ej.
      118  00800
      119  00000  Δx
```



11A, 11B, 11C, 11D, 11E, 11F är arbetsceller.

Order nr 112 är ett exempel på manuellt alternativhopp, som har intresse även i andra sammanhang: Trycker man på knappen "Start med hopp", utföres 1030C, och kurvan blir omtagen. Trycker man på "Start", blir den betydelselösa ordern 10350 (ej 70) utförd, och ett nytt a-värde ger en ny kurva eller det blir slutstopp. Figuren visar kurvorna tagna en efter en på samma film. a = 1, 3/4, 1/2, 1/4, 0.

Blandade exempel.

Exempel 6.11: Beräkna $x_1 + x_2 + x_3 + x_4 + x_5$ med varvräkning. x_1 i hec100, x_2 i hec102 o.s.v.

Exempel 6.12. Vad står i MR och AR efter operationerna

00549
00542

Exempel 6.13: Talen x_1, x_2, ... x_5 står i hec100, 102 o.s.v. Sätt det av talen, som har störst absolutbelopp, i hec200.

Exempel 6.14: Beräkna $\frac{1}{2}(x + \frac{a}{x})$, där a står i hec100, x i hec102 och a mindre än eller lika med x. Antag att spill inte kan uppkomma vid additionen av x och $\frac{a}{x}$. Talet a antas positivt liksom x.

Exempel 6.15: Hur kan man klara situationen, om spill skulle uppkomma vid beräkningen i exempel 6.14?

Exempel 6.16: Rekursionsformeln

$$x_{i+1} = \tfrac{1}{2}(x_i + \frac{a}{x_i})$$

$$x_0 = a$$

ger en talföljd, som konvergerar mot kvadratroten ur a(som antas positivt). Gör ett program, som beräknar x_i successivt och avbryter, då två på varandra följande x_i skiljer sig från varandra med ett belopp, som är mindre än en tolerans.

Exempel 6.17: Genom ett decimalt tal d stansat på remsa vill man välja mellan 7 olika vägar i programmet enligt följande:

d = 0	Hopp till	354
1		38D
2		39F
3		3C0
4		3E5
5		402
6		426

Skriv kod härför. Programmet kan börja på 101.

Lösning:

101	00659	d från remsa till AR
102	00805	d skiftas i adressposition
103	10D10	
104	10507	106 + d till 105
105	FFF0C	
106	3540C	
107	38D0C	

(forts. på nästa sida)

108	39F0C
109	3C00C
10A	3E50C
10B	4020C
10C	4260C
10D	10600

Exempel 6.18: Resultatet av en oavrundad multiplikation kommer som bekant med tecknet och de 39 första bitarna i AR, medan de 39 återstående bitarna kommer i MR. Position MR0 är alltid 0. Skriv en kod, som skiftar en dylik produkt p steg åt vänster. Innehållet i MR skall medfölja (s.k. långt skift). Resultatet sätts avrundat i helcell W.
Anvisning för lösning:

p	05
W	31
$W(2^{p-39})$	63
W	26

Exempel 6.19: $\overline{x}$ och $\overline{y}$ är besktal. Bilda $\overline{x}\cdot\overline{y}\cdot 2^{p}$ i AR med maximal noggrannhet och maximalt antal signifikanta siffror. Lagra p i valfri helcell.

Exempel 6.20: Resultatet av en oavrundad multiplikation ligger på angivet sätt i AR och MR. Skifta produkten fyra steg åt höger och lagra resultatet, så att tecknet och de första siffrorna kommer i hec100 och fortsättningen avrundat i hec102.

Exempel 6.21: Typografisk utformning av resultaten. Ändra och komplettera tryckkoden i exempel 6.5, så att man medelst snabbstansning kan trycka 500 tal absolut < 1, som ligger lagrade på trumman i kanalerna 000 och framåt. A4-format med tillräcklig plats för marginaler överst, till vänster och underst. Efter varje sida och efter sista talet skall två stopptecken (F utan femte hål) stansas på remsan ("remsstopp"). Efter sista talet skall Besk stanna automatiskt ("beskstopp").

Anvisningar: Koden skall ändras och kompletteras före vhac102 och efter hhac115. Vidare ändras stoppordern 111. Separat vagnretur före tryckning av första talet. Två mellanslag efter varje tal. A4-format fyllt från kant till kant rymmer c:a 80 tecken i sidled och c:a 70 rader med enkelt radavstånd i höjdled (utan avdrag för marginaler). Antag att man vid själva tryckningen ställer in vagnreturen manuellt på dubbelt radavstånd och efter tryckning av en sida gör manuell vagnretur fram till lämpligt startläge för nästa sida.

Varvräkning behövs för antalet tal per rad, antalet vagnreturer per sida, antalet ord per kanal och sammanlagda antalet tryckta tal.

Exempel 6.22: Tryck medelst snabbstansning talmaterialet i föregående exempel på s.k. brett papper i rulle (utan sidindelning). Gör två extra vagnreturer efter var femte rad. Använd tabulator och lämna plats för marginaler till vänster och höger. Remsstopp och beskstopp efter sista talet.

Anvisningar: "Brett papper" rymmer c:a 140 tecken i sidled från kant till kant. För att tabulatorn skall taga vid, måste det finnas ett avstånd om 2 tecken, helst 3-4 tecken, från sista tryckta tecknet till tabulatorstället. Placera tabulatorordern omedelbart efter tryckning av tal.

7. ORGANISATION AV DELPROGRAM. FLÖDESSCHEMAN. /ANPASSNING AV 800-SEKVENSER./

7.1. Flödesscheman. Ett längre program kan i allmänhet tänkas uppdelat i ett antal mer eller mindre fristående delprogram (subsekvenser). Dessa delprogram utför t.ex. decimal tryckning (stansning) av ett eller flera tal, beräkning av kvadratrot eller andra funktioner, inläsning och översättning av tal stansade decimalt på remsa eller något annat mer eller mindre ofta återkommande arbetsmoment under beräkningarna. Vissa av dessa operationer, t.ex. tryckning, ingår i så gott som alla program. För att underlätta kodningen har speciella standardprogram iordningställts för de vanligaste fallen (se kapitlet Översikt över programbiblioteket).

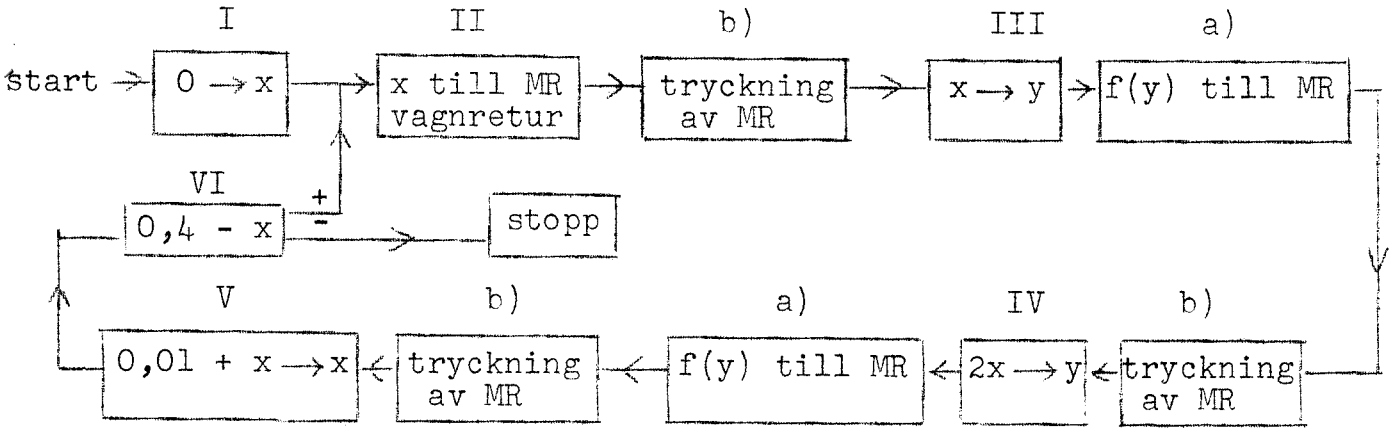
Innan man börjar koda ett problem, gör man lämpligen en mer eller mindre detaljerad uppdelning enligt ovan av programmet i delprogram. Beräkningsgången i stort mellan de olika delprogrammen kan åskådliggöras med ett flödesschema. Hur detta går till visas bäst med ett exempel.

Uppgift: tabellera funktionerna $f(x) = 0,5x^2 + 0,3x - 0,1$ och $g(x) = f(2x)$ för x varierande mellan 0 och 0,4 med steg på 0,01. Argumentet x jämte f(x) och g(x) skall skrivas på en rad; ny rad göres för varje nytt argument. Alla tal tryckas med 5 decimaler.

De väsentliga delprogrammen utgöras här av:

a) beräkning av funktionen f(y), resultat i MR
b) tryckning av innehållet i MR

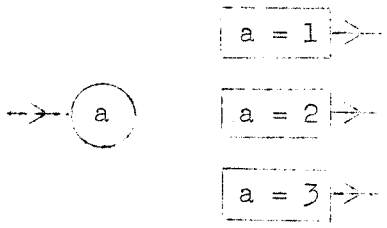
Flödesschema



Boxarna I, II, III, IV och V kallas substitutionsboxar. Pilarna inuti boxarna översätts med "ersätter" eller "insättes för". Box VI är en alternativbox - en sådan insättes, när fortsättningen beror av något villkor. Boxarna a) och

b), som f.ö. förekommer i flera exemplar i diagrammet (se nedan "Wheelerhopp") kallas operationsboxar; i sådana göras vanliga räkneoperationer ("produktivt" arbete i motsats till "administrativt").

Mera komplicerade villkor kan ritas så här:



vilket betyder, att programmet skall ta den ena eller andra vägen, då storheten a är resp. 1, 2 eller 3.

Boxarna a) och b) förekommer på flera ställen i diagrammet. Vid kodningen är det emellertid inte nödvändigt att upprepa delprogrammen a) och b) på flera ställen i koden. Det är tillräckligt, att subsekvenserna finns vardera i ett exemplar och att man hoppar till en subsekvens från huvudprogrammet (= det program som sammanbinder delprogrammen), då man har användning för den. Sedan subsekvensen är genomlöpt, vill man återgå till huvudprogrammet t.ex. vid ordern närmast efter den, där man lämnade det. (Detta återhopp sker naturligtvis inte utan att man vidtagit någon speciell åtgärd i huvudprogrammet.)

Nedan skall ett förfarande beskrivas, som är lämpligt i sådana situationer. Förfarandet i fråga, som går under namnet Wheelerhopp efter en engelsk kodare, användes i stor utsträckning i standardprogram för Besk. Olika varianter förekommer. Det normala Wheelerhoppet illustreras av följande exempel.

7.2. Normalt Wheelerhopp. Antag programmet till det förut beskrivna problemet börjar på order 040. De första orderna i programmet är följande:

kr	order	kommentar
040	W(x) 71	0 -> x
041	005 50	2^{-39} till AR
042	000 1D	vagnretur
043	W(x) 68	x till MR
044	000 04	blindorder (skift 0 steg)
045	045 50	order 045 till ARH
046	200 0C	nopp till tryckning av C(MR)
047	W(x) 70	
048	W(y) 31	x -> y

```
049     049 50
04A     240 0C      hopp till beräkning av f(y)
                    o.s.v.
```

Hoppet från huvudprogrammet till subsekvensen skall alltid ske från en order med jämnt nummer (order 046 och 04A i exemplet). Ordern närmast före hoppet (alltid udda; härav blindordern i 044) användes för att till AR föra information, som skall användas i subsekvensen, för att återhoppet till huvudprogrammet skall kunna ske till rätt ordernummer.

Antag mer generellt att man lämnar huvudprogrammet vid order n, där n är jämnt, och att b är startorder till subsekvensen (order 200 resp. 240 i exemplet). Man skriver i huvudprogrammet:

```
kr      order
n-1     n-1 50
n       b   0C
```

Order n-1 adderar högerhalvordet n-1 (där ordern själv står) till AR efter tömning. I AR kommer alltså att stå följande:

```
00000 n-1 50
```

Här representerar n-1 de 3 sedecimala (12 binära) siffrorna i adressdelen. Därefter utföres hoppordern b 0C, som ej påverkar innehållet i AR.

Första orderna i subsekvensen är:

```
kr      order
b       001 10
b+1      u  07
```

I order b adderas talet 2 till adressdelen i AR:s högerhalva. I AR kommer alltså att stå:

```
00000 n+1 50
```

Adressdelen u i nästa order anger platsen för den hopporder (0C, 0E, 4E eller 0A), varifrån återhoppet från subsekvensen till huvudprogrammet sker. Order u är alltid udda. Order b+1 kommer alltså att sätta n+1 i adressdelen i order u. Återhoppet sker alltså till order n+1 i huvudprogrammet.

Exempel: Subsekvens för tryckning av innehållet i MR med tecken och 5 decimaler. Sekvensen börjar på hac200.

```
kr      order       kommentar
200     00110       Wheeleradministration
201     21107
202     00001       x till AR
203     21631       x till hec216
```

204	2080E	är x pos. eller neg. ?
205	21669	-x till MR; endast om x< 0
206	21550	4.2^{-39} i AR
207	20A0C	
208	21668	x till MR; endast om x ≧ 0
209	20A4C	hopp med tömning av AR
20A	0001D	tecken tryckes (mellanslag eller minus)
20B	21651	0 till hac216
20C	21342	en dec. siffra till AR
20D	0001C	tryckning av siffra
20E	21646	adressen i hac216 ökas med 2
20F	2140B	5.2^{-10} subtraheras
210	20C4E	
211	0000C	uthoppsorder
212	00000	ej använd plats
213	0000A	10.2^{-39}
214	00A00	5.2^{-10}
215	00004	4.2^{-39}
216	00000	hec216: arbetscell
217	00000	

7.3. Modifierat Wheelerhopp. Utom det normala Wheelerhoppet förekommer även s.k. modifierat Wheelerhopp, som illustreras av följande exempel:

En sekvens trycker talet $10^h.x$, där x antas vara mindre än 1 i absolutbelopp och lagrat i hec W(x). Först skall h heltalssiffror tryckas, därefter decimalpunkt och d decimaler. I huvudprogrammet skriver man:

kr	order
n-1	n-1 50
n	b 0C
n+1	W(x) hd

Adressen n är jämn som förut. I hac n+1 står information som sekvensen behöver, näml. adressen till det tal, som skall tryckas, samt h och d (de 2 sista vardera 1 sedecimal siffra). Subsekvensen hämtar in och tolkar detta hao. Återhoppet sker automatiskt till order n+2. Order b är startorder till sekvensen.

Ett alternativt sätt att ge sekvensen information före hoppet vore t.ex. att placera halvordet W(x)hd i en speciell cell (eller register), densamma varje gång före hoppet till sekvensen. Den angivna metoden minskar emellertid ofta felriskerna och huvudprogrammet förenklas. Naturligtvis kan man konstruera sekvenser , där mer än ett halvord efter hoppordern används för att ge information.

I programbiblioteket förekommer även sekvenser med modifierat Wheelerhopp, för vilka adressen n måste vara udda. I alla de fall,där avvikelse föreligger från det normala Wheelerhoppet,finns detta angivet i beskrivningen till standardprogrammet.

Subsekvenser,som är arrangerade för något slag av Wheelerhopp,kallas slutna sekvenser. Motsatsen är en öppen sekvens d.v.s. en sådan som inte innehåller några speciella anordningar för återhopp.

Övning: Fullborda det program som påbörjats i det föregående.

7.4. 800-sekvenser. Anpassning. En fordran, som man måste ställa på de flesta sekvenser i programbiblioteket, är att de vid användningen skall kunna läggas på vilken plats i minnet man vill. Det är emellertid tydligt, att t.ex. den trycksekvens i hac200-217, som förekom i avsnittet 7.2, inte kan fungera,om den sättes oförändrad på någon annan plats i minnet. Å andra sidan är de ändringar, som måste göras, av synnerligen enkelt slag,och man kan genom att införa lämpliga konventioner beträffande sättet att skriva en sekvens uppnå,att maskinen själv kan anpassa den till en given plats i minnet. Denna anpassning utföres av vissa inläsningssekvenser (se kap. 9) i samband med inläsningen av programmet.

En anpassbar motsvarighet till trycksekvensen i avsnitt 7.2 fås på följande sätt: man skriver sekvensen,som om dess första halvord låg i den fiktiva cellen 800, nästa i 801 o.s.v. Koden blir:

800	00110	80C	81342
801	81107	80D	0001C
802	00001	80E	81646
803	81631	80F	8140B
804	8080E	810	80C4E
805	81669	811	0000C
806	81550	812	00000
807	80A0C	813	0000A
808	81668	814	00A00
809	80A4C	815	00004
80A	0001D	816	00000
80B	81651	817	00000

Sekvensen stansas (kopieras över) på programremsan i denna form föregången av vissa rubriker, som anger bl.a. var den skall ligga i minnet (beträffande praktiska detaljer, se kap. 9). Den läses först in i minnet i oförändrat skick

till sin rätta plats och anpassas därefter automatiskt av maskinen på följande sätt: för varje order undersöks,om dess adressdel överstiger eller uppgår till 800. (Den första binära positionen i adressen innehåller i så fall en etta). I varje sådan order subtraheras 800 från adressdelen,och sekvensens verkliga begynnelseadress i minnet adderas. Övriga order lämnas oförändrade. Resultatet blir en sekvens anpassad till den verkliga begynnelseadressen i minnet.

Det är tydligt, att maskinen endast skall behandla den del av sekvensen, som utgöres av verkliga order, inte den del,som innehåller konstanter och arbetsceller. I det angivna exemplet spelar det emellertid ej någon roll om maskinen skulle utsträcka anpassningen även till hao812-817,eftersom alla konstanter, som lagras där, har "adressdelar", som är mindre än 800. Det förekommer dock ej sällan, att en 800-sekvens innehåller konstanter, som har "adressdelar", som är större än eller lika med 800. Exempel: en sekvens innehåller konstanten 0,1 = OCCCC CCCCD i Beskform. Adressdelen CCC i högerhalvordet skulle komma att ändras vid anpassningen om denna utsträcktes även till konstanterna. Man har därför den konventionen,att alla konstanter, som skulle förstöras vid anpassning, lagras efter orderna (det egentliga programmet) i sekvensen och att anpassningen utsträckes över ett visst antal halvord från sekvensens början på 800 fram till och med ett visst högerhalvord. Information om 800-adressen till detta högerhalvord hämtas in från remsa vid inläsningen av sekvensen. (Att gränsen för anpassningen går just efter ett högerhalvord är en ren konvention, som medför en viss förenkling av den sekvens i inläsningsprogrammet, som utför anpassningen.)

Att Wheelerhopp användes i exemplet har naturligtvis inte någonting med 800-formen att göra. Det är inte heller väsentligt,att programmets startorder står på plats 800. Man behöver inte ens fordra, att plats 800 innehåller en verklig order. Det är däremot väsentligt, att plats 800 svarar mot det första halvordet, som ingår i sekvensen (arbetscell, order eller konstant).

Om ett stort antal arbetsceller förekommer sist i sekvensen, kan det vara lämpligt att inte medtaga dessa vid stansningen av sekvensen. I exemplet är det alltså inte nödvändigt att stansa ut det sista helordet (816) med nollor på remsan. Under alla förhållanden måste dock sekvensen sluta med ett högerhalvord på remsan.

Sammanfattning: Följande konventioner gäller för anpassbara sekvenser (800-sekvenser):

I. Sekvensen skrives som om dess första halvord låg i den fiktiva cellen 800, nästa i 801 o.s.v.

II. Konstanter, som skulle förstöras vid anpassning, lagras efter orderna i sekvensen. Anpassningen utsträckes från 800 fram till och med ett visst högerhalvord.

III. Då sekvensen är stansad på remsa, skall den sluta med högerhalvord.

Terminologi:
Totallängd för en sekvens är adressen i 800-form för det sista halvordet i sekvensen (i exemplet är totallängden = 817).
Remslängd kallas adressen i 800-form för det sista på remsan medtagna halvordet (i exemplet blir remslängden = 815 om det sista helordet ej stansas).
Anpassningslängd kallas adressen i 800-form för det sista halvordet i sekvensen, dit anpassningen utsträckes (i exemplet kan anpassningslängden sättas = 811, 813, 815 eller 817 efter behag).

Totallängd, remslängd och anpassningslängd bifogas på remsan vid inläsning (se kap. 9).

Övning: Skriv sekvensen för beräkning av f(y) i föregående problem som 800-sekvens. Låt konstanterna 0,5, 0,3 och 0,1 ingå i sekvensen.

7.5. Program på trumman. Om ett program är långt eller en mycket stor del av snabbminnet måste reserveras för lagring av data, kan det bli nödvändigt att använda trumman för lagring av program. Programmet uppdelas därvid på lämpligt sätt i ett antal programdelar. Var och en av dessa antas vara så kort, att den ryms i sin helhet i snabbminnet. Programmet lagras vid inläsningen till en större eller mindre del (ofta hela programmet) på trumman. Under räkningarna står endast en eller ett par av programdelarna inne i snabbminnet åt gången, nämligen den eller de, som är aktuella. Då maskinen behöver nytt program i snabbminnet nedläses den nya programdelen från trumman genom hopp (automatiskt) till en särskild sekvens (som i allmänet får stå permanent i snabbminnet). Uthoppet från sekvensen, när de aktuella kanalerna är nedlästa, sker sedan till startordern för den nya programdelen, och maskinen fortsätter i denna.

För överföring av program (och data) i båda riktningarna mellan trumma och snabbminne kan användas standardprogram D 05 eller D 06. Båda dessa standardprogram förutsätter, att varje programdel upptar en viss sammanhängande följd av celler i minnet från säg den jämna adressen bbb i minnet till den udda adressen sss. Programdelen lagras på trumman i ett antal på varandra följande kanaler med början på t.ex. kanal kkk. Det första helordet i programdelen lagras på den första platsen i kanal kkk. Man har alltså de 32 första helorden i kanal kkk, de 32 nästa i kanal kkk+2 o.s.v. I den sista kanalen kommer eventuellt oanvänd plats att finnas i slutet av kanalen. En programdel eller en mängd data, som är lagrad på trumman på detta sätt, kallas i det följande för ett block. Då ett block skall läsas ned från trumman med hjälp av D 06 skriver man i huvudprogrammet

kr	order	
n	n 50	order n alltid jämn
n+1	014 0C	hopp till D 06
n+2	bbb 3B	obs! operationsdelen
n+3	sss 00	
n+4	start 0C	adressdelen anger, vart man hoppar, sedan sekvensen är
n+5	kkk 00	genomlöpt

Man använder modifierat Wheelerhopp. Hoppet till sekvensen sker alltid från en udda order (order n+1 i exemplet). Sekvensen ligger på fix plats i minnet (014-027) och har startorder = 014. Omedelbart efter hoppordern till sekvensen skall stå två helord med information enligt exemplet. Återhoppet från sekvensen sker till en adress föreskriven i haen+4 (obs! orderdelen 0C; skriv 2C om stopp önskas före återhoppet). Det spelar ingen roll för användningen av D 06, om det inlästa blocket täcker över order n till n+5. Däremot får blocket ej täcka över D 06 själv. Beträffande koden till D 06, se kap. 9.

Observera, att om sträckan bbb-sss i snabbminnet inte består av ett jämnt antal kanallängder, kommer ett antal ord efter sss i minnet att bli överlästa med den sista kanalens innehåll.

Om ett block skall överföras från snabbminnet till trumman, skriver man i huvudprogrammet:

kr	order	
n	n 50	order n alltid jämn
n+1	014 0C	
n+2	bbb 3F	obs. operationsdelen
n+3	sss 00	
n+4	start 0C	
n+5	kkk 00	

Endast operationsdelen i order n+2 är olika mot förut. Vilken del som helst av minnet kan överföras till trumman på detta sätt, eventuellt hela minnet.

Vid stora program händer det ofta, att en programdel skall avlösas av olika programdelar i olika sammanhang under beräkningarna. Då är det gynnsamt att frikoppla administrationen av programmet i stort från programdelarna, som man då lämpligen organiserar som slutna sekvenser. Efter varje programdel bör ett uthopp göras till en skelettsekvens, som bl.a. innehåller det modifierade Wheelerhoppet till den just beskrivna trumläsningssekvensen, som tar in nästa programdel. Systemet är flexibelt och underlättar programändringar. De olika programdelarna blir oberoende av varandra. Skelettsekvensen innehåller i koncis form all information om flödesschemat i stort och dispositionen av trumman, vilket underlättar överblicken över programmet. En nackdel är, att skelettsekvensen ibland kräver ett stort utrymme.

8. SKALFAKTORER. SPECIELL ARITMETIK.

I kap. 2 infördes termen besktal som benämning för ett tal x i intervallet $-1 \leq x < 1$. Det nämndes, att de aritmetiska operationerna i Besk beskrevs enklast, om den i Besk lagrade informationen tolkades som sådana tal med binärkommat placerat mellan pos. 0 och pos. 1, och om negativa tal representerades genom komplement.

Det händer emellertid ibland, att man i sina problem har storheter, som är större än 1, och att man vill arbeta helt med sina vanliga beteckningar utanför maskinen, t. ex. vid förberedelse av decimala ingångsdata och vid läsning av räkningarnas resultat. Man får då skilja mellan två situationer, som närmare beskrivs i det följande.

8.1. Skalfaktorer. Om man på grund av problemets natur kan ange övre begränsningar för alla storheter, som förekommer under räkningarnas gång, bör programmeraren skala problemet, d. v. s. införa hjälpstorheter, som är besktal med en enkel relation till de ursprungliga storheterna. Programmeraren utför lämpligen en variabeltransformation i de grundläggande ekvationerna, men arbetet bör läggas upp så, att man får glömma bort existensen av dessa hjälpstorheter vid den reguljära körningen och vid behandlingen av resultaten. Det enklaste är att skala genom division med s. k. konstanta skalfaktorer, t. ex. en potens av 2 eller en potens av 10. Av dessa metoder rekommenderas i synnerhet skalning med 2-potenser, eftersom förlusten av noggrannhet då i regel (men inte alltid) blir mindre och kodningen för det mesta blir enklare.

Om skalning med 2-potenser används, rekommenderas standardkoderna 901 eller 902 för inläsning av decimala tal och standardkoderna 913 eller 914 för decimal tryckning.

Om skalning med 10-potenser används, rekommenderas standardkoderna 901 för decimal inläsning och 912 för decimal tryckning. Inläsnings- och tryckningssekvenserna är mer komplicerade i 2-potensalternativet, men det spelar vanligtvis ingen roll för kodaren.

Exempel 8.1. Antag att beräkningen (8.1)

$$z = xy$$

ingår som detalj i ett program, där man vet att

$$|x| \leq 5, \quad |y| \leq 12$$

i alla tänkbara sammanhang. Då är säkert $|z| \leq 2^6$. I Besk kan dessa storheter

representeras med besktalen

$$\bar{x} = 2^{-3}x, \quad \bar{y} = 2^{-4}y, \quad \bar{z} = 2^{-6}z,$$

och (8.1) kan då skrivas

$$\bar{z} = \bar{x} \cdot \bar{y} \cdot 2 \tag{8.2}$$

där operationerna utförs i den ordning, som faktorerna står: först multipliceras $\bar{x}$ avrundat med $\bar{y}$, därefter skiftas ar ett steg åt vänster. Genom vänsterskiftet blir sista biten i $\bar{z}$ osäker. Om skalningen i stället görs med formlerna

$$\bar{x} = 10^{-1}x, \quad \bar{y} = 10^{-2}y, \quad \bar{z} = 10^{-2}z$$

får man

$$z = 10\ \bar{x}\ \bar{y} = \bar{x}\ \bar{y} \cdot (10/16) \cdot 2^4 \tag{8.3}$$

Den sista omformningen antyder, hur räkningen kan göras i Besk. De fyra sista bitarna i $\bar{z}$ är osäkra, eftersom räkningen avslutas med fyra stegs vänsterskift. Om man i stället hade satt $\bar{z} = 10^{-3}z$, så får man $\bar{z} = \bar{x}\ \bar{y}$. Då blir alla bitarna i $\bar{z}$ korrekta, men i gengäld blir det så många nollor i början av talet, att den relativa noggrannheten i $\bar{z}$ i verkligheten inte blir mycket bättre än med (8.3). Om man skalar med tiopotenser, kan tabellen i appendix II vara till nytta.

Förutsättningarna för en skalning måste alltid prövas noga. Man måste även se efter, att de inte kullkastas av approximationer och avrundningsfel. Det kan ibland bli nödvändigt att lägga in spillcheckar eller att pröva om en dividend är större än divisorn. Om man utför skalningen med stor säkerhetsmarginal, behöver man inte belasta programmet med många sådana checkar. Man måste emellertid då tänka t. ex. på risken, att produkten av två generöst skalade tal kanske inte har tillräckligt många siffror kvar i ackumulatorn.

Eftersom Besk arbetar med 40 binära siffror finns det ju en del att ta på, men om svårigheter med noggrannheten ändå skulle uppstå, kan man koda multiplikation enligt följande exempel:

låt $\bar{x} = 2^{-10}x, \quad \bar{y} = 2^{-5}y$

och antag produkten $z = xy$ blir mindre än 2^{10} och lagras i form av talet

$$\bar{z} = z \cdot 2^{-10}$$

Man har

$$\bar{z} = \bar{x} \cdot \bar{y} \cdot 2^5$$

Koden blir

$\bar{x}$ 68 x i MR

$\bar{Y}$ 62 $\bar{x}$ $\bar{y}$ i AR och MR (oavrundat)

005 05 2^5 $\bar{x}$ $\bar{y}$

$\bar{Z}'$	31	
$(2^5.2^{-39})$	42	De 5 sista bitarna i $\bar{z}$ till AR.
$\bar{Z}'$	26	Slutresultat $\bar{z}$ till cell $\bar{Z}'$.

8.2 Flytande räkning. Om det är svårt att på förhand uppskatta maximala värdet av någon storhet eller om några storheter varierar inom mycket vida gränser är det lämpligt att använda variabla skalfaktorer. Beräkningarna utföras så att talen liksom förut är försedda med skalfaktorer individuellt (en skalfaktor för varje storhet) eller gruppvis (t. ex. en skalfaktor för alla element i en vektor eller matrisrad el. dyl.)

Till skillnad mot det förra fallet hållas dessa skalfaktorer inte oförändrade under räkningarna utan avpassas efter talens storlek efter varje operation, så att så många signifikanta siffror som möjligt behållas.

Om alla storheter är försedda med individuella variabla skalfaktorer talar man om räkning med flytande binärkomma eller kortare flytande räkning. I några maskiner är de grundläggande aritmetiska operationerna (addition, subtraktion, multiplikation, division) för räkning med flytande binärkomma inbyggda i den aritmetiska enheten. I Besk måste flytande räkning utföras med hjälp av speciella program. I följande avsnitt redogöres för ett sådant program. Det bör emellertid framhållas att flytande räkning är mycket långsammare än räkning med konstanta skalfaktorer. Vid problem av stor volym måste därför flytande räkning betraktas som en nödfallsutväg.

8.2.1. Konvention för representation av tal i flytande binär form. Sätt

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

($-1 \leq x' < 1$, x'' heltal). Det närmast till hands liggande sättet att representera sådana tal är att lagra taldelen x' och exponenten x'' i konsekutiva helceller. Detta är bekvämt vid själva beräkningarna men är ibland ofördelaktigt ur lagringssynpunkt. Det är därför brukligt att också representera tal i packad form, varvid information om taldel och exponent lagras i samma helcell. Positionerna 0-31 tas i anspråk för taldelen. Exponenten x'' kan vara antingen positiv eller negativ. Vid packning och uppackning av tal är det obekvämt med negativa exponenter, varför vi i pos. 32-39 i stället för exponenten x'' lagrar den s. k. karakteristiken, varmed menas $x''+80$ (sedecimalt) med enheten i pos. 39. Om vi antar att $|x''| < 80$ (sedecimalt), så är karakteristiken positiv.

Exempel: Talet 3 = $(3/4)\cdot 2^2$ kan i packad form skrivas 60000 00082. En annan packad framställning av samma tal är 30000 00083. Talet 3/16 = $(3/4)\cdot 2^{-2}$ kan skrivas 60000 0007E.

Om taldelen uppfyller olikheten $-1 \leq x' < -\frac{1}{2}$ eller $\frac{1}{2} \leq x' < 1$, säges talet vara normaliserat.

Det största positiva tal som kan lagras i packad form har

$x' = 1 - 2^{-31} =$ 7FFFFFFF

och $x'' + 80 =$ FF

alltså $x = (1 - 2^{-31})\cdot 2^{127}$ (exponenter decimalt)

Det största negativa tal som kan lagras i packad form är

$$x = -1 \cdot 2^{127}$$

eller i packad form

x = 80000000FF

De minsta packade tal som är skilda från noll är

$$x = \pm\, 2^{-31} \cdot 2^{-128}$$

Det minsta normaliserade talet skilt från noll är

$$x = \frac{1}{2} \cdot 2^{-128}$$

Talet noll representeras normalt i packad form med helordet

0000000000

Talet noll kan emellertid också representeras med ett godtyckligt helord där de första 8 sedecimala siffrorna är = 0.

Ett opackat tal har taldelen x' i en helcell och karakteristiken x" + 80 i den därpå följande helcellen. Enheten för karakteristiken är i pos. 39. För ett opackat tal får |x"| gärna överskrida 80.

8.2.2. Program R 10 (980) för flytande räkning med packade tal.

Med hjälp av ett speciellt standardprogram, R 10, kan aritmetiska operationer utföras med packade tal. R 10 är ett vanligt 800 - program och inläses till minnet och anpassas som ett sådant. R 10:s totallängd är = 9B5.

Se f.ö. speciell beskrivning av R 10.

R 10 innehåller en "pseudoackumulator" PA bestående av 2 helceller (arbetsceller till R 10), som lagrar ett tal i opackad form. Vid användning av R 10 skriver man först ett Wheelerhopp till R 10 i huvudprogrammet. Efter hoppordern till R 10 skriver man (också i huvudprogrammet) ett antal halvord, pseudoorder, ett för varje operation med flytande räkning, som skall utföras. Efter den sista pseudoordern skriver man halvordet 000EC för att markera att inga fler pseudoorder följer och att återhopp från R 10 skall ske till den följande halvcellen (ordern) i huvudprogrammet när pseudoorderna är utförda av R 10.

Pseudoorderna är snarlika vanlig beskkod, enadressoperationer används och PA i R 10 spelar en roll liknande den som AR har vid vanlig beskkodning.

Exempel: Man vill beräkna uttrycken $x = \frac{st}{s - t}$ och $y = \frac{s + t}{st}$ med flytande räkning och därefter fortsätta med vanlig kod.
Antag

$$w(s) = 200$$
$$w(t) = 202$$

s och t är lagrade i packad form och x och y skall lagras i hec 300 och 302 i packad form. Låt pa betyda innehållet i pseudoackumulatorn PA.

Man skriver i huvudprogrammet

kr	as	op	
n (jämn)	n	50	Modifierat
n+1	(800)	0C	Wheelerhopp till R 10
n+2	200	70	s till PA
n+3	202	2B	s-t till PA
n+4	200	72	s/pa till PA
n+5	202	63	t.pa till PA
n+6	300	31	pa till hec 300
n+7	200	70	s till PA
n+8	202	30	s+t till PA
n+9	200	32	pa/s till PA
n+A	202	32	pa/t till PA
n+B	302	31	pa till hec 302
n+C	000	EC	Uthopp ur R 10 till nästa order, n+D
n+D	Vanliga Beskorder i det följande.		

Obs! Innehållet i PA står kvar och kan användas vid nästa inhopp i R 10. PA består av hec 9AE och 9B0 i R 10.

Efter hoppet till R 10 avläser R 10 de olika pseudoorderna i tur och ordning och utför de olika operationerna. Uthoppet sker så snart pseudoordern 000EC påträffats.

Orderlista till R 10:

(irr betyder irrelevant adressdel)

operation	verkan	tid i additionstider
W(x) 70	x till PA	27
W(x) 6B	-x till PA	27
W(x) 30	x+pa till PA	47
W(x) 2B	-x+pa till PA	54
W(x) 26	x+pa till PA och W(x)	67 resp. 89
W(x) 63	x·pa till PA	43
W(x) 22	x·pa till PA och W(x)	62 resp. 84
W(x) 32	pa/x till PA	66
W(x) 72	x/pa till PA	73
irr 8B	-pa till PA	17,5
irr 8D	|pa| till PA	18,5
irr 8E	-|pa| till PA	16,5
W(x) 31	pa till W(x)	31 resp. 53
W(x) 71	0 till PA och W(x)	25
irr 05	blindorder	12,5
w(x) 75	innehållet i hec W(x) och W(x)+2 går som <u>opackat</u> tal till PA utan normalisering.	17,5
W(x) 35	pa som <u>opackat</u> tal till W(x) och W(x)+2, <u>normaliserat</u>.	40
W(x) 34	pa till W(x) som <u>Besktal</u> om |pa| < 1 eller pa = -1, annars kontrollutskrift	47
n 6E	<u>hopp på minus;</u> om pa ≥ 0 fortsätt med följande pseudoorder, om pa < 0 hämta nästa pseudo order från hac n. pa ostört i bägge fallen.	15,5 resp,26
n 2E	<u>hopp på plus</u> (eller 0); f. ö. analogt med föregående order.	15,5 resp.25

n	OC	ovillkorligt hopp till pseudoorder n.	22
irr	EC	uthopp ur R 10 till nästa order som åtlydes som vanlig Beskorder.	13
W(x)	8A	kontrollutskrift, se nedan.	

Metod: Beträffande operationernas interna utförande gäller att vid orderna 31 och 35 normaliseras PA före operationen. Vid order 31 avrundas taldelen korrekt till 32 bitar före hoppackningen. Vid division (32 och 72) normaliseras nämnaren före operationen.

F.ö. utföras inga normaliseringar i sträng mening men efter varje operation som innebär addition, subtraktion, multiplikation eller division testas på spill och om inte spill inträffat skiftas taldelen i resultatet ett steg åt vänster. Om vid detta skift spill inte inträffar lagras den skiftade taldelen och karakteristiken korrigeras.

Om sekvens 902 användes för inläsning av packade tal från remsa fås alltid de inlästa talen i normaliserad form.

Vid hoppackning ger alltid R 10 talet 0 i form av helordet 0000000000.

Beträffande operationstiderna gäller de kortare tiderna vid order 31, 26 och 22 om pa är normaliserat då hoppackningen skall utföras (alltså vid order 26 och 22 om skiftet på 1 steg har varit tillräckligt för normalisering av pa och vid order 31 om pa var normaliserat från början).

Vid order 2E och 6E gäller de längre tiderna för det fall att hoppet utföres.

Kontrollutskrifter:
R 10 ger på vanlig skrivmaskin eller stans utskrift vid följande tillfällen:

1. Vid order 31, 26 och 22 om pa ligger utanför intervallet $(-2^{127}, (1-2^{-31}) \cdot 2^{127})$ då hoppackning skall ske.
2. Vid order 34 om $|pa| > 1$ eller $pa = 1$.
3. Vid divisionen $\frac{0}{0}$. Däremot: Division med 0 om täljaren $\neq 0$ ger i PA taldel $\frac{1}{2}$ och karakteristik 100000 (sedec.), alltså $pa = 2^{1048447}$. Det senare fallet betraktas av R 10 som normalt och ger ej utskrift.

4. Ovillkorligt vid order W(x) 8A.

R 10 skriver vid dessa tillfällen följande:

aaa	bbbpp	xxxxxxxxxx	yyyyyyyyyy	zzzzzzzzzz
adress till pseudoorder	pseudoorder	innehållet i hec bbb sedecimalt	taldel av PA sedecimalt	karakteristik av PA sedec.

Därefter fortsätter maskinen räkningarna u t a n s t o p p.

Denna utskrift anger vid order 31, 34, 32, 72 och 8A tillståndet innan ordern utförts. Vid order 26 och 22 har additionen resp. multiplikationen men inte hoppackningen utförts då utskriften inträffar.

Order 31, 34, 32, 72 och 8A behandlas vid tillfällena 1-4 som blindorder, frånsett utskriften, då maskinen fortsätter. Vid order 26 och 22 uteblir lagringen av pa då maskinen fortsätter men den aritmetiska operationen blir utförd som ovan nämnts.

Anmärkning: Om karakteristiken till pa överstiger 00FFF FFFFF i absolutbelopp fungerar inte de aritmetiska operationerna i R 10. Något test om detta fall inträffar finns inte inlagt. För att uppnå denna karakteristik vid räkningar med packade tal fordras t.ex. c:a $16 \cdot 10^6$ på varandra följande multiplikationer av pa med tal med karakteristiken FF (eller 00) utan att pa packas ihop.

8.3. Räkning med dubbel noggrannhet. Trots att Besks noggrannhet är stor, har det förekommit problem, där det varit önskvärt att utföra delar av beräkningarna med större noggrannhet. Här skall visas några kodmallar för räkning med dubbel noggrannhet, som anvisats av G. Jonsson (MNA). En lämplig konvention för representation är att främre delen - d.v.s. teckensiffra och de 39 därpå följande bitarna - lagras i en helcell, medan bakre delen lagras i pos. 1-39 i nästa helcell. I den senare cellen är teckensiffran alltid noll.

Exempel 8.6: Beräkna med dubbel noggrannhet u = z + xy, där z är lagrad med dubbel noggrannhet i cellerna Z_1, Z_2 medan x och y är lagrade med enkel noggrannhet i hecX resp. hecY. u skall lagras på samma plats som z.

Lösning:

```
X    68                                  Z1   26
Z2   70                                  000  01
Y    22   xy+z2.2^-39 till AR, MR        Z2   31
```

Anm.: Denna mall kan vara lämplig t.ex. när en produktsumma med många termer önskas med vanlig Besknoggrannhet. Observera användningen av multiplikation utan nollställning av AR.

Exempel 8.7: y och z är givna med dubbel noggrannhet. Beräkna summan y + z med dubbel noggrannhet och lagra den på platsen för z. Spill antas ej kunna uppträda.

Lösning: Beteckningar analoga med föregående.

```
005  48                                  Z1   26
Z2   70                                  000  01
Y2   22                                  Z2   31
Y1   30
```

Analysera koden. Denna lösning har valts på grund av dess korthet. Det är möjligt att göra lösningar, som är snabbare men längre.

Exempel 8.8: Samma som 8.6 men x och y antas också vara givna med dubbel noggrannhet.

```
X1   68                  X2   68          X1   68
Y2   63                  Y1   63          Z2   70
006  08                  006  08          Y1   22
Z2   70                  Z2   70          Z1   26
005  02                  005  02          000  01
Z1   26 (addera carry)   Z1   26          Z2   31
000  01                  000  01
Z2   31                  Z2   31
```

Anm.: $(x_1 + x_2)(y_1 + y_2) + (z_1 + z_2) = x_1y_2 + z_2 + x_2y_1 + x_1y_1 + z_1$. x_2y_2 försummas. Det blir eventuellt fel i sista siffrorna.

Exempel 8.9: x är ett tal givet med dubbel noggrannhet, med absolutbeloppet större än $\frac{1}{2}$. Visa att man kan få $1/(2x)$ med nästan dubbel noggrannhet genom att först beräkna $q = 1/(2x_1)$ med enkel noggrannhet och därefter beräkna $q + q(1-2qx)$ med dubbel noggrannhet. Gör en kod för detta.

Exempel 8.10: Kvot med dubbellängd. a och b besktal, $|a| \leq |b|$. Gör en kod, som ger a/b under formen $q_1 + 2^{-39} \cdot q_2$ där q_1 och q_2 är besktal lagrade i var sin cell och $q_2 \geq 0$. Utnyttja divisionsformeln $a = b \cdot q + 2^{-39} \cdot r$ där q är beskkvot och r beskrest; r kan även vara negativ, $|r| \leq |b|$.

Lösning:

A	70	000	13	
B	32	Q_2	31	
slask	31	UTH	0E	(till uthoppscell)
000	13	004	10	
Q_1	31	Q_2	31	
slask	70	005	50	
B	32	Q_1	26	
		uthopp		

Anm. Även exempel 6.18-6.20 berör dubbel precision.

Exempel 8.11: Division mellan positiva heltal. Heltalskvot och heltalsrest, d.v.s. A/B = Q + R/B. (Stora bokstäver betecknar positiva heltal.) A kan även vara större än B. A och B tänkas lagrade som heltal i en given position; motsvarande besktal $a = A \cdot 2^{-p}$, $b = B \cdot 2^{-p}$; $a < \frac{1}{2}$. Kvoten kommer som heltal i pos. 39; $q = Q \cdot 2^{-39}$. Resten i samma position som gällde för A och B; $r = R \cdot 2^{-p}$, $0 \leq R < B$.

Lösning: Ytterligare beteckningar: $\overline{b} = b \cdot 2^k$ = normaliserat b; k = antal normaliseringssteg. q' betecknar beskkvoten i divisionen $a/\overline{b}$; $q' \approx q \cdot 2^k$ men innehåller dessutom en viss "svans" till höger om pos. 39-k; q' utan denna svans betecknas q". För enkelhetens skull betecknas i ordernas adressdelar lagringscellerna för A, B, R, Q med samma bokstäver. 11C är "slaskcell". R' och Q' betecknar provisoriska värden, som eventuellt skall korrigeras.

100	B 70	
101	00015	
102	11C31	$\overline{b}$ i slask
103	00442	
104	11B10	39-k i adresspos. i orderna 10B och 10D
105	10B07	
106	10D07	
107	A 70	
108	11C32	
109	00013	q' i AR
10A	10B0C	blind
10B	FFF04	27-k 04 d.v.s. högerskift 39-k steg

10C	Q 31	motsvarande Q' i cell i pos. 39
10D	FFF05	d.v.s. 27-k 05, vänsterskift 39-k steg, q" i AR
10E	00608	q"·$\overline{b}$ i slask
10F	11C62	
110	11C31	
111	11C6B	R'·2^{-p} i cell
112	A 30	
113	R 31	
114	B 2B	R'-B
115	1194E	till uthopp, om R' < B; Q' och R' accepteras
116	00550	annars, om R'=B,
117	Q 26	öka Q' med 1
118	R 71	och sätt R=0.
119	UTH0C	uthopp till given cell "UTH".
11A	00000	används ej
11B	02700	39 i pos. 31.
11C	00000	slask; först $\overline{b}$, sedan q"·$\overline{b}$.
11D	00000	

Exempel 8.12: Använd föregående kod för Euklides algoritm.

9. FÄRDIGSTÄLLANDE OCH INLÄSNING AV REMSOR.

9.1. Inläsning av program. I det följande kommer att beskrivas tre inläsningssekvenser A40, A41 och A42 avsedda för inläsning av sedecimalt stansade helord till kärnminnet i Besk (A42 läser även till trumman). I allmänhet är det därvid frågan om inläsning av program.

9.1.1. Inläsningssekvens A40 finns tillgänglig i en självläsande form (jfr avsnitt 6.3) stansad på en speciell grå remsa av kraftigt papper. A40 nollställer kärnminnet innan den läser in det egentliga programmet.

För att framställa en självinläsande programremsa med hjälp av A40 gör man följande sätt: i en stansutrustning kopieras den gråa A40-remsan över på en remsa av tunnare papper. På den tunnare remsan stansar man efter A40 sitt eget program uppdelat på sekvenser av helord föregångna av etiketter (rubriker).

En sekvens skall börja i en vänsterhalvcell bbb och sluta i en högerhalvcell sss. Etiketten till sekvensen blir helordet

bbb39 sss00

Sist på remsan stansas etiketten

hhh0C 00000

där hhh är adress till programmets startorder.

Start av remsa: Remsan inlägges i maskinens remsläsare, därefter trycker man på tangenten KASOP på kontrollbordet (nollställning av KR och ASOP), slutgen på knappen "start från remsa" (operationsdelen 39 sättes i OP och maskinen får startimpuls), varefter inläsningen börjar. Remsan inläses i sin helhet utan att stopp inträffar och hopp sker automatiskt till startordern hhh för programmet. Vid början av inläsningen skriver maskinen ett A.

Exempel: Programmet består av följande avsnitt: ett huvudprogram i hac018-054, en undersekvens i hac060-07F och ett antal sedecimala konstanter i hec100-127. Remsan stansas:

remsa	kommentar
A 40	kopieras från grå remsa
01839 05500	rubrik till huvudprogram

```
huvudprogram    huvudprogrammet har vid stansningen kom-
                pletterats med ett halvord nollor i slutet
                för att slutadressen skall bli udda
06039
07F00

undersekvens

10039
12700

sedecimala tal

0180C           hopp till start på 018
00000
```

Att observera:

1) 800-sekvenser kan ej anpassas av A40
2) Program kan inläsas till plats 010-7FF i minnet.
3) De permanenta konstanterna läsas in av sekvensen.
4) Antag hhh är en udda adress. Om man vill, kan man då få stopp, när remsan är inläst, om man som sista etikett stansar hhh2C 00000. Då man trycker på knappen "start" sedan stoppet inträffat utföres hoppet till order hhh.
5) Huvuddelen av A40 motsvarar programmet i avsnitt 6.2 och läses in till hac 007-00E efter nollställning av kärnminnet. Vid inläsningen av program beröras endast de delar av minnet som anges av rubrikerna på remsan. Övriga delar av minnet förblir nollställda.
6) Efter inläsningen står huvuddelen av A40 kvar i hac007-00E och kan användas i det egna programmet för inläsning av sedecimala remsor genom hopp till A40:s startorder 007. Inläsningssekvensen kan t.ex. användas för inläsning av rättelser i programmet. Bekvämast sker detta om programremsan avslutas med etiketten

```
0072C  00000
```

Inläsning av enstaka halvord kan ske genom att stansa en etikett

```
bbb19  00000
```

och därefter ett helord vari halvordet ingår med samma paritet som hac bbb. Ett enstaka helord kan läsas in på liknande sätt med etiketten bbb39 00000.

9.1.2. Inläsningssekvens A41 och A42 används för inläsning av program, där sekvenser i 800-form ingår. A41 är kortare och avsedd för inläsning till snabbminnet enbart, A42 är längre men utför även läsning till trumman. Båda finns tillgängliga i form av gråa remsor, som kopieras över först på programremsan liksom A40. Resultatet blir en självinläsande programremsa.

Start av programremsan sker på samma sätt som vid A40: KASOP = 0, tryck på "start från remsa". Sekvenserna nollställer kärnminnet före inläsning av program. Inläsningen av inläsningsprogrammet själv är summacheckad.

Inga stopp inträffar under inläsningen av remsa frånsett det fall att inläsningssekvensen själv blivit fel inläst.

Etikettsystemet för läsning till snabbminnet är gemensamt för A41 och A42 (utom beträffande decimal inläsning som ej utföres av A41).

Vid inläsning av en vanlig sedecimal sekvens stansas etiketten

bbb39
sss00

före sekvensen. Orderdelen i högerhalvordet måste vara 00. Adressen bbb måste vara jämn, slutadressen sss udda. Man kan läsa in ett halvord (vänster- eller högerhalvord) till hac bbb genom att stansa etiketten

bbb19
00000

och därefter ett helord där halvordet ingår med samma paritet som adressen bbb. (Ett enstaka helord kan läsas in på liknande sätt med etiketten bbb39 00000.)

Vid inläsning med anpassning av en 800-sekvens stansas först rubriken

bbb39
80000

där bbb är begynnelseadress till sekvensen i minnet (alltid jämn). Därefter stansas ett helord, kallat längdord:

$$\text{ttt rrr aaa} \begin{cases} 0 \\ 1 \end{cases}$$

där ttt är totallängd, rrr remslängd och aaa anpassningslängd för sekvensen (avsnitt 7.4) och den sista siffran i helordet är antingen 0 eller 1. Efter längdordet stansas sekvensen. Om sista siffran är 0 är det hela färdigt och man kan fortsätta med nästa etikett. Om siffran är 1 stansas efter sekvensen en checksumma, som skall vara så avpassad, att om alla orden i sekvensen, inklusive längdordet och checksumman, summeras som helord i Besk, resultatet blir 0 (hänsyn ej tagen till spill). Maskinen utför summeringen under inläsningen och skriver ut summan med omvänt tecken sedecimalt om den ej är 0. Detta utgör en kontroll av att maskinen läst in sekvensen rätt. Man räknar i allmänhet ej ut checksumman för hand utan låter maskinen beräkna den vid den första inläsningen av programremsan (se det följande).

Standardprogram skrivna i 800-form finns stansade på gråa remsor med längdord (sista siffran = 1) och checksumma medtagna. Vid användning av sådana standardprogram behöver man alltså endast stansa rubriken bbb39 80000 och därefter kopiera över den gråa remsan.

För att få summakontroll även på vanliga sekvenser och på etiketter kan man med lämpliga mellanrum på remsan, mellan två sekvenser, stansa etiketter av typen

FFFFF FFFFS

åtföljda av checksumma. Siffran s sist användes för numrering av summakontrollerna och skrivs ut under inläsningen. Checksumman hänför sig till alla ord på remsan efter den närmast föregående checksumman. Undantag: summacheckade 800-sekvenser inkluderas ej i summeringen, däremot inkluderas 800-program utan egen summacheck. Märk att vid A41 blir en del av remsan, som ligger mellan en vanlig checksumma och ett efterföljande summacheckat 800-program, ej checkad. Om summakontrollen ej stämmer skrivs summan med omvänt tecken ut sedecimalt. Man låter liksom vid 800-sekvenser maskinen beräkna checksummorna vid första körningen.

För hopp till startorder hhh i programmet används rubriken

hhh0C
00000

Om hhh är udda, och man önskar stopp efter inläsningen stansas

hhh2C
00000

Om hhh är en jämn order fås stopp, om man stansar t.ex.

0070C
00000

0092C
hhh0C

Startorder till A41 och A42 är 7FF. Om man vill bekvämt läsa in rättelser med inläsningssekvensen bör man avsluta remsan med etiketten

7FF2C
00000

Rättelseremsan stansas som en programremsa (men utan inläsningssekvens).

Trumetiketter. Med A42 kan man under inläsningen av programremsan föra över block från snabbminnet till trumman. Antag att man vill föra över ett till snabbminnet inläst block till trumman med början på kanal kkk. Första adress till blocket i snabbminnet är bbb (jämn) och sista adress

sss (udda). Man stansar då etiketten

bbb sss kkk F.

Observera sista siffran! Det inlästa programmet i snabbminnet påverkas ej. Man kan även läsa block från trumman. Etiketten är i så fall

bbb sss kkk B.

Rättelser till trumman. Rättelseremsan inläses med hopp till 7FF och börjar med en etikett för nedläsning till det snabba minnet av det block, som skall rättas. Rättelserna inläsas därefter som vanliga sekvenser med etiketter, och slutligen skall det vara en etikett för att föra det rättade blocket över till trumman.

Exempel: Till hac4CB och hac40E i ett block, som börjar på kanal 00A och upptar plats 400-50B i kärnminnet, skall följande rättelser läsas:

4CB: 00871

40E: 4CB0C

Rättelseremsan stansas:

40050B00AB

4CB19
00000

00000
00871

40E19
00000

4CB0C
00000

40050B00AF

hhh0C
00000

Här är hhh adress till programmets startorder. Om en annan rättelseremsa skall in efteråt stansas i stället som sista etikett

7FF2C
00000

Decimalt stansade konstanter mindre än 1 i absolutbelopp kan inläsas i ett program med hjälp av A42. Antag en följd av tal skall läsas in till ett antal på varandra följande helceller. Det första talet skall stå i helcell bbb. Före talen stansas en etikett

bbb 39 D00 00

Antag att talen -0,473, 0,00378 och 0,07 skall stå i cellerna bbb, bbb+2 och bbb+4. Efter etiketten stansas då

473B 00378D 07D A

Endast decimaler stansas. Även nollor omedelbart efter decimalkommat stansas, däremot behöver inte nollor i slutet av talet stansas. Antalet decimaler kan variera från tal till tal men bör ej överstiga 10 i antal. Tecken stansas sist, + = D, - = B. Efter sista talet stansas ett A. Tecknet F överhoppas av inläsningssekvensen. Det kan alltså användas för utsuddning av ett felaktigt stansat tecken (siffra) genom överstansning (se fig. 5, avsnitt 9.2). Teckenkombinationen

F473B00F378D07FDFA

efter etiketten inläses alltså som talen -0,473, 0,00378 och 0,07. Det kan hända, att det endast är ett tal i sviten efter rubriken. Sekvenser med decimala tal summeras för summakontroll liksom övriga sekvenser vid inläsning med A42. Därvid summeras de inlästa talen i översatt form som helord i Besk. Även rubriken ingår i summan.

Anm. Inläsning av decimalt stansade tal, som bildar utgångsmaterial för beräkningar och varierar från körning till körning, sker med speciella inläsningssekvenser för decimal inläsning (se kap. 11). +)

9.1.3. A41 speciellt. A41 använder hac7C0-7FF och 000-00F i minnet. Program kan läsas till hac008-7BF. Om sekvensen ligger i minnet med hac7C0-7FF och 000-007 (de permanenta konstanterna) intakta, kan omstart ske med hopp till 7FF.

Inläsningen av A41 själv är summacheckad. Maskinen skriver i början bokstäverna AA. Om summachecken inte stämmer, inträffar stopp. Om en vanlig summacheck inte stämmer, skrivs summacheckens indikeringssiffra. Om en summacheckad 800-sekvens är fel inläst, skrivs sista siffran i etiketten. Vid A41 är denna siffra godtycklig men däremot ej vid A42. Om allt är rätt vid inläsningen, blir endast AA i början utskrivet.

Om omställaren "kontrollutskrift" på kontrollbordet sättes i läget "E_2-stopp" före inläsningen av remsan, fås alla etiketter på remsan utskrivna med en särskild skrivmaskin under inläsningen. Efter varje 800-sekvensetikett skrivs dessutom ett helord, vars 3 första siffror utgör sekvensens verkliga slutadress i minnet räknad efter remslängd. Om en summacheck inte stämmer, skrivs den av maskinen erhållna summan (ej 0) med omvänt tecken efter respektive summa-

---

+) Observera, att principen för summakontrollen då är en helt annan.

checketikett eller slutadress. Antag, att man först stansat helord med nollor i stället för de riktiga checksummorna på remsan. Vid inläsningen av remsan får man då de riktiga checksummorna beräknade och utskrivna av maskinen. De kan sedan kopieras in på remsan.+)

9.1.4. A42 speciellt. A42 använder hac014-027 (D 06 för överföring av block mellan trumman och kärnminnet) och hac7C0-7FF jämte de permanenta konstanterna i det snabba minnet. På trumman upptar A42 kanalerna IEC-IF4. Vid starten läses följande innehåll till hac000-03F och till kanal IEC:

000 00200	014 00010
001 00200	015 01807
002 00100	016 00010
003 00100	017 01C07
004 80000	018 FFF70
005 00001	019 02707
006 00000	01A 01404
007 00839	01B 01D11
008 00000	01C FFF68
009 00000	01D FFFFF
00A 00000	01E 00001
00B 00000	01F 02611
00C 00000	020 00108
00D 00000	021 02550
00E 01348	022 01D06
00F 7C03F	023 0270B
010 00149	024 01D4E
011 7C03B	025 04000
012 7C00C	026 FFFFF
013 IEA00	027 FFF00
	hac028-03F: noll

I hac014-027 ligger ovannämnda sekvens D 06. Order 00E-013 användes för nedläsning från trumman av sekvenser för sedecimal och decimal utskrift av minnet och trumman vid inkörning (se kap. 10, normalläge).

Program kan läsas till plats 028-7BF i det snabba minnet och kanal 000-IEA på trumman.

A42 finns lagrad på trumman, då maskinen är i normalläge (se kap. 10). Om normalläge förutsättes på trumman, kan program inläsas med hjälp av en kort hjälpremsa A42A, som kopieras först på programremsan i stället för A42.

+) Man kan också använda C30, som skriver ut en självinläsande remsa (kap. 10) med summacheckar.

Om man vill använda A42 i det egna programmet för att läsa in remsor automatiskt, kan detta ske genom hopp till 7FF eller, om utrymmet 7C0-7FF blivit förstört, genom att sätta kanaladressen 1F4 i MR, läsa ned kanalen till området 7C8-007 och hoppa till order 7FF (kanalen innehåller de permanenta konstanterna i slutet). Om trumetiketter finns på remsan, måste D 06 vara intakt.

Utskrifter under inläsningen: Inläsningen av A42 själv är summacheckad. Om summachecken inte stämmer, inträffar stopp. Om summachecken stämmer, skrivs de tre tecknen A42. Om en vanlig summacheck inte stämmer, skrivs med den vanliga skrivmaskinen summacheckens indikeringssiffra s ut åtföljd av den erhållna summan (ej 0), sedecimalt, med omvänt tecken. Om en summacheckad 800-sekvens är fel inläst, skrivs både etiketten och summan med omvänt tecken ut. Om omställaren står på "E_2-stopp", fås liksom vid A41 etiketterna utskrivna. Däremot skrivs inga checksummor.

Särskilda beskrivningar finns för var och en av sekvenserna A40, A41 och A42. Exempel på remsa:

```
 A 41
06239
80000
standardprogram
01039
06100
vanlig sekvens
12E39
80000
83F83    längdord
F82D0
800-sekvens
FFFFF    etikett för summakontroll
FFFF1    (jfr 9.1.2)
00000    checksumma stansad
00000     preliminärt = 0
7FF2C    stopp; remsan tages bort
00000    om rättelseremsa skall in
0100C    hopp till start
00000
```

9.2. Remsor. Fig. 5 visar utseendet på den remsa, som används av Besk för in- och utmatning av information. Sådan remsa kan framställas

1) genom stansning i en stansutrustning försedd med tangentbord för stansning av de sedecimala siffrorna 0 till 9, A, B, C, D, E och F (samt vissa typografiska tecken)
2) genom kopiering av remsa (utföres med stansutrustningen)
3) av maskinen själv genom utstansning på snabbstans varvid en omställare på manöverbordet skall stå i läget "stans".

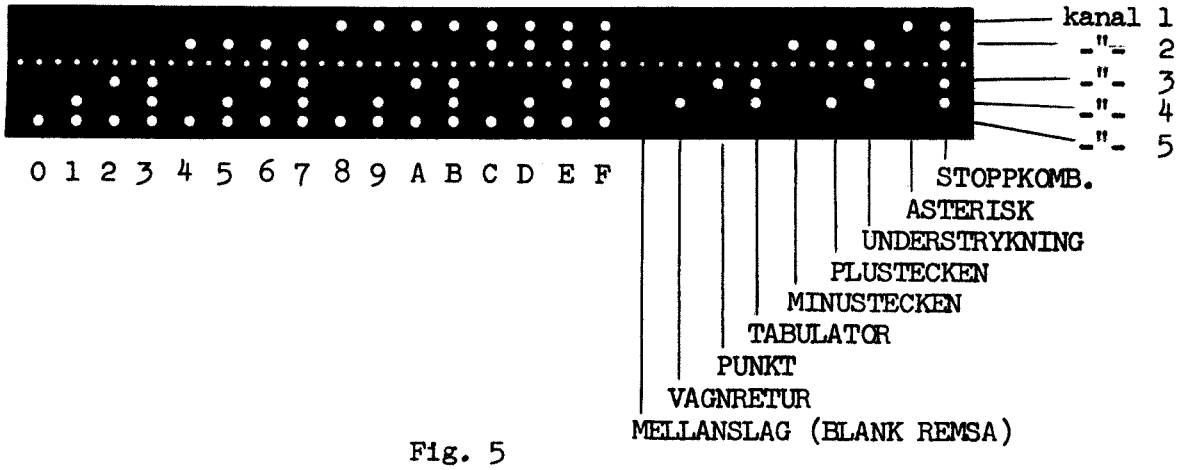


Fig. 5

På figuren finns en rad små hål något ovanför mitten på remsan. Dessa hål används som styrhål vid kopiering och utskrivning av remsa i stansutrustningen och saknar informationsvärde vid inläsning i Besk.

Bortsett från dessa hål kan 5 hål stansas i bredd på remsan. De 5 olika platserna för dessa hål kallas kanaler. Om ett hål finns stansat i nedre kanten, säges det befinna sig i kanal 5. Läget av kanal 1, 2, 3 och 4 framgår av figuren. Om ett hål är stansat i kanal 5, uppfattar maskinen detta vid inläsning som en indikation, att information finns lagrad på remsan i kanalerna ovanför hålet. Denna information består av en av de sedecimala siffrorna 0 till F. (Kanalerna 1 till 4 kan tolkas som binära positioner; kanal 4: enhetsposition, kanal 3: 2-talsposition, kanal 2: 4-talsposition o.s.v. Ett hål motsvarar den binära siffran 1, avsaknad av hål den binära siffran 0.) Platsen på remsan för denna information kallas en rad. Även varje avsnitt av remsan, där plats finns för en rad (vid varje styrhål), kallas en rad. På figuren förekommer rader, där kanal 5 ej är stansad. Vid utskrivning av remsan på skrivmaskin ger dessa rader olika typografiska operationer t.ex. vagnretur och mellan-

slag. På remsan har typografiska tecken, stansade av Besk och stansade manuellt med stansutrustningen, i vissa fall olika utseende. Detta har till följd, att man måste ställa en omkastare på stansutrustningen i ett visst läge vid utskrivning av manuellt stansad remsa och i ett annat läge vid utskrivning av Beskremsa (d.v.s. remsa, som stansats av Besk). De sedecimala (decimala) siffrorna 0-F stansas däremot på samma sätt av Besk och av stansapparaten. De på figuren angivna typografiska tecken är de, som stansas ut av Besk. Det tecken längst till höger, som är märkt "stoppkombination", har till uppgift att stoppa utskrivning (eller kopiering) av remsa, då remsan är slut. (Praktiskt men ej absolut nödvändigt.) En omkastare måste slås till på stansutrustningen vid användning av detta tecken.

I detta sammanhang kan påpekas följande: om ett program är skrivet för utmatning på skrivmaskin, kan man, om man föredrar detta (för att vinna tid), få ut resultaten på remsa i stället genom att ställa omkastaren på kontrollbordet på "stans". Den remsa, man får, skrivs sedan ut som Beskremsa med stansutrustningen. Man behöver alltså inte göra några ändringar i programmet för att få ut resultat på remsa i stället för skrivmaskin.

Anm. Man kan stansa följande typografiska tecken manuellt: vagnretur (med radframmatning), tabulator, minustecken, plustecken, stopptecken, mellanslag samt blank remsa. Tecknen mellanslag och blank remsa på manuellt stansad remsa ser ut som punkt resp. mellanslag på Beskremsa. Övriga tecken har samma utseende på båda slagen av remsor.

9.3. Remsapparatur.

Remsläsaren, fig. 6, används för inläsning i maskinen av information från remsa. På dess framsida finns omkopplare för olika hastigheter, omkopplare för till- och frånslagning av broms samt tvåvägs återfjädrande omkopplare för fram- eller backspolning. Vidare finns det en del rullar, ett läshuvud och en remshållare.

Remsan inläses normalt med en hastighet av 400 rader/sek. men ibland, vid decimal inläsning, kan det vara nödvändigt att använda en lägre hastighet, 200 rader/sek., om man konstaterat felinläsning vid den högre hastigheten.

Remsorna förvaras i allmänhet upprullade på bakelitrullar. Då en remsa skall läggas in i remsläsaren, för man först bromsomkostaren i läge FRÅN. Därvid öppnas automatiskt locket på den elektromagnetiska bromsen och friktionsrullen föres upp från framdrivningsrullen. Den till höger om omkastaren placerade

röda lampan lyser i detta läge. Därefter sätter man den upprullade remsan i dess hållare och för remsan över två glidrullar och över den elektromagnetiska bromsen samt genom läshuvudet och sedan mellan framdrivnings- och friktionsrullen. Slutligen för man bromsomkastaren i läge BROMS, varvid remsan fixeras i sitt läge. Allt är nu klart för inläsning.

Att observera: kanal 5 på remsan skall ligga utåt (närmast den som lägger in remsan), kanal 1 inåt, då man trär in remsan i apparaten.

Remsor, som på ett eller annat sätt blivit skadade, t.ex. vid uppspolning, bör man inte läsa in i Besk, ty i regel får man då en felinläsning eller i sämsta fall remsbrott, som kan förorsaka kortare driftstopp, medan läshuvudet demonteras och befrias från pappersrester. En god regel är att alltid ha dubletter av remsorna.

Anm. För att undvika förväxlingar måste man märka alla remsor, vilket bor göras på ovansidan och framändan av remsan.

Snabbstansen, fig, 7, används för utmatning av information från maskinen. Den består av två delar, en styrkrets och den egentliga stansdelen. På styrkretsens framsida finns en återfjädrande omkopplare för frammatning av blankremsa och stansning av stoppkombination. Endast denna omkopplare får manövreras av personer, som inte tillhör den tekniska staben. Övriga omkopplare - för inställning av fördröjning och hastighet - manövreras av Beskpersonalen, som även ombesörjer insättning av ny remsa i snabbstansen. Snabbstansens hastighet är för närvarande omkring 150 rader/sek. Om längre räknetid än c:a 1 sek. förflyter mellan två stans(tryck-)order, uppträder fördröjningar (se kap. 13).

För manuell stansning och för kopiering av remsa används en separat anordning kallad stansutrustning, bestående av en stansmaskin och en elektrisk skrivmaskin, fig. 8. Med denna utrustning kan man dels tillverka en ny remsa, dels kopiera om eller skriva ut en redan befintlig remsa.

Stansmaskinen är försedd med 5 omkastare: NÄT, BESKREMSA, MÄRKT REMSA, SKRIVMASKIN och KOPIERING, 2 rattar: VAGNRETUR och TABULATOR samt ett vred med 3 lägen: STANSNING, KOPIERING och KOMPARERING. Innan utrustningen skall användas, tillse att omkastaren NÄT står på TILL.

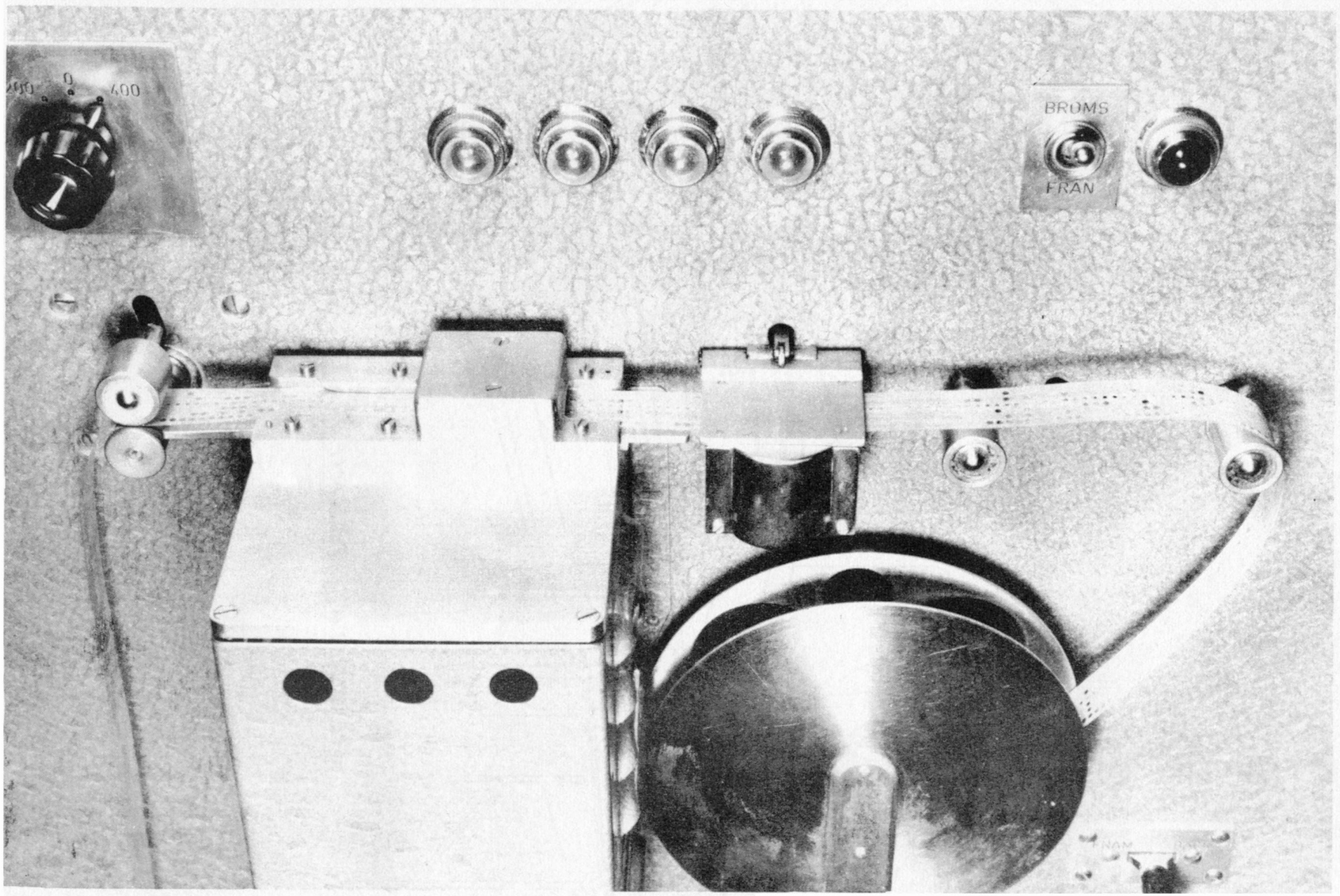


Fig.6 Inorgan

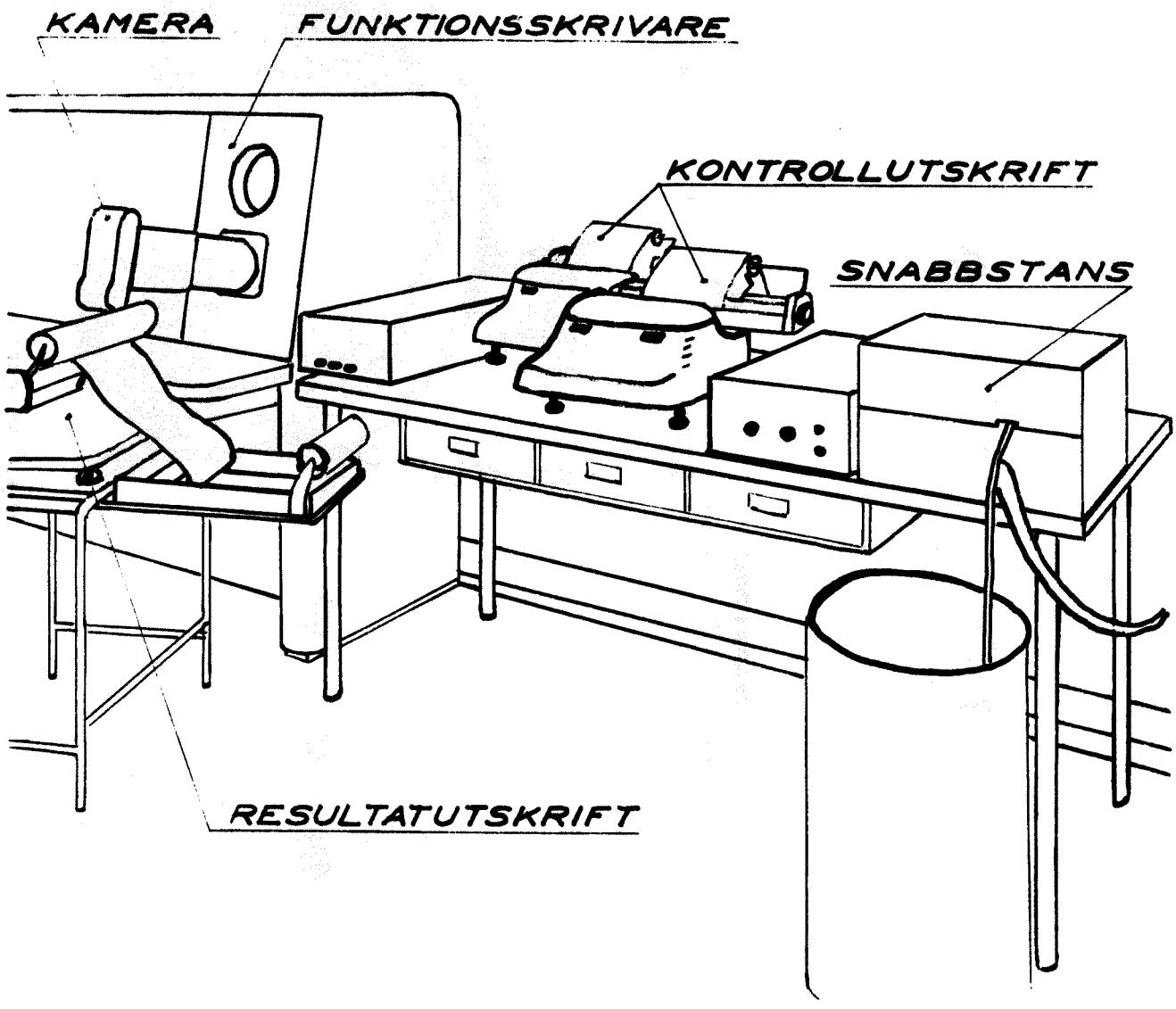

KAMERA
FUNKTIONSSKRIVARE
KONTROLLUTSKRIFT
SNABBSTANS
RESULTATUTSKRIFT

Fig. 7 Utorgan

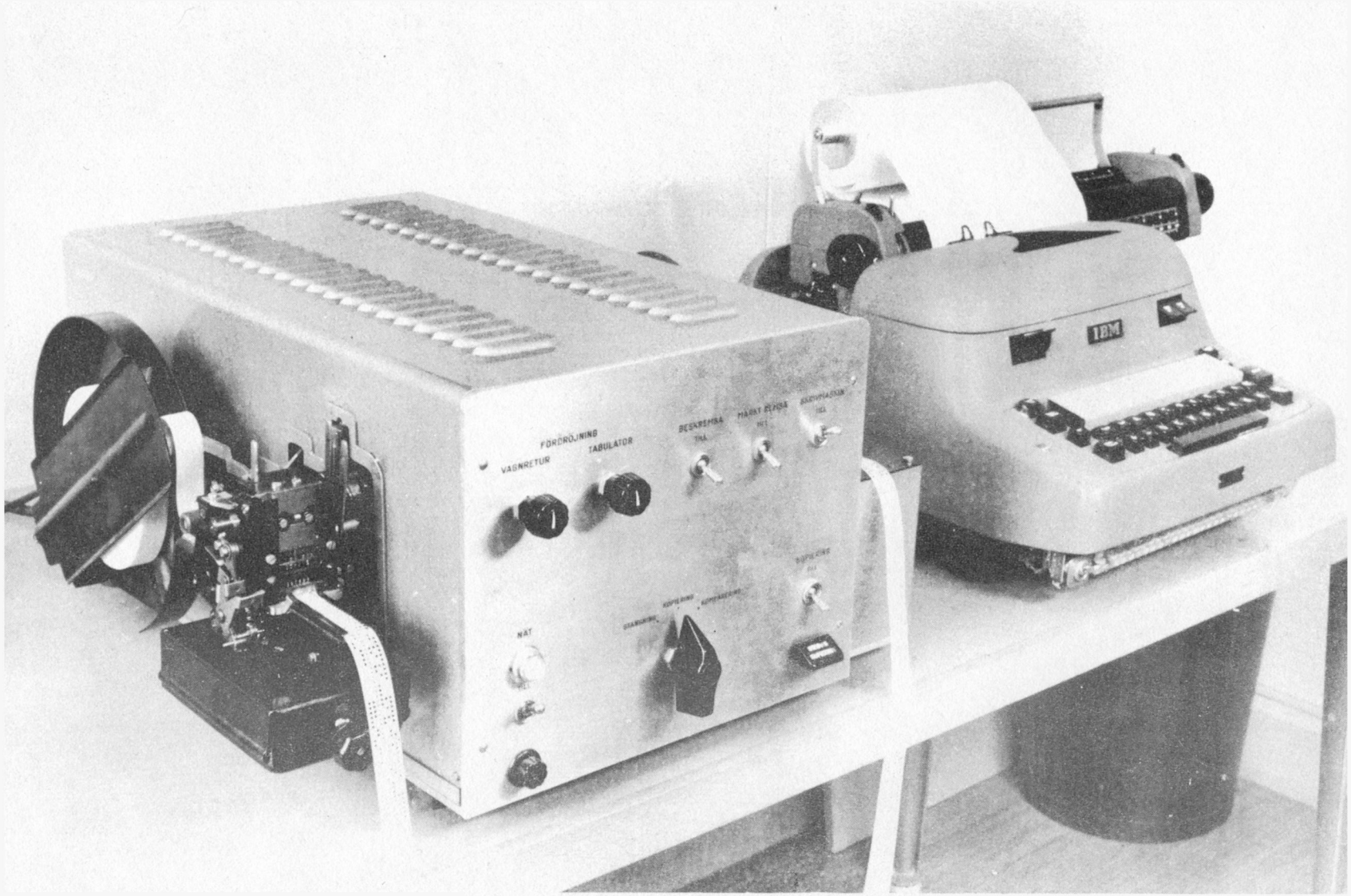


Fig. 8 Stans och kopieringsapparat

Stansning. Omkastaren SKRIVMASKIN skall stå på TILL, vredet på STANSNING. Ett par decimeter remsa framdrages för hand, samtidigt som den yttersta spaken på stansmekanismen tryckes bakåt. Därefter måste man ovillkorligen låta maskinen mata fram blank remsa minst 1 dm genom samtidig tryckning på tangenterna BLANK och REPETERING, innan första raden stansas för att undvika, att papperet sid förskjuts i sin bana. En dylik förskjutning kan göra hela remsan oanvändbar, enär hålen då ej kommer mitt på remsan, vilket har till följd, att Besk ej kan läsa dem.

Själva stansningen sker helt automatiskt vid tryckning på skrivmaskinens tangenter. Utskriftens utseende beträffande radlängd, radavstånd o.d. kan varieras hur som helst, då Besk ej läser typografiska tecken, mellanslag, vagnretur eller tabulator.

När remsan är färdigstansad, är det lämpligt, att låta maskinen göra en liten bit blank remsa liksom i början, för att vid en ev. omkopiering de sista två raderna överhuvud taget skall bli återgivna.

Kopiering. Vid kopiering eller utskrift av en remsa måste man veta, om det är en Beskremsa eller en vanlig dito. Är det en Beskremsa, skall omkastaren, märkt BESKREMSA, stå på TILL, är det en vanlig remsa, skall den stå i motsatt läge. Om utskrift på skrivmaskin önskas, skall omkastaren SKRIVMASKIN stå på TILL. Härvid får rattarna märkta FÖRDRÖJNING/VAGNRETUR och TABULATOR justeras efter behov. Normal inställning för A4-papper är med pilarna rätt upp. Om bredare papper användes får man öka fördröjningen genom att vrida ratten till höger. Är remsan försedd med stopptecken (ett F utan hål i kanal 5), skall omkastaren märkt MÄRKT REMSA stå på TILL. I annat fall reagerar apparaturen ej för stoppmärkningen.

Innan omkastaren märkt KOPIERING slås över på TILL, bör man som vid stansning se till, att det blir minst 1 dm blank remsa. Skall en remsa ändras på något sätt, kopierar man om den till det ställe, där den skall ändras. Sista biten kan man köra för hand genom att trycka på tangenten STEGVIS KOPIERING, varefter vredet vrides över på STANSNING och det är klart att stansa. Märk väl, att originalremsan går fram en rad för varje tecken, som stansas.

Om utrustningen inte fungerar enligt beskrivningen, skall Beskskötaren omedelbart meddelas.

Card-to-tape-utrustningen. För överföring av information från hålkort till remsa finns en särskild apparatur på MNA, card-to-tape-utrustningen, bestående av en hålkortsmaskin "IBM Printing Card Punch, Type 26", kombinerad med en vanlig stansutrustning för utskrivning och stansning av remsa.

Stansning av kort. Vid stansning av kort och manövrering av tangentbordet bör instruktionerna i IBM:s beskrivning av "Printing Card Punch, Type 26" följas med följande undantag:

a) Multi Punch fungerar ej.
b) Duplicering av kort kan ej utföras (för duplicering av stort antal kort kan tillfälligt en omkoppling göras så att dup. fungerar).
c) "Printing" kan ej utföras (omkopplaren "Print" användes för till-och frånkoppling av översättningen).

Läsning av kort. Då kortet av matningsmekanismen föres från höger till vänster passerar detta först en stansstation och en kortdelning och till vänster därom en lässtation.

I det fall att kortens hela innehåll skall översättas till remsa frikopplas avkänningsanordningen från programtrumman. Före start av översättningen bör omkopplarna "Auto Feed", "Auto Skip and Auto Dup" och "Print" stå i läget "on". Vidare bör omkopplaren på remsutrustningen stå i läget för kopiering. Vid start matas första kortet fram så att dess 1:a kolumn ligger under lässtationen. Detta åstadkommes genom tre (3) tryckningar på tangenten "REL." på tangentbordet. Därefter ställes omkopplaren på remsutrustningen i läget för stansning. Sedan sättes översättningen i gång genom att omkopplaren "Print" ställes i "off"-läge. Härvid läses kortets innehåll kolumn för kolumn samtidigt som motsvarande tecken stansas på hålremsan. Översättningen fortsätter sedan till och med det näst sista kortet, varefter maskinen stannar. För att även det sista kortet skall läsas, bör man lägga 1 st. blankt kort sist i högen. Detta bör göras innan översättningen startas men kan även göras efteråt genom att det blanka kortet matas in med hjälp av tangenten "Feed".

Programmering av översättningen. I de fall då endast vissa delar av kortens innehåll skall överföras till hålremsa och för indikation av tecken kan översättningen programmeras. På hålkortsmaskinen finnes en programtrumma på vilken ett mot programmet svarande programkort kan upplindas.
De programmeringsmöjligheter, som finnas, äro följande:

a) Blankt kort ger översättning.
b) Ett hål i pos. 11 innebär, att motsvarande kolumn ej översättes.

c) Ett hål i pos. 11 omedelbart följt av ett eller flera hål i pos. 12 innebär, att samtliga kolumner från och med 11-hålet t.o.m. sista 12-hålet ej översättes.

d) Ett hål på programkortet i pos. 3 innebär, att motsvarande kolumn först översättes, varefter på remsan stansas ett B eller D beroende på om det lästa kortet har eller inte har ett hål i pos. 11 i den aktuella kolumnen. Hålet i pos. 11 på det lästa kortet betyder som bekant, att det just lästa talet har neg. tecken.

Stansning av programkortet. Ett hål i pos. 11 erhålles genom att tangenten "Skip" nedtryckes. Ett hål i pos. 12 erhålles genom att tangenten "Num" hålles nedtryckt samtidigt som tangenten "P" nedtryckes. Ett hål i pos. 3 erhålles genom att samtidigt som tangenten "Num" hålles nedtryckt tangenten "3" nedtryckes. Blankt kort erhålles genom att "Space"-tangenten nedtryckes. Obs! Se till att programkortet sitter rätt i hållaren. Obs!

Vilka tecken kan översättas? Siffrorna 0-9 kan överföras från kort till remsa. Vidare vid tecken och endast vid tecken bokstäverna B eller D. I det fall man önskar överföra bokstäverna A, B, C, D, E och F till remsa kan detta göras endast om motsvarande kombinationer stansas på kortet. Därvid ger stansning av följande hålkombinationer i samma kolumn:

8	och	2	- A
8	"	3	B
8	"	4	C
8	"	5	D
8	"	6	E
8	"	7	F

Vidare ger alltid "skip" (hål i pos. 11 på programkortet) kombinationen för mellanslag på remsan.
Matning av nytt kort ger kombinationen för vagnretur på remsan.

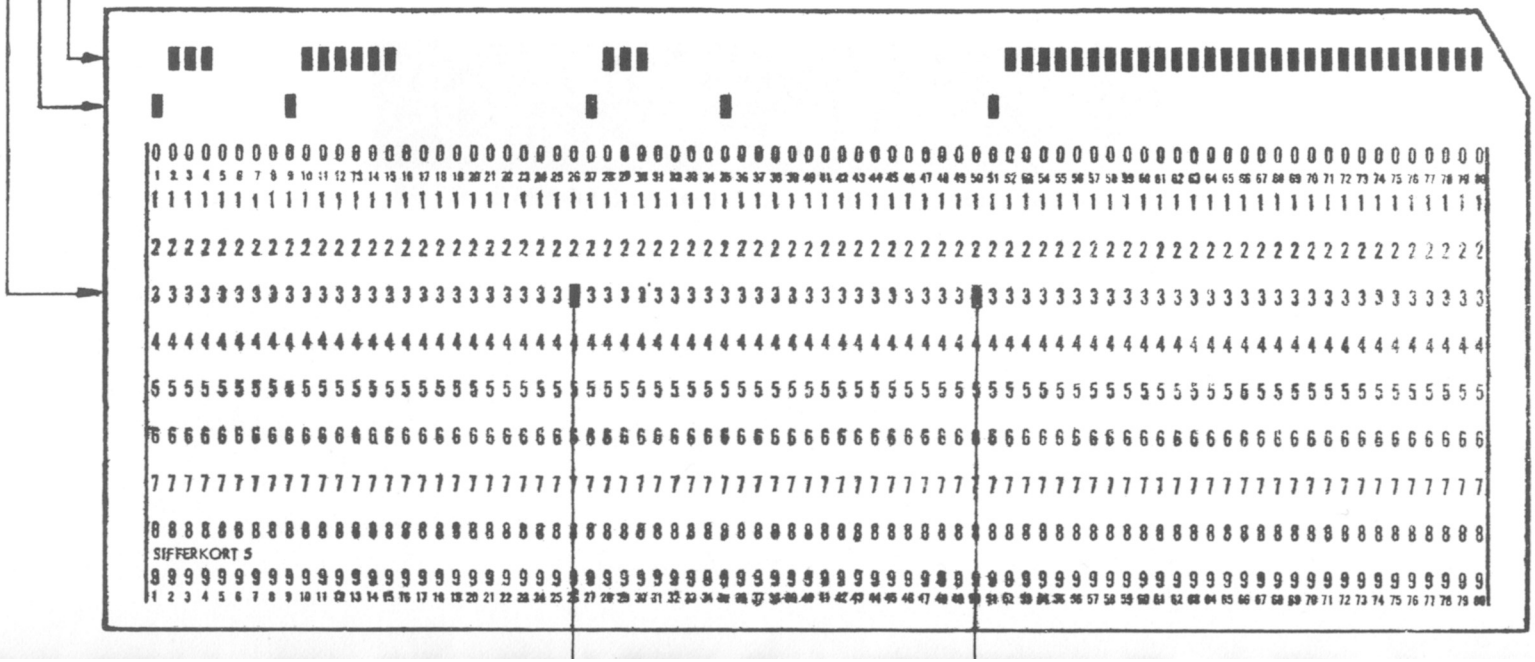
Anslutning av el. skrivmaskin. Samtidigt som remsan stansas, kan avskrift erhållas på skrivmaskin. Det enda, som därvid behöver iakttagas, är , att tillräckligt lång fördröjning för vagnreturen utväljes. Vagnretursfördröjningen omkopplas med en vippströmbrytare, placerad på baksidan av hålkortsapparaten på den inbyggda tillsatsenheten.

Exempel på översättning från kort till remsa. Antag, att på varje kort följande kolumner skall överhoppas: 1-4, 9-15, 27-30, 35, 51-80. I kolumn 26 och 50 står på varje kort tecken stansat jämte en siffra. Programkortets utseende framgår av figuren, vilken också visar exempel på hur ett kort översatts.

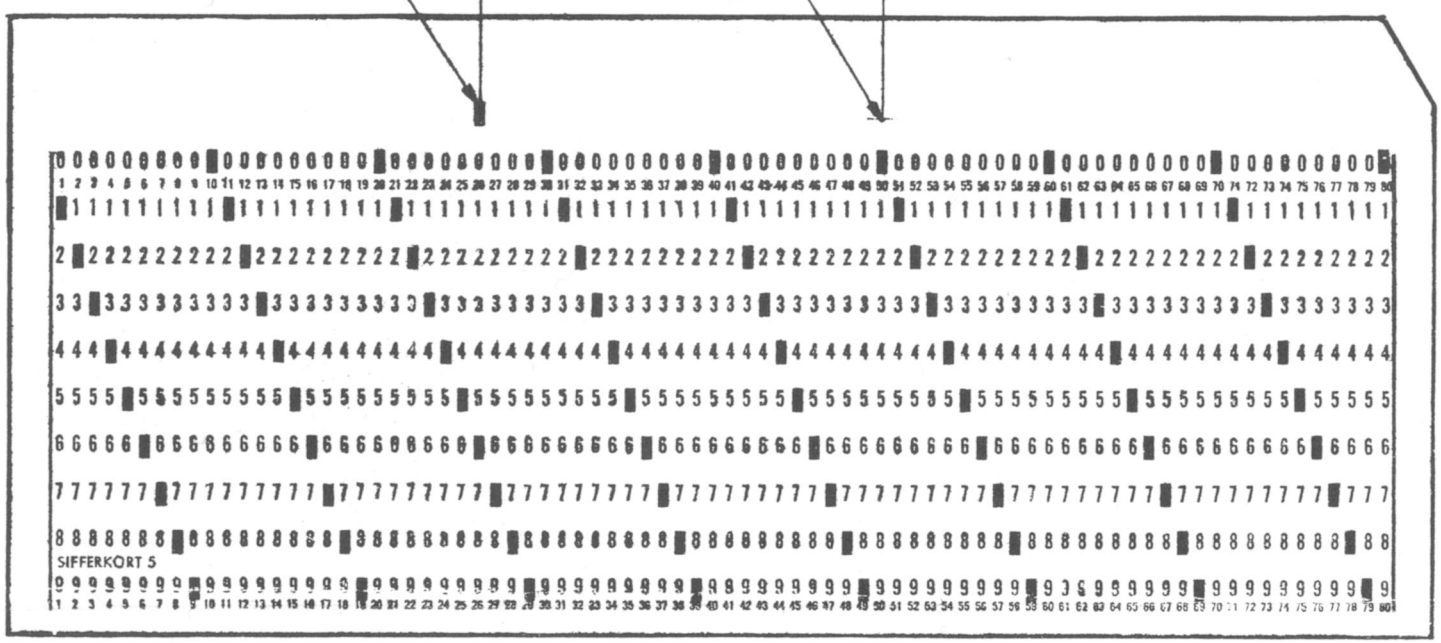
Exempel på översättning från hålkort till remsa.

Programkort

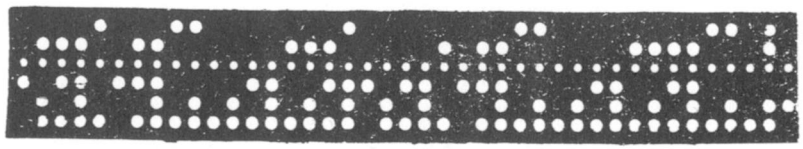
Pos. 3 - tecken
Pos.11 - hoppa över
Pos.12 - fortsätt att hoppa



Kort för avläsning och översättning



Perforerad remsa samt utskrift på skrivmaskin



5678 67890123456B 1234 67890123456789OD

Beskrivning av översättare från telex-kod till Besk-kod. För att bereda kunderna vid Besk möjlighet att via telex översända koder och data till Besk (BESK STH, telexnr 1613) har en av de vid maskinen placerade remsutrustningarna försetts med en tillsatsenhet för översättning från telex-kod till Besk-kod. Följande bokstäver, siffror och tecken översättas. De få den betydelse i Besk-koden, som tabellen visar.

Telex-kod	Besk-kod
oo. A	o .o o A
o . oo B	o .ooo B
o.oo C	oo. o C
o . o D	oo. oo D
o . E	oo.o o E
o .oo F	oo.ooo F
o .o S	oo.oo Stoppkomb. (F utan 5:e hål)
. o T	.oo Tabulator
o.o o 0	. o 0
oo.o o 1	. oo 1
oo. o 2	.o o 2
o . 3	.ooo 3
o. o 4	o. o 4
. o 5	o. oo 5
o .o o 6	o.o o 6
oo.o 7	o.ooo 7
o.o 8	o . o 8
. oo 9	o . oo 9
o . o +	o. o +
oo. -	o. -
. o Vagnretur	. o Vagnretur
.ooo .	.o .
.oo ,	.o .
o .o ´	o . Asterisk
o. Radmatning	oo. Spilltecken (C utan 5:te hål)
oo.ooo Skiftning till bokstäver	-"-
oo. oo -"- -"- siffror och tecken	-"-
.o Mellanslag	Mellanslag - Blank remsa

Samtliga bokstäver, siffror och tecken, som ej medtagits ovan ger Blank remsa och därmed Mellanslag i Besk-koden.

För att åstadkomma viss enhetlighet i samband med översättningen föreslås följande standardförfarande:

All text skrives i enlighet med de instruktioner som gäller för telex. För att emellertid tydligt avskilja koder och data från övrigt text låter man varje sammanhängande kod eller grupp av data föregås av ett antal mellanslag, motsvarande en remslängd av c:a 3 dm. Efter mellanslag, tryckes Start, Retur, Ny Rad samt återigen Start, varefter skrivningen kan börja. Sedan man skrivit den grupp av tecken, som skall översättas, avslutar man med ett antal mellanslag motsvarande c:a 5 dm remsa. Därefter tryckes Start, Retur, Ny Rad, Start samt skrives Slut.

En remsa behandlad enligt ovan ger i översättaren en remsa med 3 dm blank remsa i början och 5 dm i slutet. Översättaren stannar automatiskt, då den läser tecknet för S i det Slut som avslutar remsan.

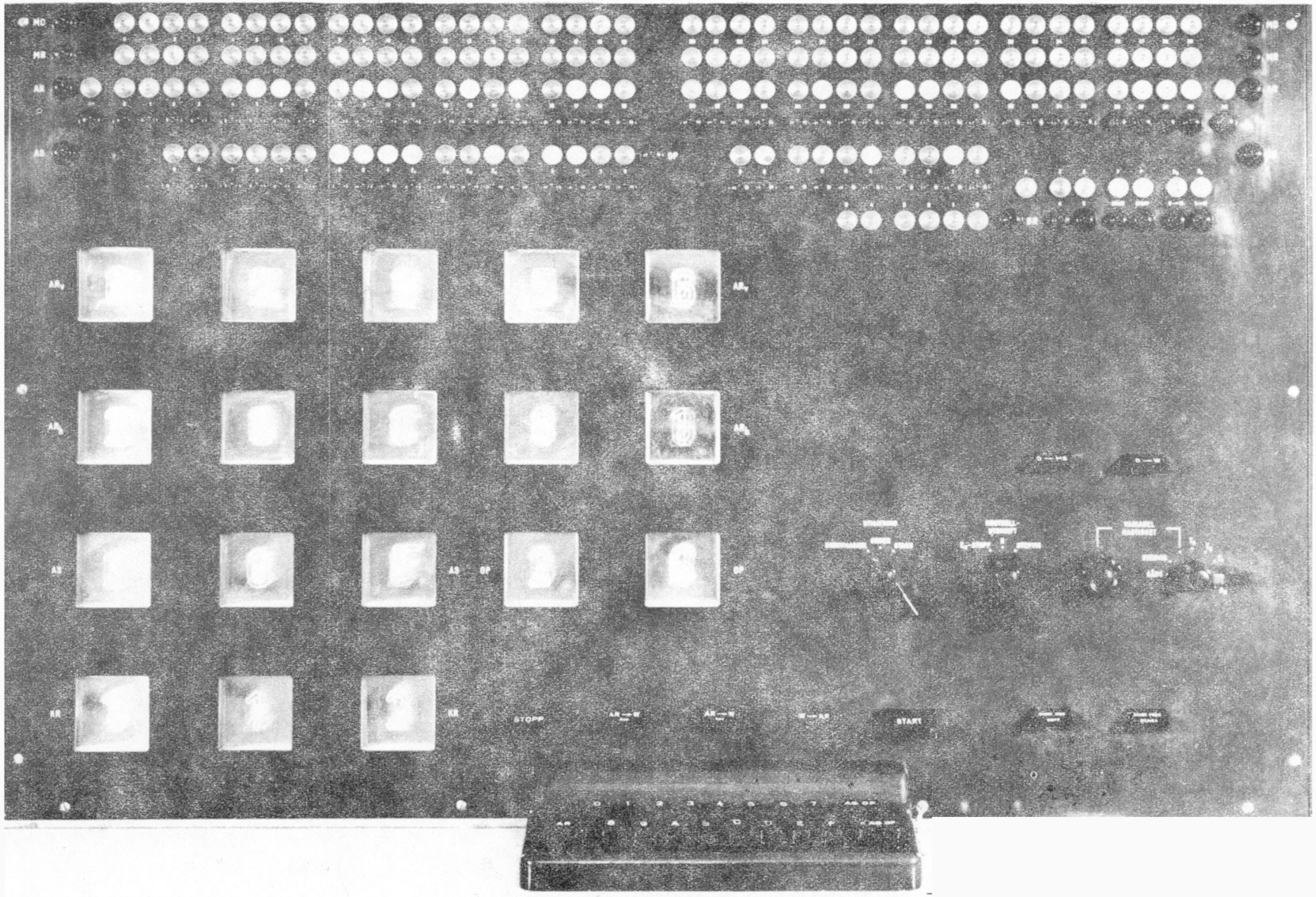


Fig. 9 Manöverbord

10. INKÖRNING AV PROGRAM.

10.1. Manöverbordet. Till Besk hör ett manöverbord för det manuella handhavandet av maskinen, som innehåller start- och stoppanordningar, indikeringslampor för de olika registren i maskinen samt knappar för manuell inmatning till dessa. (Obs! att KR och AS nu innehåller en pos. mer än på fig. 9). Där finns 18 st. kontrollfönster, som visar sedecimalt innehållet i KR, AS, OP och AR, samt ett sedecimalt tangentbord för manuell inslagning i dessa register. Vidare finns det vred för olika slag av utmatning, kontrollutskrift, variabel hastighet och stegvis körning. Förutom en mängd för teknikerna erforderliga vred, knappar och indikeringslampor kan nämnas en högtalare, kopplad till aritmetiska enheten, i vilken man kan höra rytmen av de olika sekvenserna i programmet. Man kan t.ex. höra om maskinen råkat fastna i en räkneprocess, som ej konvergerar.

Nedan skall en del exempel ges på när och hur de olika vreden och knapparna skall användas.

Start och stopp. När maskinen befinner sig i normalt stopptillstånd lyser kontrollfönstren. Vid tryck på knappen START sker då följande:

1. Den order, som står i AS och OP, utföres i den omfattning, som framgår av mikroorderschemat (sid. 13.12 f.)
2. C(KR) ökas med ett eller (vid hopporder) C(AS) överföres till KR.
3. Innehållet i den halvcell, vars adress anges av KR, överföres till AS och OP.
4. Den order, som nu står i AS och OP, utföres och maskinen fortsätter på detta sätt med full räknehastighet (vid inställning på "gång", se nedan).

Under räkningarnas gång lyser inte kontrollfönstren. Stopp kan erhållas genom att maskinen träffar på en stopporder i räkneprogrammet eller genom tryck på STOPP-knappen. Före stopp utföres den del av ordern, som framgår av mikrooperationsschemat. Härvid lyser kontrollfönstren och visar stoppordern samt innehållet i AR. Man kan även erhålla s.k. mörkt stopp, d.v.s. kontrollfönstren lyser inte vid stopp. Detta betyder, att Besk har råkat på en omöjlig order, t.ex. inläsningsorder, när ingen remsa finns i remsläsaren, och man stoppar maskinen genom tryck på (knappen 0→MS och) den röda g-knappen (se även kap. 3).

Inläsning av remsa.

1. Tryck på knappen 0→MS (knappen till höger om denna på fig. 9 märkt 0→W är utbytt mot en ograverad sådan och saknar för närvarande funktion). 0→MS innebär, att alla registren blir nollställda men observera att kärn-

och trumminnet icke beröras av denna manöver. För att nollställa dessa måste man läsa in separata nollställningssekvenser, som finnas tillgängliga vid Besk. Märk dock, att inläs-ningssekvenserna A40, A41 och A42 nollställer snabbminnet automatiskt före inläsningen.

2. Tryck på knappen KASOP. Om man sedan på tangentbordet trycker t.ex. 00039 får man 000 i KR- och AS-registren och 39 i OP-registret.
3. Vid tryck på knappen START inläses automatiskt den i inorganet placerade remsan, förutsatt att den börjar med någon av inläsningssekvenserna A40, A41 eller A42 (se kap. 9).

Nu är emellertid denna inmatningsoperation 39, läs ord från remsa, inbyggd i maskinen och utföres genom tryckning på knappen START FRÅN REMSA. Därför räcker det om man trycker på "O→MS" och sedan på START FRÅN REMSA; detta förfarande är det vanliga.

Start med hopp. Möjlighet finns att hoppa till önskad plats i programmet. T.ex. man önskar hoppa till halvcell 007. Detta kan göras på följande sätt. Knappen ASOP nedtryckes, varefter man på tangentbordet trycker 0070C och sedan trycker på knappen START. Denna hopporder är också inbyggd i maskinen och utföres enklast genom att i AS sätta önskad adress, i detta ex. 007 och sedan trycka på knappen START MED HOPP.

Manuell insättning i AR. Insättning i AR kan göras manuellt från manöverbordet enligt följande.

1. Man nollställer AR genom att trycka på tangentbordets knapp AR.
2. Det önskade talet sätter man i AR genom att trycka på tangentbordets knappar.

Manuell insättning i kärnminnet. Ex. Sätt i helcell 112 ordet 0E656 24C81.

1. Tryck på knappen ASOP, detta innebär nollställning av dessa register.
2. Tryck på tangentbordet siffrorna 112, dessa siffror kommer då att stå i AS-registret.
3. Tryck på knappen AR, ackumulatorn nollställes.
4. På tangentbordet slår man in det önskade helordet.
5. Nu står i AS 112 och i AR 0E656 24C71. Om man nu trycker på knappen AR→Whec så överföres AR:s innehåll till cell 112 i kärnminnet(kr ökas därvid med ett).

Man kan även överföra halvord från AR till kärnminnet. Detta går till på liknande sätt. Vid insättning i hhac t.ex. slår man i ARV in nollor, i ARH halvordet och i AS halvcellen och själva överföringsoperationen sker genom att trycka på knappen AR→Whac.

Manuell överföring från kärnminnet till AR. Man vill t.ex. se vad som står i helcell 200.

1. Tryck på knappen KASOP, kontroll-, adress- och operationsregistren nollställes.
2. Tryck på knappen ASOP och slå på tangentbordet in siffrorna 200; nu står i AS 200, alltså den helcell i kärnminnet, som skulle inspekteras.
3. Tryck på knappen AR, ackumulatorn nollställes.
4. Tryck på knappen W AR. Man kan nu i AR avläsa vad som står i helcell 200.

   (kr ökas till 001. OBS denna hac får inte innehålla en nollställnings-order, jfr sida 3.10 nederst.)

Utmatningsväljare. På manöverbordet finns en omkopplare, UTMATNING, som har tre lägen, SKRIVMASKIN, ORDER och STANS. När omkopplaren står i läge SKRIVMASKIN får man resultatet på en skrivmaskin med en medelhastighet av 12 tecken per sekund. Med omkopplaren i läge ORDER fås ingen utskrift. Slutligen med omkopplaren i läge STANS erhålles resultatet på remsa från en snabbstans med en hastighet av för närvarande c:a 145 tecken per sek.

Utmatning kan även ske på en funktionsskrivare, i fortsättningen kallad FS, som medger utmatning av resultatet i kurvform. En utförlig beskrivning av FS ges i kapitel 6. Utöver denna är det dock nödvändigt att känna till följande.

När FS skall användas, skall kunder eller körledaren före körningens början meddela Beskskötaren, att FS skall användas eller, om Beskskötaren skall utföra körningen, skall sådan upplysning finnas angiven i kör-instruktionen. Det ligger nämligen till på det viset, att om man medan maskinen räknar slår till eller från spänningarna för FS kommer detta att förorsaka en urspåring, vilket har till följd, att pågående körning får starta från början.

Normalt står FS:s rattar med pilen upp. Skötsel av kameran ombesörjes i regel av Beskskötaren.

Kontrollutskrift. På manöverbordet finns ytterligare en omkopplare, KONTROLLUTSKRIFT, som kan ställas i ettdera av tre lägen. Om den står i läge E_2-STOPP skrivs för var order, där operationsdelens första bit (den s.k. E_2-siffran) är 1, innehållen i KR, AS, OP och AR ut på en speciell skrivmaskin. Om omkopplaren står i mittläget O har siffran E_2 ingen inverkan på maskinens funktion. Om omkopplaren slutligen står i läge STEGVIS får man utskrift för varje order med innehållen i KR, AS, OP och AR på en skrivmaskin. Denna möjlighet är ibland värdefull vid inkörning, men den bör användas med omdöme eftersom den är tidsödande.

För underlättande av felsökning finns ytterligare två möjligheter. Man kan nämligen låta maskinen arbeta steg för steg med stopp mellan varje order eller med en variabel hastighet. Inställning och reglering av detta sker med omkopplare och ratt, som sitter omedelbart till höger om omkopplaren KONTROLLUTSKRIFT.

Allmänt om manuella operationer. För att eliminera manuella feloperationer är det nödvändigt att ha en utförlig körinstruktion. Detta gäller framför allt när Beskskötaren ombesörjer körningen. Om ett flertal rättelser eller korrigeringar skall göras, är det bättre att stansa en rättelseremsa och läsa in den än att utföra rättelserna manuellt. Det har nämligen visat sig, att vid sådana manuella operationer en del fel kan uppkomma på grund av att man trycker på fel knapp. Vid osäkerhet om utförandet av manuella operationer, t.ex. insättningar till minnet eller utskrifter av kärn- eller trumminnet, låt då Beskskötaren göra detta.

Om av någon anledning maskinfel misstänkes, t.ex. en manuell operation icke fungerar, anmäl då detta genast till Beskskötaren.

Med lite träning och kännedom om denna instruktion kan kunden snart använda manöverbordets vred och knappar och om så önskas själv utföra körningen av Besk.

10.2. Allmänt om inkörning. Då man avslutat kodningen av ett problem och erhållit den korrekturlästa programremsan av stanserskan, vidtager den ofta smärtsamma process, som kallas inkörning, d.v.s. man försöker få sitt program att fungera på Besk. Härunder finner man fel av de mest skiftande slag. Vi skall inte här syssla med de mera allvarliga fel, som beror på grundläggande fel i den matematiska behandlingen av problemet, utan endast rena kodningsfel och slarvfel. Redan vid uppläggningen av programmet bör man ge akt på de möjligheter till kontroll av delresultat (av typen $\sin^2 x + \cos^2 x$ bör vara nära 1), som kan byggas på inneboende egenskaper hos de funktioner man sysslar med. När väl sådana självcheckar är inbyggda i programmet (och programmet fungerar), tjänar de till att kontrollera, att Besk fungerar riktigt. Längre program kan alltid delas upp i sina logiska beståndsdelar, och dessa bör provköras var och en för sig.

Till hjälp för inkörningen finnes i en låda vid Besk ett antal självinläsande

programremsor, vilkas funktioner framgår av bifogade beskrivningar. De flesta kan endast användas med fördel under inkörning (C 10, 12, 20 och 21). Andra är avsedda att kunna användas även vid reguljära körning, för att man skall kunna fortsätta en körning efter ett avbrott (C 30, 31). Dessutom finns en sekvens för nollställning av trumminnet (C 13) samt en för komparering av remsor (C 15).

Det finns även två program i 800-form, som kan användas för felsökning, när man råkat ut för särskilda besvärligheter (C 2 och 3). Med C 2 kan man få sedecimal eller decimal utskrift av valfria celler när som helst under räkningarnas gång, C 3 signalerar under räkningarnas gång varje gång en cell i ett valfritt område i W-minnet tas i anspråk som adress.

Kodningsfel och rena slarvfel uppträder (nästan) varje gång man provkör en ny programremsa (eller längre rättelseremsa). Några av de vanligaste följer här.

Etikettsfel, d.v.s. remsan "går inte in". De hör till kategorin onödiga fel och undvikes genom kontroll av remsan före körningen.

Standardprogramfel i allmänhet, d.v.s. man glömmer parametrar, slaskceller, som standardprogrammet använder, används i huvudprogrammet o.s.v.

Parentesfel. Alla konsekvenser, som ett tillägg till koden har, har inte beaktats.

Halvordsfel.

Adressmodifikationsfel, d.v.s. fel adress modifieras eller rätt adress modifieras fel (eller bådadera). Det första alternativet kan bemästras genom att man i koden sätter den fiktiva adressen 7FF eller FFF i de adresser, som skall modifieras, varvid Besk eventuellt stannar där, och det hela uppdagas, eller man lätt ser det, då man granskar en utskrift av minnet.

Divisionsfel av två slag. Operation 13 bortglömd eller täljaren trots alla försiktighetsmått större än nämnaren.

Hoppfel av flera slag. Wheelerhopp felkodat, fel adress i hoppet eller hoppet ur en cykel åt "fel håll" (OE i stället för 4E exempelvis).

Spill uppträder, där man icke väntat det.

Under inkörningen använder man med fördel E_2-utskrift. Man kan därigenom skaffa sig en uppfattning, om vilka delar av programmet som genomlöpes, innan det hela hänger upp sig. E_2-stoppen bör inte läsas in manuellt (undvik manuell körning!), utan man har på förhand planerat var man vill ha dem och läser in dem respektive över dem med rättelseremsa.

Följande allmänna regler bör iakttagas vid inkörning. Beställ om möjligt högst 15 minuter för den första inkörningen. Se till att remsorna är "manikurerade", så att de går att sätta i remsläsaren utan onödig tidsförlust. O-ställ W-minnet och ev. trumminnet (C 13) före inläsningen. När maskinen har stannat eller det är uppenbart, att den löper runt i en cykel utan att kunna komma ut, anteckna då var den befinner sig och beställ av Beskskötaren det C-program, som bedömes lämpligast (planerat på förhand). När utskriften genom detta C-programs försorg är klar, kan man (läsa in remsorna på nytt och) starta på något annat lämpligt ställe för att finna ytterligare fel och så tillämpa ett (annat) C-program. Det är viktigt att försöka få ut så mycket information ur Besk som möjligt, om vad som händer i programmet; man har sedan god tid att fundera över utskrifterna. Det är således förkastligt att slösa tid vid Besk med att försöka manuellt rätta enskilda fel under inkörningen. Kan man inte utnyttja tiden är i regel ingen skada skedd; andra är tacksamma för några extra minuter.

10.3. Kort beskrivning av de självinläsande elementära hjälpsekvenserna.

C 10 Sedecimal utskrift med adress av kons. helord, 2 versioner.
Utrymme i W: 000-03B (början) resp. 7C0-7FF (slutet).
Start för inläsningen: 000 resp. 7FC. Begynnelseadressen för utskrivningen sättes i adresspos. i ARV, slutadressen i de 3 följande sedecimala positionerna varefter man trycker på start.
Omstart: 009 resp. 7C1.

C 12 Sedecimal utskrift av kons. trumkanalers innehåll.
Utrymme i W: 780-7FF.
Start för inläsningen: 7FC. Begynnelsekanal i adresspos. i ARV och sedan startknappen.
Omstart: 7FC.
Man kan vid inläsningen välja mellan 1/1, 3/4, 1/2 eller 1/4 kanal.

C 13 O-ställning av trumminnet.
Utrymme i W: 000-035, 7C0-7FF.
Start = start för inläsningen: 000.

C 15 Komparering av remsor.
Utrymme i W: 000-0BD (förstör perm.konst.)
Start för inläsningen: 000.
" " remsa I: 001; remsa II: 000.
E2-utskrift samt stopp vid olikhet i ett helord på remsa I och II.

C 20 Decimal utskrift med adress, tecken och valfritt antal decimaler av kons. helord (utan avrundning), 2 versioner.
Utrymme i W: 000-039 (början) resp. 7CE-7FF (slutet).
Start för inläsningen: 000 resp. 7FC. Begynnelseadress i adresspos. i ARV, antal decimaler i AR pos. 19 och sedan startknappen.
Omstart: 009 resp. 7FC.

C 21 Decimal utskrift med tecken och valfritt antal decimaler av kons. trumkanalers innehåll (utan avrundning).
Utrymme i W: 784-7FF.
Start för inläsningen: 7FC. Begynnelsekanal i adresspos. i ARV, antal decimaler i pos. 19 och sedan startknappen.
Omstart: 7FC.
Man kan vid inläsningen välja mellan 1/1, 3/4, 1/2 eller 1/4 kanal.

10.4. Speciella hjälpprogram.

C 2 Hjälp vid kodcheckning. Ger antingen decimal eller sedecimal utskrivning av godtyckligt föreskriven del av K-minnet när som helst under räkningarnas gång med hjälp av modifierat Wheelerhopp.
Utrymme i W: 800-841.
(Identisk med sekvens 924 i kap. 9).

C 3 Hjälp vid felsökning. Utför det program man vill testa order för order och ger E2-utskrift av KR, ASOP, AR och om så önskas även MR för varje order, vars adressdel ligger mellan två på förhand uppgivna adresser.
Utrymme i W: 800-855
(Se standardprogrambeskrivningen.)

C 30 Åstadkommer en självinläsande remsa med Σ-check och ind. utskrift, som reproducerar valfria delar av W-minnet.
Utrymme i W: 000-007, 7E4-7FF (förstör perm. konst.).
Start: Omkastaren "utmatning" på "stans", tryck fram några decimeter blank remsa från snabbstansen och starta från 000. Efter inläsningen sätter man

en etikett (bbb39sss0n) i AR och trycker på startknappen. Då maskinen åter stannar, kan man sätta en ny etikett i AR om man så önskar eller om allt är klart "-1", varpå man trycker på startknappen. Rör ej remsan C 30 förrän allt är klart, den räcker till 3 etiketter.

C 31 Åstadkommer en självinläsande remsa med Σ-check och ind. utskrift, som reproducerar valfria delar av trumminnet.
Utrymme i W: 780-7FF, 000-007, 00E-017.
Start: Som vid C 30, dock att varje etikett skall innehålla begynnelsekanal och slutkanal (udda) i stället för adresser. Operationsdelen i etikettens vho är likgiltig men sättes lämpligen till 7F, sista sedecimala siffran skrives ut vid inläsningen. När allt är klart, sättes "-1" i AR, varefter man trycker på startknappen.

10.5. Normalläge II utgör en omarbetning av normalläge I, beskrivet i PM 2 för Beskkodare, 23.2.56.

Med hjälp av en remsa märkt Normalläge II, som finns tillgänglig i en låda invid maskinens remsläsare, kan följande innehåll läsas till trumman:

kanal	000 - IEA	:	noll
"	IEC - IF4	:	A 42
"	IF6	:	C 21
"	IF8	:	C 20
"	IFA	:	C 14
"	IFC	:	C 12
"	IFE	:	C 10

I snabbminnet står efter inläsning av remsan:

000 - 007	:	de permanenta konstanterna
008 - 7FD	:	noll
7FE	:	7FF2C

Maskinen har stoppat på order 7FE: 7FF2C, när remsan är inläst.

Sekvenserna C10 - C21, som används vid inkörning av program, är alla anpassade till utrymmet 7C0 - 7FF.

Vid användning av remsa A42 eller A42A för inläsning av program läses följande innehåll till plats 00E - 013 i snabbminnet (och till kanal IEC, se A42 beskrivning):

```
00E   01348
  F   7C07F   7C0 - 7FF till kanal 1EC
010   00149
  1   7C03B   C10 in
  2   7C00C   hopp till C10
  3   1EA00
```

Om detta innehåll är oförstört kan man alltid efter stopp av maskinen få decimal eller sedecimal utskrift av lämpliga delar av snabbminnet eller trumman eller ämförelse mellan snabbminnet och trumman genom följande åtgärder:

Hoppa manuellt till 00E. Därvid läses 7C0 - 7FF i snabbminnet till kanal 1EA, sekvens C10 hämtas in från kanal 1FE, maskinen hoppar till order 7C0 i C10 och stannar på order 7C1: 7C52E AR: C1000 1FC00. Skall C10 användas (ger sedecimal utskrift av snabbminnet): nollställ AR, sätt adress till första helord. som skall skrivas ut i vänsterhalvan av AR och tryck på start. Skall någon av de övriga sekvenserna användas: tryck på start utan att nollställa AR. Sekvens C12 läses därvid in till 7C0 - 7FF, och man får stopp på 7C1 : 7C52E med AR : C1200 1FA00 (obs. att den aktuella sekvensens nummer står i vänsterhalvan av AR). Skall C12 användas: se nedan. I annat fall tryck på start utan att nollställa AR Därvid läses C14 in med stopp på samma order som förut. I vänsterhalvan av AR står C1400. På detta sätt är alla sekvenserna C10 - C21 tillgängliga i tur och ordning. Efter C21 följer så C10 på nytt (för den händelse man skulle missa någon sekvens).

Innehållet i 7C0 - 7FF från början är tillgängligt för utskrift i kanal 1EA (se ovan) F.ö. är området 000 - 7BF oförstört, innan någon av sekvenserna C10 - C21 använts. Sekvenserna använder ej permanenta konstanter eller arbetsceller i området 000 - 7BF utom C12, C14 och C21 som använder 780 - 7BF för lagring av en nedläst trumkanal.

Bruksanvisningar till programmen i normalläget:

C 10 Sedecimal utskrift av snabbminnet. Adress till första helord i vänsterhalvan av AR, tryck på startknappen.

C 12 Sedecimal utskrift av trumman. Första kanal i vänsterhalvan av AR, tryck på startknappen. Området 780 - 7BF av snabbminnet förstöres vid utskriften.

C 14 Jämförelse mellan snabbminne och trumma. Sätt AR i begynnelseadress i snabbminnet (3 sedecimala siffror), slutadress (3 sedecimala siffror) och adress till första kanal (3 sedecimala siffror). Sista siffran i AR användes ej. Tryck på start. Alla helord i snabbminnet,

som skiljer sig från motsvarande på trumman, skrivs ut föregångna av adress. Området 780 - 7BF av snabbminnet förstöres. Jämförelse mellan detta område och trumman kan ej utföras.

C 20 Decimal utskrift av snabbminnet. Sätt första adress i vänsterhalvan av AR åtföljt av antalet decimaler i position 19, tryck på start-knappen.

C 21 Decimal utskrift av trumman. Första kanal i vänsterhalvan av AR åtföljd av antal decimaler i pos. 19. Tryck på start, varvid maskinen fortsätter till ett stopp på order 7CA : 00130. Om hel kanal önskas utskriven, tryck vidare. Om 1/4, 1/2 eller 3/4 kanal skall skrivas ut, sätt i vänsterhalvan av AR (pos. 11) 010, 020 eller 030 och tryck vidare. Området 780 - 7BF av snabbminnet förstöres vid utskriften.

Omstart. Om en av sekvenserna stoppas under en utskrivning och skall användas för utskrivning från en ny begynnelseadress (kanal), hoppa till 7C0, sätt därefter erforderlig information i AR och tryck på start. Om en annan sekvens skall användas, hoppa till 7C0 och tryck sedan på start, tills önskad sekvens lästs in till kärnminnet.

11. BIBLIOTEKET FÖR STANDARDPROGRAM.

11.1. Allmänt. Under den tid Besk varit i bruk har så småningom ett bibliotek av standardprogram växt fram. I detta kapitel ges en översikt över de mest använda av dessa program. Endast "moderna" program har medtagits.

Fullständiga beskrivningar över standardprogram kan rekvireras från Kodbiblioteket, Matematikmaskinnämndens arbetsgrupp, Box 6131, Stockholm 6.

Standardprogrammen kan indelas i följande grupper:

1. Fullständiga program för lösning av standardproblem.
   Körning av uppdrag för dessa program sker normalt genom MNA.

2. Snabbkodsystem, med vilka ett problem snabbt kan kodas upp om man avstår från extrema krav på utnyttjande av maskinens räknesnabbhet och minneskapacitet.

3. Sekvenser för inläsning av kod.

4. Sekvenser för inläsning av decimalt stansade datamängder.

5. Sekvenser för utskrivning i decimal form av data.

6. Sekvenser för elementära funktioner.

7. Sekvenser för speciella uppgifter.

Sekvenserna i grupp 3 - 7 är avsedda att ingå som delar (undersekvenser) i större program.

För program gjorda utom MNA anges resp. institution eller firma inom parentes sist i programrubriken.

Grupp 1. Fullständiga program för standardproblem.

L 4. Polynomapproximation enligt minsta kvadratmetoden (max. 28:e graden max. 1024 punkter).

O 2. Lösning av enkla reella rötter till algebraiska ekvationer med reella koefficienter.

O 3. Lösning av algebraiska ekvationer med komplexa eller reella koefficienter.

P 2. Lösning av osymmetriska linjära ekvationssystem med ett eller flera högra led. (max. ordn. 30 vid ett högra led).

P 4. Pos. definita linjära ekvationssystem med maximalt 255 obekanta. Koefficienter som är noll lagras ej. (Stiefel-Hestenes metod).

P 10. Symmetriska linjära ekvationssystem. Maximalt 41 obekanta.

Q 1. Invertering av matriser av högst ordningen 29 med Jordans metod.

Q 3. Invertering av matriser och lösning av linjära ekvationssystem (max. ordn. 79).

Q 8. Lösning av överbestämda linjära ekvationssystem enligt minsta kvadratmetoden. Ett eller flera högra led. (max. 29 obekanta).

Q 9. Allmänna egenvärdesproblem: $(A-\lambda B)x = 0$;A symmetrisk, B positiv definit.

Q 10. Egenvärden och egenvektorer till symmetriska matriser (max. ordn. 20).

S 1. Fouriersynteser i två dimensioner (Stockholms Högskola).

S 10. Fourierintegraler (en variabel).

U 4. Tidsserieanalys; produktsummor och korrelationskoefficienter.

Geodetiska program, som gör en så gott som fullständig databehandling från nätvärden till slutresultat. Beräkning av triangelnät, m.m.

Fastighetsteknik: Beräkning av optimala skogsvägnät (Fast 1, Fast 2). Aptering av träd i timmer och massaved (Apt 1).

Grupp 2. Snabbkodsystem.

Flinta. Interpretativt system med en operationslista omfattande flytande aritmetik (packade tal), administrativa order, elementära funktioner, inläsning och utskrift av data samt trumadministration av datamängder.

Alfakod. Autokodsystem med alfabetiskt beteckningssätt (telex) för storheter och operationer. Flytande räkning, dubbel noggrannhet, komplexa tal, trumadministration av program och data, indexsymboler, ett väl utvecklat system för kodning av undersekvenser samt hjälpmedel för felsökning ingår (Firma Autocode).

STAC 3. Autokodsystem med ungefär samma innehåll som Flinta men arbetande enligt en annan princip (SAAB).

Z 1. Interpretativt system för flytande räkning med komplexa tal (opackade tal). Innehåller flytande aritmetiska operationer,

elementära funktioner, inläsning och utskrift.

SAMS. Interpretativt system för flytande räkning med matriser. Innehåller även inläsning och utskrift av matris (SAAB).

R 10. Interpretativt system för flytande räkning. Innehåller aritmetiska operationer samt hopporder. Beskrivning i Kap. 8.

ÅSA. Interpretativt system för flytande räkning (packade eller opackade tal). Innehåller aritmetiska operationer(AB Åtvidabergs Industrier).

FA 5. Översättning av kod skriven med fiktiva (symboliska) adresser till maskinkod. (AB Åtvidabergs Industrier). Beskrivning i Kap. 12.

FADAC. Interpretativt system för räkning med flytande binärpunkt med dubbel noggrannhet. Sekvenser för polynomberäkning, syntetisk division av polynom, kvadratrot, inläsning, tryckning m. m. ingår. (Tel. AB L. M. Ericsson)

För Facit EDB finns autokodsystemet FA 6 (AB Åtvidabergs Industrier).

Grupp 3. Inläsning av kod. (se Kap. 9).

A 42. Inläsning av Besk-kod och kod i "800-form", om trumman används.

A 41. Inläsning av Besk-kod och kod i "800-form", om trumman ej används.

A 40 Kort inläsningssekvens för Besk-kod.

Grupp 4. Inläsning av decimalt stansade talföljder.

	Representation på remsan (decimal)	Representation i maskinen (binär)
900.	Heltal (beskstansning)	Heltal, enhet i pos. 39.
901.	Tal med konstant antal decimaler, som anges på remsan (beskstansning).	Besktal med skf.
902.	Tal med decimalkomma och/eller 10-exponent (beskstansning).	Packade tal eller besktal med skf.
904.	Tal med decimalkomma och/eller tal med 10-exponent. Telexstansning.	Packade tal.

Grupp 5. Tryckning (stansning) i decimal form av tal.

Om ej annat anges utför sekvensen utskrivning av ett tal.

Representation i maskinen (binär)	Representation utanför maskinen (decimal)

910.	Heltal, enhet i pos. 39	Heltal.
911.	Besktal med decimal skalfaktor.	Tal med föreskrivet antal decimaler.
912.	En följd av högst tio tal. F. ö. som 911.	Tal med föreskrivet antal decimaler.
913.	Besktal med binär skf.	Tal med föreskrivet antal decimaler.
914.	En följd av högst fem tal. F. ö. som 913.	Tal med föreskrivet antal decimaler.
916.	Packat eller opackat tal.	Föreskrivet antal decimaler.
917.	Besktal med skf.	Föreskrivet antal signifikanta siffror (om talet ≥ 1, föreskrivet antal decimaler).
918.	Packat tal.	Föreskrivet antal signifikanta siffror jämte 10-exponent.
920.	Packat tal.	På telexremsa antingen: föreskrivet antal decimaler eller: föreskrivet antal signifikanta siffror jämte 10-exponent.

Grupp 6. Elementära funktioner.

a) Argument och resultat besktal.

930. $x^{\frac{1}{2}}$

937. $\ln x$ $(\frac{1}{2} \leq x < 1)$

941. x^y $(x > 0,\ y \geq 0)$

945. $\sin 2\pi x$

946. $\sin 2\pi x$, $\cos 2\pi x$

948. $\frac{1}{2\pi} \arcsin x$

949. $\frac{1}{2\pi} \operatorname{arc\,tg} x$

b) Argument och resultat med föreskrivna binära skalfaktorer.

931. $x^{\frac{1}{2}}$

933. e^x

938. $\ln x$

942. x^y

c) Argument och resultat opackade tal.

932. $x^{\frac{1}{2}}$

934. e^x

939. $\ln x$

943. x^y

d) Argument och resultat packade tal.

947. $\sin x$, $\cos x$, $\operatorname{tg} x$, $\cot x$.
(Flinta innehåller pseudooperationer för $x^{\frac{1}{2}}$, e^x, $\ln x$, $\sin x$, $\cos x$, $\operatorname{arc\,tg} x$.)

e) Argument och resultat komplexa opackade tal.

System Z 1 innehåller pseudooperationer för e^z, lnz, $1/\Gamma(z)$ samt polynomsekvens. Potensfunktioner, t. ex. kvadratrot, kan utföras med hjälp av e^z och ln z.

Grupp 7. Speciella program.

Fullständigare förteckning över dessa program kan erhållas från MNA.

C 2. Decimal eller sedecimal kontrollutskrift av föreskriven del av W-minnet.

C 10. Sedecimal kontrollutskrift av föreskriven del av W-minnet. Självinläsande.

C 12. Sedecimal kontrollutskrift av konsekutiva trumkanaler. Självinläsande.

C 20. Decimal kontrollutskrift av föreskriven del av W-minnet. Självinläsande.

C 21. Decimal kontrollutskrift av konsekutiva trumkanaler. Självinläsande.

D 6. Överföring av block av konsekutiva trumkanaler mellan trumman och W-minnet.

K 2. Kvadratur enligt Simpsons formel med föreskriven noggrannhet och automatisk reglering av steglängden.

L 1. Interpolation av föreskriven ordning mellan ekvidistanta funktionsvärden.

L 2. Linjär interpolation mellan ekvidistanta funktionsvärden (SAAB)

L 3. Kvadratisk interpolation mellan ekvidistanta funktionsvärden enligt Bessels formel (SAAB).

M 3. (Ersätter M 1). Integration av system av ordinära differentialekvationer med begynnelsevillkor. Runge-Kuttas metod av fjärde ordningen.

M 2. D:o, d:o. Automatisk reglering av steglängden (SAAB).

O 4. Lösning av icke-linjära ekvationssystem. Maximalt 30 ekvationer.

P 11 B. Symmetriska linjära ekvationssystem. Maximalt 126 obekanta (SAAB).

Q 1 B. Invertering eller triangulering av matriser av högst ordningen 29 med Jordans metod.

Q 2. Egenvärden och egenvektorer till symmetriska matriser (max. ordn. 20).

Q 3 B. Invertering av matriser av högst ordningen 85.

Q 4. Multiplikation av två matriser varav den ena är lagrad på trumman och den andra tas från remsa.

R 12. Normering av en grupp packade tal till största förekommande karakteristik.

R 13. Normalisering och packning av tal med gemensam skalfaktor.

S 10 B. Beräkning av fourierintegral (endimensionell) med hjälp av funktion lagrad på trumman.

SR 6. Program för flytande räkning med dubbel precision (SAAB).

U 1. Generering av rektangulärfördelade (pseudo-) slumptal.

U 2. Generering av (pseodo-) slumptal med diskret fördelning.

U 3. Generering av (pseudo-) poissonprocess.

11.2. Närmare om handhavandet av programmen i grupp 4 - 6.

Sekvenserna i grupp 4 - 6 har decimala nummer mellan 900 och 990. De är alla skrivna och stansade i 800-form (kap. 7). Remsorna är försedda med längdord och checksumma.

En stor del av numren 900 - 990 är ännu lediga. Numren är fördelade på följande sätt:

900 - 909	Decimal inläsning
910 - 924	Tryckning
925 - 959	Elementära funktioner
960 - 990	Övriga standardkoder

För flera uppgifter finns det två eller tre olika standardprogram beroende på den representation av tal, som används i problemet.

a) man räknar med besktal

b) man räknar med konstanta skalfaktorer (2-potenser) eller

c) man använder flytande räkning.

I nedanstående beskrivningar betyder HP huvudprogrammet. Symbolen n i (modifierade) Wheelerhopp betyder alltid ett jämnt tal. De horisontella strecken markerar gränser mellan helord. (800) betecknar standardsekvensens startorder, som alltid är lika med adressen till det första halvordet. På den plats i Wheelerhoppet, där det står (800) i beskrivningen, måste man för varje särskilt fall sätta in den verkliga startordern.

I kategori b) innehåller de modifierade Wheelerhoppen 2-exponenterna själva i adressposition (vänster eller höger) - icke adresserna till celler, där exponenterna lagras. Exponenterna blir med detta system

utspridda här och var i programmet, vilket kan vara obekvämt, om "omskalning" måste företas eller om skalfaktorerna beräknas av Besk. Systemet har emellertid en del fördelar, och den nämnda nackdelen kan ändå inte undvikas helt, eftersom dylika skalexponenter ofta förekommer som adressdelar i skiftorder. Negativa exponenter representeras i kategori b) med komplement i adressdelen. Om exponenten är exempelvis -3, skriver man FFD.

För flera av standardkoderna har det modifierade Wheelerhoppet ungefär följande utseende, som kanske fordrar en kommentar

```
n      p  10
n+1    n  70
n+2 (800)0C
n+3    W  op
```

Observera att helordet n står i AR vid inhoppet till subsekvensen. Alltså finns där två informationer: parametern p i vänstra adressdelen och adressen n i högra adressdelen.

Observera operationsdelarna i Wheelerhoppen. De är ibland av betydelse. Förutsätt därför aldrig att de saknar betydelse.

11.2.1. Inläsning av decimalt stansade talföljder. 900

Ändamål: En följd av decimalt stansade heltal x_0, x_1, x_2, ... x_q läses in från remsa och lagras i binär form med enheten i pos. 39 i konsekutiva helceller W, W+2, W+4, ... Checksumma.
Man skriver i HP: (modifierat Wheelerhopp)

```
n-1   n-1  50
n     (800) 0C
n+1    W   00
```

Man stansar på remsa: Till varje tal stansar man först absolutbeloppet, därefter stansar man ett D (om talet är positivt) eller ett B (om talet är negativt). Efter sista talet i följden stansar man storheten

$$S = -(x_0 + x_1 + \ldots + x_q)$$

och dess teckenbokstav (D eller B). Allra sist stansar man ett A, som markerar följdens slut. Tecknet F ignoreras av sekvensen. En felstansad siffra kan därför överstansas med ett F, varefter man stansar den riktiga siffran. OBS! Även om man inte är intresserad av summacheck och inte har tid att räkna ut S för hand, måste någonting (åtminstone ett D) stansas på platsen för S, för att sista talet skall bli rätt inläst.

Resultat: Efter inläsningen av talföljden sker återhopp till order n+2. Då står storheten:

$$T = - |x_0 + x_1 + \ldots + x_q + S| \cdot 2^{-39}$$

i AR. Om man vill, kan man utnyttja denna storhet för en summacheck i HP. (Om $T < -1$, så kommer det i AR ett tal, som är kongruent med T modulo 2.)
Totallängd: 82F.

Ändamål: En följd av decimalt stansade tal x_0, x_1, x_2, ... x_q, alla med d decimaler ($1 \leq d \leq 11$), läses från remsa, divideras med en gemensam föreskriven skalfaktor 2^p, sådan att $x_i \cdot 2^{-p}$ är absolut mindre än ett, och lagras i binär form i konsekutiva helceller W, W+2, W+4, ... (Antalet heltalssiffror får vara olika för olika tal.) Checksumma. Totala antalet siffror ≤ 11. 901
Man skriver i HP: (modifierat Wheelerhopp)

n	p	10
n+1	n	70
n+2(800)		0C
n+3	W	31

Man stansar på remsa: Före första talet i följden stansar man antalet decimaler (d) i decimal form och därefter bokstaven D. Se i övrigt beskrivningen av 900. Man stansar inte något decimalkomma. Om $|x_i| < 1$ får man utelämna decimaler i början av talet. Däremot får decimaler i slutet icke utelämnas, ty storheten d anger decimalkommats plats relativt sista siffran i talet.
Resultat: Efter inläsningen av talföljden går kr till n+4. Då står storheten

$$T = - |x_0 + x_1 + \ldots + x_q + S| \cdot 10^d \cdot 2^{-39}$$

i AR. Se i övrigt beskrivningen av 900.
Totallängd: 87F.

Ändamål: En följd av tal x_0, x_1, x_2, ..., x_q med varierande antal heltalssiffror och decimaler läses in från remsa och lagras i konsekutiva helceller eller vänsterhalvceller (högerhalvceller) W, W+2,... efter att ha dividerats med en föreskriven skalfaktor 2^p (alternativ a och b) eller som packade tal (alternativ c och d). Totala antalet siffror ≤ 11. 902

Man skriver i HP:

Alternativ:	a)		b)		c)		d)	
n	p	10	p	10	irr	30	irr	30
n+1	n	70	n	70	n	70	n	70
n+2	(800)	0C	(800)	2C	(800)	0C	(800)	2C
n+3	W	31	W	31	W	31	W	31

Operationsdelen i n+3 skall vara 11 om halvcellslagring önskas.

a) och c). Om inhoppsoperationen är 0C sker inläsning, översättning och lagring i konsekutiva celler ända tills ett A påträffas på remsan. Då sker återhopp till order n+4 i HP. Därvid står - |T| i AR, där T är summan av de tal, som lästs in sedan A senast påträffades på remsan. Vid summabildningen tas ingen hänsyn till decimalkommats plats i talen, utan de betraktas som heltal (med tecken). Vi kallar denna summa för heltalssumma. Den bildas modulo 2^{39} och har enheten i pos. 38. Det tal, som står närmast före A, bör vara en checksumma, närmare bestämt heltalssumma med ombytt tecken av de tal, som lästs in med 902, sedan tecknet A påträffades förra gången. Om inläsningen gjorts rätt, skall T vara noll[+]. Det sista talet blir inte lagrat i kärnminnet. Även om man inte är intresserad av summacheck, måste man därför stansa ett extra tal - det räcker att stansa ett D - mellan det sista talet av intresse och bokstaven A.

b) och d). Om inhoppsoperationen är 2C blir det återhopp till order n+4 i HP efter varje tal, som läses in. Varje inläst tal adderas till T, och - |T| står i AR vid återhoppet. Om ett A läses in, återställs ursprungliga innehållet i den cell, som det sista talet lagrades i.

Anm. T är noll när programmet 902 läses in. Sedan nollställs T endast om bokstaven A förekommer på remsan. Därför kan tal, som lästs in med förfarande b), inkluderas i samma summacheck som tal, som lästs in med förfarande a).

Man stansar på remsa: För varje tal stansar man först heltalssiffrorna sedan bokstaven C (som representerar decimalkommat), sedan decimalerna och sist bokstaven D (om talet är positivt) eller B (om talet är negativt).[++] För ett heltal behöver man inte stansa C. Om ett tal är mindre än ett i absolutvärde, behöver man ej stansa heltalsnolla. Däremot måste man stansa C. Om ett tal är noll, stansar man enbart tecknet D. Sekvensen ignorerar tecknet F (jfr 901). Tal av typen $7{,}1 \cdot 10^{-6}$ stansas E 7C1D 6B. Till checksumman adderas i detta fall de båda talen 71 och -6.

Totallängd: 8BD.

+) Själva undersökningen om T är noll eller ej måste göras i HP.

++) Efter sista talet i gruppen stansar man (eventuellt) checksumma och tecknet A enligt anvisningarna ovan.

Ändamål: En följd av tal stansade på remsa med telexstans läses in och lagras översatta till packade tal (se kap. 8) i konsekutiva helceller i ferritminnet med början på en uppgiven cell W. Checksumma kan bifogas. Efter talföljden stansas ett T eller ordet SLUT. 904

Man skriver i HP:

n	n	50
n+1	(800)	0C
n+2	W	00

Både udda och jämnt n är tillåtna.
Vid uthoppet står antalet tal i följden i AR med enheten i pos. 10.

Man stansar på remsa: Före talföljden stansas bokstavsskift vilket fås genom att man trycker på tangenten "start". Därefter stansas talföljden enligt vanliga konventioner på så sätt att före ett negativt tal skall stå -, före ett positivt ett + eller ett mellanslag eller något annat tecken (ej minustecken). Mellan två tal måste alltid finnas något tecken (t. ex. mellanslag men ej punkt eller komma) som skiljer dem åt. Decimalkomma (decimalpunkt) stansas som komma (punkt). Heltal kan stansas utan komma. Tal av typen $3{,}1 \cdot 10^{-4}$ stansas

3,1 E - 4 eller 3,1 E = -4

(Tecknet = behandlas som mellanslag).
Exempel: talföljden $-3 \cdot 10^{-7}$; 5,1; 32 stansas:

-3 E = -7 + 5,1 + 32 SLUT

eller med uteslutande av alla inte nödvändiga tecken:

-3E-7 5,1 32T

Observera: högsta antalet siffror som får finnas i ett tal är 11 därvid frånräknat nollor i början, t. ex. om talet är ett äkta decimalbråk. Vid exponenttal frånräknas de siffror som ingår i 10-exponenten.

Om summakontroll önskas beräknas summan S av alla ingående tal med hänsyn tagen till decimalkommats plats. Om tal av typen $3{,}1 \cdot 10^{-4}$ ingår skall "taldelen" 3,1 plus "exponenten" -4 uppfattad som heltal adderas till summan. Summan S medtas i talföljden och stansas föregången av ett A eller ordet SUMMA. Observera att tecknet på summan ej får vändas!

Exempel: föregående talföljd stansas med summa:

-3 E = -7 5,1 32 SUMMA 27,1 SLUT

eller i mera kondenserad form

-3E-7 5,1 32A27,1T

Summan föregången av A kan f. ö. placeras in var som helst i talföljden, eventuellt först. Antalet signifikanta siffror i summan är ej begränsat.

Summan blir ej lagrad i minnet. Om summakontrollen inte stämmer är innehållet i AR negativt vid uthoppet. I så fall finns antalet tal på remsan i vhac (95C). I annat fall står antalet tal i följden i AR med enheten i pos. 10. Checksumman är ej medräknad i antalet.

Ett F på remsan medför att närmast föregående tal (ev. endast delvis utstansat) ignoreras. Man kan på detta sätt rätta felstansningar om man upptäcker dem innan man börjat stansa nästa tal.

Följande tecken får användas (se f. ö. telexalfabetet sid. 6.2a):

komma, punkt:	decimalkomma(-punkt).
+, - :	tecken före ett tal (+ före ej nödvändigt).
decimala siffror och bokstaven O:	bokstaven O tolkas som siffran 0.
A :	före summa.
T :	markerar slut.
S, L, U, M, : , = , mellanslag, vagnretur, ny rad :	uppfattas alla som mellanslag och är oväsentliga utom då de tjänar att skilja två tal åt.

Följande stopp kan inträffa under inläsningen:

1) KR: (94A), ASOP: 00170.
Ett inläst tal är för stort (större än c:a $1,7 \cdot 10^{38}$ i absolutbelopp) eller för litet (mindre än c:a $2,94 \cdot 10^{-39}$ i absolutbelopp men ej noll) för att kunna representeras som packat tal i maskinen.

Om man trycker på start och har omställaren "utmatning" i läget "skrivmaskin" får man utskrivet 1 vagnretur och ett E varefter nytt stopp inträffar på KR:(94F) ASOP: (929)24. Om man drar tillbaka remsan till utgångsläget och trycker på start får man inläsning på nytt av talföljden.

2) KR: (84B) ASOP: (94B)6C
2 decimalkomman(-punkter) har lästs in i ett tal eller också har ett tecken som ej finns med i listan över tillåtna tecken ovan lästs in. Om man trycker vidare händer samma som vid 1) men med tryckning av ett C i stället för ett E.

Totallängd: 96F.

11.2.2. Trycksekvenser (Stanssekvenser). Några gemensamma drag för de följande sekvenserna (med undantag av 924):
a) om det beordrade antalet heltalssiffror är större än nödvändigt, görs extra mellanslag tills första signifikanta siffran kommer,
b) före ett positivt tal görs mellanslag, före ett negativt tal trycks minus,
c) decimalpunkt trycks mellan heltalsdel och bråkdel,
d) korrekt avrundning görs,
e) summan av alla tecken, som tryckts med operationen 1C, finns vid återhoppet i AR med enheten i pos. 39 (Undantag: sekvens 920).

Ändamål: Tryckning i decimal form av ett heltal, som är givet i binär form i hexX med enheten i pos. 39. 910

Man skriver i HP: (modifierat Wheelerhopp)

n-1	n-1 50
n	(800) 0C
n+1	X st

s = (maximala) antalet decimala siffror
t = kodbeteckning för typografisk operation som utförs efter tryckningen.

Totallängd: 837.

Ändamål: Tryckning av ett tal $x = \bar{x} \cdot 10^h$ med (maximalt) h heltals- 911
siffror och d decimaler.

Man skriver i HP: (modifierat Wheelerhopp).

n-1	n-1 50
n	(800) OC
n+1	W($\bar{x}$) hd

Totallängd: 849.

Ändamål: Tryckning av en följd med högst tio tal $x_i = \bar{x}_i \cdot 10^{h_i}$ 912
($i = 0,1,2...9$) med (maximalt) h_i heltalssiffror och d_i decimaler. $\bar{x}_i$ förutsättes vara lagrad i helcell W+2i. Efter varje tryckt tal utförs en bestämd typografisk operation.

Man skriver i HP:

n-1	n-1 50
n	(800) OC
n+1	W S_1S_2
n+2	$h_0h_1h_2h_3h_4$
n+3	$h_5h_6h_7h_8h_9$
n+4	$d_0d_1d_2d_3d_4$
n+5	$d_5d_6d_7d_8d_9$

S_2 är kodbeteckning för den typografiska operation, som skall utföras efter varje tryckt tal. Om $S_1 = 0$, tryckes minustecken före negativa tal. Om $S_1 = 8$, tryckes (stansas) teckenbokstäver (B, C, D) i stället för tecken och decimalpunkt för att det skall vara möjligt att läsa in remsan med t. ex. 902.

Om $h_i + d_i = 0$ avslutas tryckningen och kr går till n+6.

Totallängd: 87F.

Ändamål: Tryckning av ett tal x med föreskriven binär skalfaktor 913
p. d decimaler. Antalet heltalssiffror (h) beräknas av sekvensen som det minsta tal, för vilket $10^h \geq 2^p$; $x = \bar{x} \cdot 2^p$.

Man skriver i HP:

n	p 10
n+1	n 70
n+2	(800)0C
n+3	W($\bar{x}$)0d

Totallängd: 871.

Ändamål: Tryckning av en följd med högst fem tal $x_i = \bar{x}_i \cdot 2^{p_i}$, 914
($i = 0,1,2,3,4$) med d_i decimaler. Antalet heltalssiffror h_i beräknas av sekvensen som det minsta icke negativa tal, för vilket $10^{h_i} > 2^{p_i}$.
$\bar{x}_i$ antas vara lagrad i W+2i.

Man skriver i HP:

n-1	n-1 50
n	(800) 0C
n+1	W S_1S_2
n+2	$p_0^1p_0^2p_1^1p_1^2p_2^1$
n+3	$p_2^2p_3^1p_3^2p_4^1p_4^2$
n+4	$d_0d_1d_2d_3d_4$

S_1 och S_2 har samma innebörd som i 912. $p_i^1p_i^2$ är en tvåsiffrig sedecimal representation av exponenten p_i. Om t. ex. $p_i = -7$, så skrivs $p_i^1 = F$, $p_i^2 = 9$. Om $p_i = 18$ (decimalt) så är $p_i^1 = 1$, $p_i^2 = 2$. Tryckningen avbryts om $h_i + d_i = 0$, varvid kr går till n+5.

Totallängd: 8A9.

Ändamål: Tryckning av ett packat eller opackat tal med h heltals- 916
siffror och d decimaler.

Man skriver i HP:

n	X' 68
n+1	n+1 50
n+2	(800) 0C
n+3	X" hd

Är talet packat sätter man 000 i stället för X".
Nollor i början av talet ersättes med mellanslag. Om $x > 10^h$ tryckes talet ändå riktigt, man får då flera än h heltalssiffror.

Totallängd: 885.

Ändamål: Tryckning av ett tal av formen $x = \bar{x} \cdot 2^p$ ($\bar{x}$ besktal , $-2047 \leq p \leq 2047$) med b signifikanta siffror om $|x| \geq 1$ och med b decimaler om $|x| < 1$. $b \leq 11$. 917

Man skriver i HP:

n	p	10
n+1	n	70
n+2	(800)	0C
n+3	W($\bar{x}$)	bt

b = antalet signifikanta siffror = en sedecimal siffra, $b \leq 11$.
t = kodbeteckning för typografisk operation, som utföres efter tryckningen.

Decimalpunkt trycks före eller mellan siffrorna eller om antalet heltalssiffror är precis b, efter talet. Om talet har fler än b heltalssiffror trycks de b första siffrorna och efter talet trycks sedecimalt en understruken siffra som anger antalet utelämnade heltalssiffror.

Totallängd: 857.

Ändamål: Decimal tryckning av ett packat tal x i form av "taldel" u och 10-exponent v, $0{,}1 \leq |u| < 1$. u tryckes med d decimaler. Typografi: $234{,}72 = 0{,}23472 \cdot 10^3$ tryckes i formen 918

23472 03 om d = 5.

Man skriver i HP:

n	W(x)	68	x till MR.
n+1	n+1	50	
n+2	(800)	0C	
n+3	000	0d	

Totallängd: 877.

Ändamål: Decimal stansning av ett packat tal på remsa enligt te- 920
lexalfabetet. Talet stansas med d decimaler och så många heltals-
siffror som behövs eller med h heltalssiffror, d decimaler och 10-
exponent.

Typografi: 234.78 d = 2
eller 2.3478 E 2 h = 1 d = 4

Man skriver i HP:

x till MR (packat tal)

n	n	50	n jämnt eller udda
n+1	(800)	0C	
n+2	PPP	PP	parameterord

1) PPP PP = 0hd B0: vanlig tryckning enligt första alternativet ovan. d = antalet decimaler, h = maximala antalet heltalssiffror som kan förekomma. Om antalet heltalssiffror i det aktuella talet är = S stansas h - S mellanslag före talet (som börjar med tecken = mellanslag eller minustecken). Insignifikanta nollor kommer ej att stansas före. Om S är större än h stansas talet enligt alternativ 2) nedan med de parametrar h och d som står i parameterordet.

2) PPP PP = 0hdB1: stansning med 10-exponent. Först stansas "taldelen" med h heltalssiffror och d decimaler. Därefter kommer ett mellanslag, ett E, exponentens tecken, exponent med 1 eller 2 siffror (ett extra mellanslag efter E om exponenten bara har 1 siffra).

3) PPP PP = 000B2: checksumma stansas, se nedan.

4) PPP PP = 00093 eller 000B3: checksumma nollställes, se nedan.

Sekvens 920 innehåller arbetsceller för lagring av de stansade talens summa. Enligt punkt 4) kan man genom ett särskilt hopp till sekvensen nollställa summan och enligt punkt 3) kan man stansa ut summan varvid summeringscellerna automatiskt blir nollställda efteråt. De är alltid nollställda första gången sekvensen används i programmet. Summan beräknas av sekvensen med hänsyn tagen till decimalpunkter och exponenterna adderade som heltal (jfr beskrivningen av 904). Vid stansning av summan gör sekvensen först vagnretur,

radframmatning stansas 2 ggr, därefter stansas ett A, summa och ett T sedan vagnretur och radframmatning 2 ggr.

Observera: Om summan skall bli rätt stansad får inte något tal stansas med fler än 11 decimaler. Om den stansade remsan skall kunna läsas in igen av sekvens 904 får dessutom inte något tal ha fler än 11 signifikanta siffror, vilket alltid blir uppfyllt om h + d $\leq$ 11.

Totallängd: 973.

924

Ändamål: Hjälp vid inkörning av program. Ger antingen decimal eller sedecimal tryckning av helord i kärnminnet från bbb(jämn) till sss (udda) när som helst under beräkningarna.

Man skriver i HP: a) Sedecimal utskrift önskas (10 sedec. siffror).

n-1	n-1 50	
n	(800) 0C	
n+1	000 10	(konstanten $16 \cdot 2^{-39}$)
n+2	bbb 68	
n+3	sss 00	

b) Decimal utskrift (9 decimaler).

n-1	n-1 50	
n	(800) 0C	
n+1	000 0A	(konstanten $10 \cdot 2^{-39}$)
n+2	800+bbb 68	
n+3	800+sss 00	

I det decimala fallet skall alltså begynnelse- och slutadresserna ökas med 800 (inte med standardprogrammets startorder).
Hoppen till 924 sätts lämpligen in som parenteser, när programmet fullbordats.

Totallängd: 841.

11.2.3. Elementära_funktioner. Sekvenserna bygger i de flesta fall på att man med en funktionalekvations hjälp kan återföra beräkningen av funktionen på dess beräkning i ett begränsat intervall, där en polynomapproximation kan användas[1]. Noggrannheten är i de flesta fall åtminstone elva decimala siffror. För kvadratroten används en iterationsmetod.

1) Jfr Lanczos: Applied Analysis, sid. 457-463.

Kvadratrötter:

Ändamål: Beräkning av $y = x^{\frac{1}{2}}$ x och y besktal. 930

Man gör i HP: x till MR, normalt Wheelerhopp.

Resultat: y i AR.

Totallängd: 81B.

Ändamål: Beräkning av $y = x^{\frac{1}{2}}$. Föreskrivna binära skalfaktorer. 931

$$x = \bar{x} \cdot 2^{p}, \quad y = \bar{y} \cdot 2^{q}$$

Man skriver i HP:

(n-1	W($\bar{x}$) 68)	$\bar{x}$ till MR
n	p	10
n+1	n	70
n+2	(800)	0C
n+3	q	00

Resultat: $\bar{y}$ i AR.

Totallängd: 82D.

Ändamål: Beräkning av $y = x^{\frac{1}{2}}$ för opackade tal. Taldelen för x i hecX, karakteristiken i hec(X+2). 932

Man skriver i HP:

n	X	10
n+1	n	70
n+2	(800)	0C

Resultat: Taldelen för y i AR, karakteristiken i MR.

Totallängd: 829.

Exponentialfunktioner.

Ändamål: Beräkning av $y = e^{x}$. Föreskrivna binära skalfaktorer. 933.1

Ers. sekv. 931. $x = \bar{x} \cdot 2^{p}, \quad y = \bar{y} \cdot 2^{q}$

Man skriver i HP och Resultat: Som 931.

Totallängd: 84F.

Ändamål: Beräkning av $y = e^x$. Opackade tal. 934
Man skriver i HP och Resultat: Som 932.
Totallängd: 84B.

Logaritmfunktioner.
Ändamål: Beräkning av $y = \ln x = {}^e\log x$. $\frac{1}{2} \leq x < 1$. 937
Man skriver i HP: x till MR, normalt Wheelerhopp.
Resultat: y i AR.
Totallängd: 835.

Ändamål: Beräkning av y = lnx. Föreskrivna binära skalfaktorer. 938.1
Ersätter sekvens 938.
Man skriver i HP och Resultat: Som 931.
Totallängd: 871.

Ändamål: Beräkning av y = lnx. Opackade tal. 939
Man skriver i HP och Resultat: Som 932.
Totallängd: 86B.

Allmänna potensfunktioner. 941
Ändamål: Beräkning av $z = x^y$. x och z besktal, y positivt besktal.
Man skriver i HP:

n-1	(W(x) 68)	x till MR
n	W(y) 10	
n+1	n 70	
n+2	(800) 0C	

Resultat: z i AR.
Totallängd: 88B.

Ändamål: Beräkning av $z = x^y$, Ersätter sekvens 942. 942:1

$$x = \bar{x} \cdot 2^p, \quad y = \bar{y} \cdot 2^q, \quad z = \bar{z} \cdot 2^r$$

Man skriver i HP:

(n-1	W($\bar{x}$) 68)	$\bar{x}$ till MR
n	p 10	
n+1	n 70	

```
n+2  (800) 0C
n+3   r    00
n+4  W(ȳ)  00
n+5   q    00
```

Resultat: $\bar{z}$ i AR.
Totallängd: 8B7.

Ändamål: Beräkning av $z = x^y$. x och z opackade, y med binär skalfaktor. $y = \bar{y} \cdot 2^q$. 943

Man skriver i HP:

```
n      X  10
n+1   n  70
n+2 (800)0C
n+3  000 00
n+4  W(ȳ)00
n+5   q  00
```

Resultat: Taldelen av z i AR. karakteristiken i MR.
Totallängd: 8AD.

Trigonometriska funktioner. 945
Ändamål: Beräkning av $y = \sin 2\pi x$. x, y besktal.
Man skriver i HP: x till MR, normalt Wheelerhopp.
Resultat: y i AR.
Totallängd: 839.

Ändamål: Beräkning av $y = \sin 2\pi x$, $z = \cos 2\pi x$. x, y,z besktal. 946

Man skriver i HP:

```
(n      W(x) 68)  x till MR
 n+1   n+1  50
 n+2  (800) 0C
 n+3   W(y) 31
 n+4   W(z) 31
```

Resultat: y och z till W(y) och W(z) som besktal.
Totallängd: 84D.

Ändamål: Beräkning av y = sin x, cos x, tg x eller cot x. x och y packade tal. 947

Man skriver i HP:

n-1	n-1 50
n	(800) 0C
n+1	W(x) s0
n+2	$W(y_1)$ 31
(n+3	$W(y_2)$ 31)

Resultat: Om s = A: sin x packat till $W(y_1)$
" s = B: cos x " " $W(y_1)$
" s = C: sin x " " $W(y_1)$
och cos x " " $W(y_2)$
" s = D: tg x " " $W(y_1)$
" s = E: cot x " " $W(y_1)$
" s = F: tg x " " $W(y_1)$
och cot x " " $W(y_2)$

Obs! Ordern i cell n+3 ingår i modifierade Wheelerhoppet endast om s = C eller F.

Totallängd: 90B.

Ändamål: Beräkning av $y = \frac{1}{2\pi}$ arc sin x. x besktal, $|y| \leq \frac{1}{4}$. 948

Man skriver i HP: x till MR, vanligt Wheelerhopp.

Resultat: y i AR.

Totallängd: 867.

Ändamål: Beräkning av $y = \frac{1}{2\pi}$ arc tg x. x besktal, $|y| \leq \frac{1}{8}$. 949

Man skriver i HP: x till MR, vanligt Wheelerhopp.

Resultat: y i AR.

Totallängd: 847.

12. AUTOMATISK KODNING. SYSTEM FA(5).

12.1. Översättning och interpretation. Vissa delar av kodningsarbetet är ganska rutinbetonade, och tanken ligger nära att överlåta dem till ett mekaniskt hjälpmedel, lämpligen till Besk själv. Man kan då skriva programmet i en pseudokod, som ligger närmare den ursprungliga matematiska formuleringen, och Besk får själv sköta om att omtolka pseudokoden till den önskade beräkningsgången. Omtolkningen kan i maskinen ske bland annat på följande principiellt skilda sätt:

Den första metoden kallas översättningsmetoden. Maskinen översätter i samband med inläsningen pseudokoden till vanlig beskkod, som sedan lagras i minnet. Varje order i pseudokoden svarar mot en eller flera order i beskkoden. Beräkningarna utförs sedan enligt den vanliga beskkoden. Om ett program skall användas många gånger är det lämpligt att ta ut en självinläsande remsa, där programmet är representerat i vanlig beskkod och sedan använda den vid de reguljära körningarna. Detta sätt kallas översättningsmetod.

Den andra metoden kallas den interpretativa metoden. Då används pseudokoden hela tiden. Varje ord i pseudokoden granskas var gång det behövs av en s.k. interpretationssekvens, som omtolkar ordet till en eller flera beskoperationer. De standardsekvenser som kräver modifierade Wheelerhopp kan betraktas som mycket enkla interpretationssekvenser. Pseudokoden utgörs av de halvord, som skrivs i HP efter hoppordern. Se t.ex. trycksekvensen 912. Ett mera utvecklat interpreterande program är R 10 för flytande räkning (se kap. 8).

Interpretativa metoder är i regel betydligt långsammare än översättningsmetoder vid reguljär användning, men de fordrar iblard mindre minne. Vid större amerikanska och engelska maskiner kommer översättningssystem allt mer i bruk. För kodaren är det inte så stor skillnad mellan metoderna men för maskinen är de väsensskilda.

Men hur skall en pseudokod se ut? Det mest radikala vore att koda i en form, som liknar den matematiska formalismen, med uttryck som t.ex.

$$(1 - e^{-2x}) \;/\; \sin x^{\frac{1}{2}} \longrightarrow y$$

(Pilen betyder "ersätter"). Detta kräver att bokstäver och andra symboler kan tolkas av maskinens remsläsare, något som dock numera i viss

utsträckning är möjligt eftersom man kan läsa in telexremsa (se sid. 3.9). Översättningssekvensen måste vara komplicerad och omfattande, ett stort programbibliotek måste ingå. Med betydligt enklare medel kan maskinen emellertid behandla pseudokoder, som i högre grad påminner om den vanliga beskkoden men som ändå besparar kodaren några av de mest irriterande och tidsödande arbetsmomenten.

För Besk finns numera flera olika kodningssystem för pseudokod av detta slag i bruk. Bland dessa kan nämnas

FA-5 Översättning av kod skriven med symboliska adresser (AB Åtvidabergs Industrier).

FLINTA Utbyggnad av det interpretativa systemet R 10 (se kap. 8 sid. 8.3). Operationslistan omfattar flytande aritmetik (packade tal), administrativa order, elementära funktioner, inläsning och utskrift av data samt trumadministration av datamängder.

STAC 3 Autokodsystem av ungefär samma innehåll som FLINTA men arbetande enligt översättningsmetoden (SAAB).

Dessutom finns för FACIT EDB systemet FA-6 (AB Åtvidabergs Industrier).

Under utarbetande befinner sig f.n. (den 28.1.58) systemet Alfakodning som i motsats till de föregående använder sig av alfabetiskt skrivsätt och även i övrigt går längre i fråga om förenklad kodning (Firma Autocode).

För användningen av systemet FLINTA hänvisas till en speciell beskrivning som kan rekvireras från Matematikmaskinnämndens Arbetsgrupp, Kodbiblioteket, Box 6131, Stockholm 6.

Fortsättningen av detta kapitel kommer att ägnas åt begreppet fiktiv (symbolisk) adress och en närmare beskrivning av systemet FA-5.

12.2. Relativa och fiktiva adresser. Det första steget mot automatisering av kodningsarbetet är metoden med relativa adresser, som i flera år har använts av kodare vid Besk. Metoden innebär att kodaren delar in problemet i delprogram och skriver dessa var för sig i 800-form (jfr kap. 7). Anpassningen till verkliga adresser görs av Besk med hjälp av en vanlig inläsningssekvens, som kan anpassa standardprogram, t.ex. A41 eller A42. Etiketterna för inläsningen måste kodaren bestämma. Om han i ett delprogram behöver hänvisa till ett annat delprogram, måste

han antingen räkna ut den verkliga adressen eller också måste han konstruera någon form av modifierat Wheelerhopp. Det kan vara lämpligt att skriva alla delprogram som slutna sekvenser och låta alla övergångar och överföringar av information mellan dem förmedlas av en skelettsekvens (sid. 7.9).

Mekaniseringen är längre driven i den s.k. FA-kodningen, vilket betyder kodning med fiktiva adresser. Som läsaren säkert har märkt redan i den här bokens enkla övningsexempel, måste man vid vanlig kodning ofta hänvisa till ord längre fram i programmet, vars adresser man inte känner. Dessutom finner man ofta under kodningens gång, att man borde ha gjort vissa saker tidigare, t.ex. nollställt någon cell i minnet. Allt detta gör, att det är obekvämt att vara bunden av ordernumreringen redan från början i kodningsarbetet. I regel blir det nödvändigt att införa provisoriska beteckningar för de celler man måste hänvisa till. En sådan provisorisk beteckning skall vi kalla fiktiv adress eller helt enkelt ett namn. När man sedan kommer ner till en order (eller en lagrad konstant), som det tidigare hänvisats till, bör man skriva namnet på kodpapperet vid sidan om ordern. När man kommit fram till sista ordern i koden, kan man gå igenom den och sätta ut adresser till ordercellerna och ev. sätta in blindorder, om det visar sig nödvändigt med hänsyn till fordringar om höger- och vänsterplacering. Sedan måste man gå igenom koden en gång till och ersätta namn med verkliga adresser i adressdelarna. I större program måste man skriva en lista över alla namn, där man i samband med första genomgången av koden antecknar den motsvarande adressen. En sådan lista skall vi kalla lexikon i det följande.

Det här var alltså en skildring av ett möjligt tillvägagångssätt för en systematisk kodare, när han skriver koder, som innehåller mycket hopp och ordermodifikationer. Att sätta in de verkliga adresserna är enkelt men tidsödande och irriterande, eftersom man tycker att man på det stadiet egentligen är färdig med programmet. Dessutom löper man risken att behöva numrera om flera gånger, om man vill göra rättelser eller arbeta in nya uppslag, som man får under kodningsarbetets gång. Men principen är så enkel att Besk kan genomföra den, förutsatt att de fiktiva adresserna (namnen) är sådana tecken, som kan stansas på remsa och läsas av Besk. I nästa avsnitt skall visas hur man använder ett program FA(5), som i huvudsak arbetar på det just beskrivna sättet, när den översätter en FA-kod (kod med fiktiva adresser) till en beskkod med verkliga

adresser. Märk att det problem som motiverade att man införde fiktiva adresser inte helt kan lösas med relativa adresser. Inom varje delprogram är ju de relativa adresserna ordnade på samma sätt som de verkliga adresserna. Ordningen i mängderna av fiktiva adresser och verkliga adresser behöver däremot inte ha något samband med varandra. Detta är den mest betydande skillnaden mellan begreppen relativ och fiktiv adress.

12.3. Kodning med system FA(5). Huvuddragen. Vid MNA utvecklades av G. Hellström i början av år 1956 en sekvens för att översätta en kod med fiktiva adresser till en kod med verkliga adresser. Den har sedan vidareutvecklats vid AB Åtvidabergs Industrier till system FA(5). I förra avsnittet infördes några av de grundläggande termerna: verklig adress, fiktiv adress = namn, FA-kod, beskkod.

Den grundläggande byggstenen i FA-koden är beskrivningen av det, som i den färdiga beskkoden är ett halvord, denna byggsten kallas en FA-order. FA-ordern har en adressdel och en operationsdel. Operationsdelen erbjuder inga speciella problem, den skrivs precis så som den skall vara i beskkoden. För adressdelen finns olika möjligheter:
a) Om adressdelen är ett heltal $N \geqq 100$ med tre decimala siffror tolkas den av FA(5) som den fiktiva adressen (namnet) till den cell, som skall deltaga i operationen.
b) Om man skriver ett B före FA-ordern, tolkas de fem följande sedecimala siffrorna som ett vanligt beskhalvord, adressdelen tolkas som en verklig adress. Om adressdelen är mindre än 100, får B utelämnas.

Till varje namn som förekommer i programmet måste finnas en (och endast en) märkning. Denna märkning består av namnet föregånget av bokstaven A, C eller E. Märkningen skrives före det halvord eller helord (order, konstant eller arbetscell) vars verkliga adress namnet representerar. Bokstaven (A, C eller E) anger:

A: det märkta halvordet skall ha jämn adress (vara vänsterhalvord). En blindorder 000 05 kommer eventuellt att inskjutas omedelbart före i den översatta koden.

C: det märkta halvordet skall ha udda adress (vara högerhalvord). En blindorder 000 05 kommer eventuellt att inskjutas omedelbart före

E: det är likgiltigt om det märkta halvordet har jämn eller udda adress.

Om namnet syftar på ett helord märker man vänsterhalvordet i helordet (A-märkning).

I exemplet nedan står till vänster ett stycke FA-kod, till höger dess utseende efter översättningen varvid antagits att den första ordern hamnar i hac 418:

```
E201   100 70                                418 41C 70
       101 30                                  9 420 30
       102 30                                  A 41E 30
       200 2C                                  B 423 2C
A100 B 400 00                                  C 400 00
       000 00     obs. att B före              D 000 00
                  halvorden får
A102   00C CC     utelämnas eftersom           E 00C CC
       000 00     de börjar på 0.              F 000 00
A101   0CC CC                                420 0CC CC
     B CCC CD                                  1 CCC CD
C200   201 0C                                    2 000 05  blindorder
                                                 3 418 0C
```

En vänsterhalvcell (helcell) får gärna ha ett udda namn och vice versa.

Om ett märkt halvord föregås av B måste märkningen stå före B-et.

Om en arbetscell förekommer i programmet måste den skrivas ut i koden till exempel i form av ett helord eller halvord med innehållet 0. Detta för att man skall kunna införa den märkning som svarar mot ordet. Ett halvord som skall innehålla 0 kan skrivas ut i form av bokstaven D enbart. Ovanstående kod kan alltså också skrivas:

```
E201   100 70
       101 30
       102 30
       200 2C
A100 B 400 00
         D
A102   00C CC
         D
A101   0CC CC
     B CCC CD
C200   201 0C
```

Antalet olika namn får ej överstiga 480.

Ett halvord kan ha flera namn. Märkningar till alla namnen skrivs då ut före halvordet. De måste emellertid alla ha samma märkningsbokstav (A, C eller E).

FA-5 arbetar så att den först läser in en remsa (eller flera), där man stansat FA-order, namn, sedecimala konstanter, sedecimalt skrivna

delprogram (varom mera nedan) i den ordning de uppträder i kodarens manuskript. Efter översättningen kommer det färdiga programmet att lagras dels i ferritminnet med början på en föreskriven adress, dels på trumman från och med en föreskriven kanal.

Observera att startorder till ett program alltid blir = det första halvordet i programmet (som eventuellt kan vara en hopporder till något annat ställe).

Före koden stansar man ett s.k. belägenhetsord, ett helord

bbb 00 ttt 0N

där bbb är den önskade verkliga begynnelseadressen (alltid jämn och minst lika med 024), ttt är den önskade begynnelsekanalen (programmet får inte lagras på någon av kanalerna 100-11C eller 144-1BE, som tas i anspråk av FA-5 under översättningen) samt N är en sedecimal siffra vilken som helst, som blir utskriven under översättningen och kan användas som identifieringssiffra för FA-koden. (Belägenhetsordet kan också sättas manuellt i AR. Se avsnitt 12.5.1.) I slutet på varje remsa stansas signalen ED som utlöser stopp. I slutet på den sista remsan stansas EC som är en signal att beskkoden skall bildas.

Märk att programmet i översatt form i ferritminnet inte får räcka längre än fram till och med hac 7DF.

Då remsorna är inlästa ligger på trumman (kanal 140) en arbetskod, ett helord för varje halvord i den slutliga beskkoden. (Blindorder som eventuellt påtvingats genom krav på lagring i vhac resp. hhac har då också bildats). Varje gång en namnsignal (A, C eller E) inläses, konstrueras ett lexikonord. Det är ett helord, som innehåller både namnet och den verkliga adressen. Lexikonorden lagras i kanal 100 ff. Med hjälp av lexikonet utföres sedan översättningen av de fiktiva adresserna till verkliga adresser.

Exempel på FA-kod. Skriv sedecimalt ut innehållet i det snabba minnet från och med det helord, vars adress sättes i ARV:s adresspositioner i starten, till och med det helord, vars adress + 1 sättes i pos. 23 i AR vid starten.

	FA-kod		Förklaring
	7B000	1C001	Belägenhetsord
	A100	101 2C	Stopp för manuell insättning i AR
	C101	102 07	Beg. adress sättes
		00C 05	12 vänsterskift
		103 07	Slutadress lagras
	E109	104 70	Antal ord per rad sätts: 4 → k
		105 07	
	A103	B103 1D	Vagnretur. Adressdelen skall lagra slutadr.
		102 48	Adress tryckes först i raden
	A106	106 50	Wheelerhopp till sekvens som trycker de i första
		107 0C	sed. siffr. i MR
	E110	005 5D	i = 3 och samtidigt mellanslag
108 4E	C114	000 08	konstant
116 0B	A102	B102 68	Order som skall modifieras
108 0E	A108	108 50	Wheelerhopp till tryckning av ett
000 5C		107 0C	helord i MR
117 0C		014 00	i = 10
	E117	102 46	Nästa helord tryckes
		103 0B	Är alla tryckta?
		100 0E	Hopp om så är fallet
		105 46	Ny rad?
		109 0E	Hopp om så är fallet
		110 0C	
	E107	112 07	Tryck de i första sed. siffr. i MR
		112 46	Adressen till i sättes
		113 07	Uthopp sättes
	A112	B112 4B	Modifierad order
		112 07	- i → n
	A113	B113 0E	Modifierad order. n = 0 uthopp
		114 42	1:a siffran fås genom mult. med 8
	A105	B105 1C	Antal ord per rad i adressdelen
		112 46	n + 1→n
		113 0E	Hopp om n = 0
		115 42	följande siffr. fås genom mult. med 16
		105 0C	
	C115	000 10	konstant 16
	A104	BFF8 00	" -4 i pos. 10
	C116	000 01	"
		ED	stopp för rättelser och tillägg
		EC	bilda koden

Programremsan stansas enligt ovanstående manuskript. Startorder är enligt ovan = den första ordern, 101 2C.

Anmärkningar:

a) När denna kod var färdigskriven men ostansad, fick man idén att endast en nolla bör skrivas om ett helord är 0. Det var alltså nödvändigt att ändra koden. Detta var lätt gjort, man satte bara in de felande

orderna med en klammer (se vid namnet 102 ovan). Därefter kunde man skicka manuset till stansning.
b) Vissa adressdelar modifieras av programmet, bl.a. i den order, som har namnet 103. De kan därför väljas godtyckligt, men det är lämpligt att sätta dem lika med namnet och sätta B före. Anledningen är, att man vid inkörningen av ett FA-kodat program med fördel använder C 14 (se sid. 10.9) för att jämföra koden, som är lagrad på trumman, med koden i kärnminnet efter en stunds räkningar. De helord i kärnminnet, som inte överensstämmer med motsvarande ord på trumman, blir då utskrivna, vilket underlättar identifieringen av fiktiva och verkliga adresser.
c) Hur många blindorder sätter FA(5) in i denna kod?

12.4. Speciella detaljer vid kodning med FA-5:

1) Tecknet F ignoreras om det ingår i ett namn. Ordern

   1F23 2B

   t.ex. tolkas alltså vid inläsningen som 123 2B (adressen 123 = ett namn). Ett halvord eller en märkning kan suddas ut genom överstansning med 5 respektive 4 stycken F.
2) Programdelar skrivna sedecimalt i 800-form kan medtagas i FA-program och blir anpassade. Man stansar i följd:
   a) signalen EE åtföljd av ett namn
   b) längdord: ttt rrr aaa x till 800-programmet (se kap. 7, sid. 5-7 och kap. 9, sid. 3-4).
   c) 800-program
   d) eventuell checksumma (som skall inkludera längdordet).

   Namnet efter EE kommer att syfta på det första halvordet i 800-programmet (vänsterhalvord). 800-programmet kommer att lagras i minnet i direkt anslutning till den kod som står på programremsan före och efter 800-programmet (så när som på eventuella blindorder inlagda av FA-5).

   Om anpassningslängden är 800, anpassas inte något ord, d.v.s. sekvensen uppfattas som en vanlig sedecimal sekvens.

   Om remslängden är 800 kommer inte något program att läsas in. Däremot kommer det att reserveras så mycket plats i minnet som anges av totallängden i längdordet. En användning av detta är följande:

Man önskar en lucka på 128 helordsnollor med början i en cell med namnet 300. Man lägger då ut ett fiktivt 800-program på plats 300 med **rem**slängd 800, anpassningslängd 800 och totallängd 8FF:

EE300 8FF8008000

Anmärkning:
Om adressen i ett halvord i ett 800-program är större än eller lika med E00 kommer den inte att anpassas. Adressen i halvordet DFFFF kommer inte heller att anpassas.

3) Rättelser. Om en längre programremsa redan är stansad, är det obekvämt att behöva kopiera om den för att införa rättelser. Man kan i stället göra rättelser med rättelseremsa som inläses efter programmet.

Antag en rättelse består i att en order i programmet skall ersättas med en hopporder ut till ett tillägg som skall placeras i minnet efter det övriga programmet. Från tillägget skall sedan återhopp ske till ordern närmast efter den där uthoppet skedde.

Antag att uthoppet skall ske YYY halvord, räknat decimalt, efter närmast föregående märkta halvord, som har namnet XXX. I antalet YYY skall halvordet XXX också medräknas. Om den märkta ordern XXX själv skall ändras sättes alltså YYY = 001. Ordern närmast efter uthoppsordern dit återhoppet skall ske skall ges ett namn, ZZZ (ej använt i det övriga programmet).

Rättelseremsan stansas:

```
E A  XXX  YYY  ZZZ     VVV0C
                       uthoppsorder
tillägg: EVVV  ——
               ——
               ——
               ZZZ0C   återhopp
               ED EC
```

Tillägget är en bit vanlig FA-kod som kommer att inläsas som en ren fortsättning av programremsan och lagras efter det övriga programmet i minnet. Rättelseremsan skall inläsas då programremsan stannat vid den sista ED-signalen.

Exempel: I följande redan stansade FA-kod har man glömt att göra två vagnreturer omedelbart före 30150. Man vet att namnen 601-2 är lediga. Det är inte tid att reperforera hela remsan, vilken omfattar ett stort antal FA-order.

	FA-kod och remsa	Förklaring
	0400008001	belägenhetsord
	.	
	.	ett antal FA-order m.m.
	.	
E171	173 50	Den namngivna FA-order som
	000 1C	står närmast före rättelse-
	207 50	stället
	000 1C ————	4:de ordern fr.o.m. 171, här
	301 50	skall två vagnreturer in
	00A 31	
	008 0B	
Ursprung-	018 04	
lig remsa	.	
	.	ett antal FA-order
	.	
	ED	stopp
C999	991 00	sista order (förklaras nedan, avsnitt
	EC	bilda koden 12.6.1.)

Man stansar då följande rättelseremsa:

	Remsa	Förklaring
EA171	004 601 6020C	insättning av hopporder
E602	000 1C	start för parentes (glöm ej den till
	005 50	hopp ändrade ordern!)
	000 1D	2 vagnreturer
	000 1D	
	601 0C	tillbaka till nästa order i huvudsekv.
	ED	stopp
C999	991 00	sista order
	EC	bilda koden

Man läser in huvudremsan med FA(5) och sedan stopp skett vid komb. ED sätter man rättelseremsan i remsläsaren och trycker på startknappen. Då stopp åter sker (vid rättelseremsar ED) har man tillfälle att läsa in ytterligare rättelser. Om koden är färdigläst trycker man på startknappen igen (för EC) och FA(5) börjar då med ledning av arbetskod och lexikon (vilka innefatta information enligt rättelsen) att tillverka motsvarande BESK-kod.

4) Insättning av E2-ettor i alla helord i arbetskoden som innehåller namnetXXX kan göras med signalen

EAA XXX

12.5. Handhavande av programmet FA-5.

12.5.1. Inläsning av FA-remsor.

Först måste program FA-5 läsas in. Sätt en remsa märkt FA(5) med FA(6)AF 6.11.57 i remsläsaren. Sådana remsor förvaras i en låda invid Besks remsläsare. Ställ "utmatning" på "skrivmaskin", tryck på knappen "0→MS" och på "start från remsa". Härvid utför skrivmaskinen en vagnretur och inläsningen börjar. Under inläsningen skrivs indikationen

F⊁A⊁5⊁F

Därpå utföres ytterligare en vagnretur varpå maskinen stannar med KR = = 0B5, ASOP = 0B72C och AR = 8000000000.
Allt är nu klart för inläsning och översättning av FA-remsor.
Man sätter den första remsan med FA-kod som man vill omvandla i remsläsaren. Om den börjar med ett belägenhetsord (se sid.12.6),behöver man endast trycka på "start"; om den däremot ej gör detta, måste man först sätta belägenhetsordet manuellt i AR. Därefter trycker man på "start".

Om allt är riktigt läses nu denna remsa in och eventuellt stopp sker vid varje ED-signal med KR = 11E och ASOP = 0EB2C. Vid varje ED-stopp kontrollerar man om en ny remsa skall sättas in i remsläsaren (då programmet är uppdelat på flera remsor) eller om man kan trycka på startknappen direkt. Då den sista FA-remsan stannar vid sitt sista ED, övertygar man sig om att en EC-signal finns stansad i slutet, varpå man åter trycker på startknappen. Då EC-signalen är inläst börjar maskinen översättningen.

Då översättningen är avslutad göres, om allt är rätt, vagnretur och N⊁skrivs, där N är sista siffran i belägenhetsordet. Om utskrifter vid fel, se nedan avsnitt 12.5.5 och 12.5.6.

Sedan asterisken skrivits ut sker stopp med KR = 2C6, ASOP = 2C72C och AR = 8000000000. Den färdiga koden finns nu lagrad på trumman på kanal angiven i belägenhetsordet.

Två fortsättningar är nu möjliga, nämligen att antingen ta den färdiga koden till inre minnet för att "köra" den eller att stansa ut en beskremsa med den färdiga koden.

1) Färdig kod till inre minnet.

Tryck på "start" utan att nollställa AR. Härvid läses den färdiga koden till sin av belägenhetsordet angivna plats i minnet och stopp

sker med KR = 00A och ASOP = 00B2C. I 00B står hopp till det bildade programmets första adress (enligt belägenhetsordet). Samma hopporder står också i hac 009. Man har nu bara att trycka vidare på "start" för att räkningarna skall börja med den första ordern i programmet.

2) Stansning av beskremsa.

AR nollställes, omkastare "utmatning" ställes på "stans" och därefter trycker man på "start", Härvid stansas först ett stycke blank remsa och därefter en självinläsande version av det översatta programmet avslutad med ett stycke blank remsa. Remsan inledes med en inläsningssekvens, FA(6)AF. Under stansningen av denna sker summacheck. Om denna visar sig misstämma, stannar maskinen med KR:2EE och ASOP=2DBAC. Åtgärd: Tryck på start. Maskinen försöker då en gång till att stansa ut FA(6)AF.

Sedan hela remsan är stansad, stannar maskinen med KR = 324, ASOP = 3332C och AR = 8000000000. Man kan då checka den nyss stansade remsan genom att sätta den i remsläsaren, nollställa AR och trycka på "start". Härvid kontrolleras stansningen av remsan. Om något inte stämmer, sker stopp med KR = 2EE, ASOP = 2DBAC och man kan stansa ut en ny remsa genom att trycka på "start", Om allt stämmer sker stopp med KR = 352, ASOP = 3532C och AR = 8000000000. Man slår nu över omkopplaren "utmatning" till "skrivmaskin". Om man därefter trycker på "start" utan att nollställa AR, så läses den färdiga koden (som ju bör vara identisk med den på remsan) från trumman till inre minnet och stopp sker med KR = 00A, ASOP = 00B2C. I 00B står hopp till det bildade programmets begynnelseadress. Om man å andra sidan vid stoppet KR = 352 nollställer AR innan man trycker på startknappen så kommer, efter det att en vagnretur utförts, stopp att ske med KR = 0B5, ASOP = 0B72C och AR = 8000000000, d.v.s. man har kommit tillbaka till utgångsläget och kan läsa in och "översätta" ytterligare FA-remsor.

12.5.2. Inläsning av beskremsan.

Den självinläsande remsa som FA(5) levererat läser in det översatta programmet till den plats i minnet och på trumman som angavs i belägenhetsordet.

Inläsningen sker genom att man trycker på "0 → MS" och "start från remsa". Remsan är summacheckad. Om checkningen stämmer skrivs under inläsningen vagnretur och tecknen N×F, där N är sista siffran i belägenhetsordet.

När remsan är inläst inträffar stopp med KR = 008, ASOP = 0092C. I 009 står hopp till programmets första order. Förutom programmet står nu i minnet de permanenta konstanterna och FA(5):s trumadministrationssekvens (se avsnitt 12.6.2. nedan).

Man har nu bara att trycka på "start" varvid räkningarna börjar.

Om summachecken skulle misstämma vid inläsningen av den remsa som FA(5) tillverkat, stannar maskinen utan att skriva asterisk efter N-et med KR = 7F5 och ASOP = 7F9AC. I AR står det negativa absolutbeloppet av summan. Programmet har ännu inte förts till trumman. Om man trycker på start sker skrivning av programmet på trumman. Därest man inte önskar föra programmet till trumman startar man på 00F.

12.5.3. Universalstart för FA-5.

Om FA-5 finns på trumman (kanal 1CE - 1EC) kan man komma till utgångsläget för inläsning av FA-kod genom att hoppa manuellt till order 02C. Därvid nedläses FA-5 till minnet och stopp sker efter vagnretur med KR = 0B5, ASOP = 0B72C och AR = 8000000000. Det fordras emellertid att order 02C - 031 i minnet är oförstörda jämte de permanenta konstanterna och trumadministrationssekvensen.

12.5.4. Summacheck vid programavsnitt i 800-form.

I en FA-kod kan ingå summacheckade programdelar i 800-form. Det är sådana där belägenhetsordets sista siffra är 1 och där negativa summan mod 2 av längdinformationsordet och helorden i programdelen följer efter programdelen på remsan (se avsnitt 12.4). Om man inte känner denna checksumma, stansar man en helordsnolla som checksumma. Inläsningen av FA-remsor, som innehåller summacheckade programdelar, utföres om check önskas med E_2-stopp inkopplat. Härvid utskrives för varje sekvens där checken ej stämmer, på E_2-maskinen följande:

```
17C  02AB1  TTTRRRAAA1
17E  17F8C  SSSSSSSSSS
```

Det övre helordet = längdordet till sekvensen, det undre = rätta checksumman om helordsnolla stod efter sekvensen, annars skrivs det tal som skall adderas till checksumman för att den skall stämma.

12.5.5. Felindikationer under inläsning av FA-kod.

1) Om ett A, B, C, D eller E förekommer eller läses i ett namn föregånget av märkningsbokstav, sker stopp med:

KR	ASOP
120	0A120 om A
121	0B120 " B
122	0C120 " C
123	0D120 " D
124	0E120 " E

2) Om efter signalen EA i det därpå följande namnet förekommer någon av siffrorna, B, C, D eller E, sker stopp med

KR	ASOP
1EF	EA120 om B
1F0	EA320 " C
1F1	EA520 " D
1F2	EA720 " E

3) Om det namn som följer omedelbart efter signalen EA på en rättelseremsa inte kan återfinnas i den del av lexikonet, som har konstruerats innan rättelseremsan inlästes, utskrives en indikation på huvudskrivmaskinen, nämligen (om det felaktiga namnet är XXX)

EA XXX

Sedan indikationen utskrivits, fortsätter inläsningen utan stopp.

4) Om signalen EB uppträder, sker stopp med KR:11C och ASOP = 1012C.

Vid varje stopp enligt något av fallen 1), 2) eller 4) är det mest rekommendabelt att

a) anteckna stoppet och göra ett märke (ej blyerts) på remsan,

b) starta med hopp till 0C2.

12.5.6. Felindikationer under översättning av FA-kod.

Sedan EC-signalen på remsan med FA-kod är inläst, börjar översättningen. Därvid göres utskrift om följande fel skulle förekomma i koden:

1) Samma namn förekommer i flera märkningar. Antag namnet 306 förekommer i 2 märkningar och namnet 195 i 3 märkningar medan resten av namnen endast förekommer i en märkning vardera. Då skrivs på skrivmaskinen indikationerna:

306
195
195

×

Detta kontrolleras av FA-5 innan den egentliga översättningen börjar. FA-5 fortsätter sedan med den egentliga översättningen utan stopp. Därvid kommer FA-5 att stryka överflödiga förekomster av ett namn på så sätt att hänsyn tas endast till den märkning som står tidigast i koden. Samtidigt med översättningen kontrolleras om:

2) En order har en adressdel som varken kan tolkas som ett namn eller en sedecimal adress. Exempel: ordern 7C03F utan B före, ordern 99920 om "namnet" 999 inte har någon märkning i koden.

Då utskrives:

Plats i minnet:	Order:	Förklaring:
0B0	8203F	"decimal motsvarighet till 7C0" = 820
35B	99920	999 återfinnes ej bland märkningarna

I den färdiga koden kommer dessa felaktiga order att stå i den form utskriften anger (sedecimalt).

12.6. Praktiska råd till FA-kodare.

12.6.1. Förutsättning: programmet ryms i sin helhet i ferritminnet.

Beträffande större program, se avsnitt 12.6.2.

1) Börja FA-koden med ordern

nnn 0C

där nnn är namnet på programmets egentliga startorder. Lägg efter denna order ut konstanter och arbetsceller. Man bör referera till dem med hjälp av namn (ej sedecimala adresser) för den händelse man skulle behöva skjuta programmet fram eller tillbaka i minnet. Att dessa celler med märkningar bör läggas i början beror dels på att FA(5) då arbetar fortare vid översättningen, dels på att man då lättare kan beräkna deras sedecimala adresser, vilket kan vara fördelaktigt om man vill bygga ut programmet så att det kommer att bestå av flera programdelar som lagras på trumman, avlöser varandra i ferritminnet och har ett antal arbetsceller i ferritminnet gemensamma. Man bör under kodningen skriva ut konstanter och arbetsceller på ett särskilt blad allteftersom de behövs.

2) Om man vill ha reda på den verkliga adressen till ett ord i programmet skriver man invid detta en felaktig order, d.v.s. en order med en adressdel som är ett namn utan motsvarande märkning. Man reserverar

namnet 991 för denna roll (991 får således icke förekomma som namn). Speciellt är man ofta intresserad av att veta var programmet slutar i minnet. Varje remsa som kan avsluta FA-koden avslutas därför lämpligen med:

ED	stopp
99100	sista order
EC	bilda koden

Observera, att om rättelseremsor finns skall detta också stå efter den sista rättelsen!

Har man satt in en "felaktig" order på detta sätt i slutet på ett program så utskrives på skrivmaskinen under översättningen följande:

	Förklaring
✳	alla namn unika
329 99100	den "felaktiga" ordern står på plats 329
N ✳	N = sista siffran i belägenhetsordet.

12.6.2. Åtgärder då programmet inte ryms i ferritminnet.

Om ett program är så långt att det inte ryms på plats 024 - 7DF i minnet, måste trumman användas för lagring av program. Programmet uppdelas därvid på lämpligt sätt i ett antal programdelar. Var och en av dessa skall vara så kort att den ryms i sin helhet i ferritminnet. Man låter FA-5 tillverka en remsa för var och en av dessa programdelar som är kodade formellt som självständiga FA-5-program. Observera att program inte kan läggas direkt av FA-5 på kanal 100 - 11C eller 144 - 1BE, som är upptagna av lexikon resp. arbetskod under översättningen. Däremot kan dessa kanaler naturligtvis disponeras under körningen av programmet t.ex. för lagring av data.

Under räkningarna står endast en eller ett par av programdelarna inne i minnet åt gången, nämligen den eller de i vilka maskinen arbetar för tillfället. Då maskinen behöver nytt program i ferritminnet nedläses den nya programdelen med hjälp av en hjälpsekvens för läsning (eller skrivning) av block från (eller i) trumminnet som ingår i FA-5 och som något avviker från den som ingår i A42.

Sekvensen står kvar i ferritminnet jämte de permanenta konstanterna då FA-5 bildat beskkoden och kan användas under räkningarna. Den finns också med på de självinläsande programremsor som FA-5 producerar.

Utrymme i inre minnet: 010 - 023.

Handhavande: Om man önskar läsa ett block som ligger på trumman med början på kanal ttt till ferritminnet området från och med adress bbb (jämn) till och med adress sss (udda) skriver man i huvudprogrammet

n (jämn)	n 50	
n+1	010 0C	
n+2	bbb 3B	obs. operationsdelen
n+3	sss 00	
n+4	hhh 0C	uthopp till hhh efteråt
n+5	ttt 00	första kanal

Här är hhh den adress i huvudprogrammet dit man önskar hoppa sedan trumläsningen är klar. Sekvensen fungerar även om order n - (n+5) skulle övertäckas vid trumläsningen. Däremot får trumläsningen inte beröra området 000 - 023. Observera att om området bbb - sss inte omfattar ett helt antal kanallängder kommer ett visst antal ord efter plats sss att ersättas med den sista kanalens innehåll.

Om man tvärtom önskar skriva blocket på trumman så skriver man i huvudprogrammet:

n (jämn)	n 50	
n+1	010 0C	
n+2	bbb 3F	obs. operationsdelen
n+3	sss 00	
n+4	hhh 0C	uthopp till hhh efteråt
n+5	ttt 00	första kanal

Vilket område som helst i minnet kan skrivas på trumman på detta sätt. Observera att om området bbb - sss inte omfattar ett helt antal kanallängder kommer sista kanalen att fyllas ut med det som står efter sss i ferritminnet.

Det är praktiskt att inleda varje programdel med orderna

A bbb	bbb 50	
	010 0C	
	bbb 3B	
	sss 00	
	hhh 0C	
B	ttt 00	
	_____	arbetsceller och
	_____	konstanter med
	_____	tillhörande märkningar
E hhh	.....	egentlig startorder.

bbb, sss och hhh är namn. De står för respektive första adress (jämn), sista adress (udda) och egentlig startorder till programdelen. ttt är första kanal på trumman där programdelen är lagrad.

Man ordnar i så fall hoppen mellan programdelarna t.ex. på så sätt att man permanent i ferritminnet har en styrsekvens enligt följande exempel:

kr	order	
024	027 48	⟵ till nedläsning av programdel 1.
025	bbb 3B	
026	bbb 0C	
027	ttt 00	första kanal programdel 1
028	02B 48	⟵ till nedläsning av programdel 2
029	ccc 3B	
02A	ccc 0C	
02B	uuu 00	första kanal programdel 2

o.s.v.

Här står bbb och ccc för första adress i programdel 1 och 2 i sedecimal form, ttt och uuu är adresser till första kanal i programdelarna.

Styrsekvensen utgör en särskild programdel. Den kan kombineras med övrig information som skall stå permanent i ferritminnet.
Den kan naturligtvis läggas på annan plats än 024. Styrsekvensen läses in sist, efter de övriga programdelsremsorna. Den kan antingen kodas sedecimalt och läsas in med någon av de vanliga inläsningssekvenserna eller också tillverkas med FA-5.

När programdelsremsorna inklusive styrsekvensen är inlästa skall starten ske på order 024 i styrsekvensen. Man får då nedläsning av första kanal i programdel 1 och hopp till de inledande orderna som medför att hela programdelen blir nedläst och att maskinen hoppar till programdelens startorder.

Om man under räkningarna vill ha hopp till en annan programdel skriver man t.ex. 028 0C (sedecimal adress) om man vill komma över till programdel 2. Om man avbrutit en räkning och vill starta om från början hoppar man manuellt till order 024.

För att tillmötesgå mer komplicerade fordringar kan naturligtvis styrsekvensen utformas annorlunda.

Varje delremsa som tillverkats av FA(5) innehåller först en inläsningssekvens, FA(6)AF. Därefter följer ett stycke blank remsa och 2 etiketter

bbb39sss0N
80000ttt00

Här är bbb begynnelseadress (jämn), sss slutadress (udda) till programdelen i ferritminnet, siffran N är sista siffran i belägenhetsordet och ttt är första kanal till programdelen på trumman.

Därefter följer programmet och sist kommer helorden

SSSSS	SSSSS	checksumma till programmet
0070C	0000F	
0092C	bbb0C	

Remsan avslutas med blank remsa.

När programmet är färdigt och inkört är det ofta obekvämt att ha flera programremsor.

Man kan i så fall framställa en remsa som innehåller hela programmet och inläses utan stopp om man kopierar ihop alla remsorna på så sätt att FA(6)AF uteslutes från alla remsorna utom den första och de 2 sista orden på varje remsa uteslutes utom på den sista remsan som innehåller styrsekvensen. Denna sista remsa bör i detta fall också vara tillverkad av FA(5) eller i varje fall stansad enligt ovan. Styrsekvensen blir därvid också inläst till trumman, vilket i allmänhet inte bör medföra någon olägenhet.

13. UTFÖRLIGARE TEKNISK BESKRIVNING AV BESK.

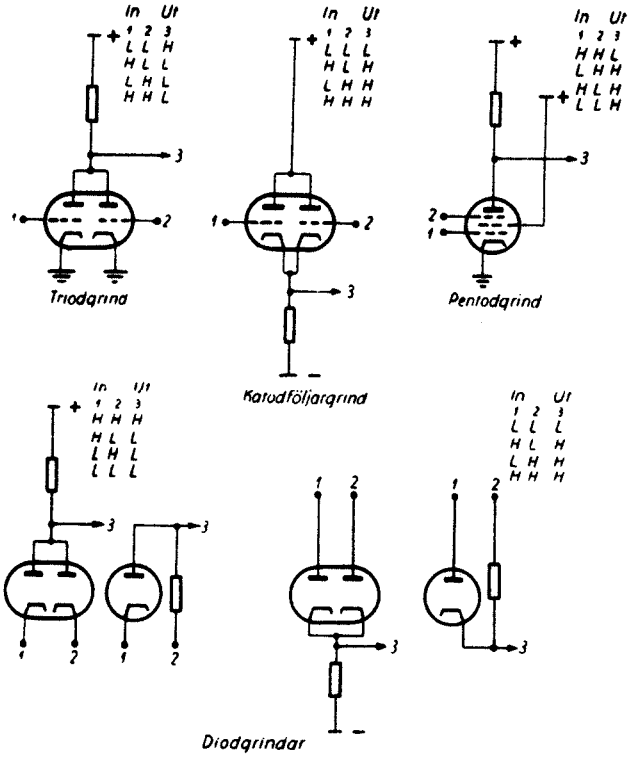
Följande beskrivning av de olika enheterna i Besk är i huvudsak hämtad ur en artikel i Teknisk Tidskrift för den 29 mars 1955, författad av Besks chefskonstruktör civilingenjör Erik Stemme.

Aritmetiska enheten. Den aritmetiska enheten är uppbyggd av två fundamentala typer av element, nämligen grindar och minneselement. Allmänt kan sägas att de förra har till uppgift att i rummet fördela informationen inom maskinen medan de senare har till uppgift att ordna informationen i tidsföljd. Det finns dock icke någon skarp gräns mellan dessa båda element. Många typer av grindar visar sig vid närmare analys innehålla minneselement, t.ex. i form av strökapacitans, medan de flesta minneselement fordrar grindar för att över huvud fungera.

De vanligaste typerna av grindar i Besk är triodgrind, katodföljargrind, pentodgrind och diodgrind (fig. 10). Karakteristiskt för dessa är att de på ett bestämt sätt kombinerar information från de båda ingångarna så att man i ett men endast ett fall av de fyra möjliga kombinationerna får hög eller låg spänning på en utgång.

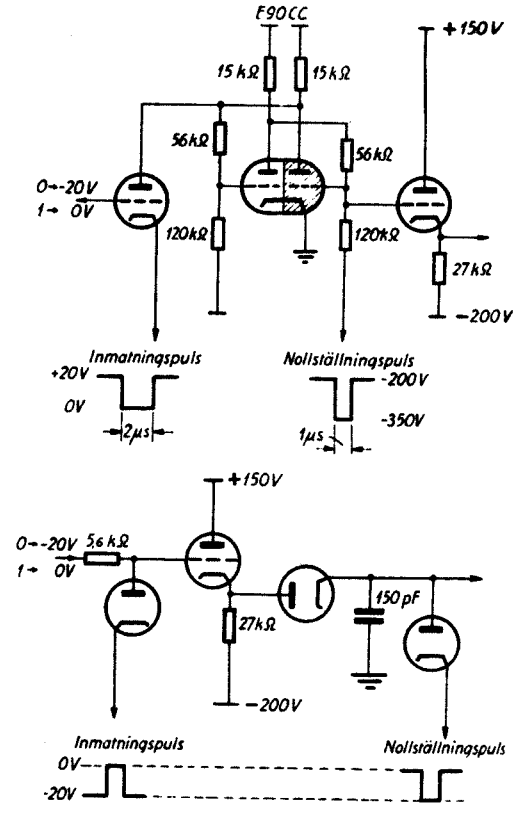
Den aritmetiska enheten innehåller tre register för vardera 40 binära siffror. Som minneselement används bistabila vippkopplingar, "flipflops", fig. 11 upptill. Dessa består av symmetriskt kopplade dubbeltrioder, som kan kantras i två stabila lägen. I den aritmetiska enheten representeras siffrorna genom "höga" resp. "låga" spänningar, där en hög spänning - en etta - motsvarar 0 V och en låg spänning - en nolla - ungefär - 20V. Inmatning av information till en vippkoppling sker först genom att denna ställs i nolläge med en nollställningspuls av 1 /u s varaktighet. Därefter sker inmatning genom att det till minneselementet hörande inmatningsrörets katod pulseras från +20 V till 0 V. Om den inkommande informationen utgörs av en etta (hög spänning på gallret) blir inmatningsröret strömförande och rörvippan kantras över till sitt ettläge. För att få låg utgångsimpedans från minneselementet följs detta av en katodföljare. Varje register är uppbyggt av 40 enheter av detta slag motsvarande 40 siffror.

För skift fordras ett extra minneselement för att ej informationen i två närliggande vippkopplingar skall kollidera. Man måste till detta hjälpelement under ett kort ögonblick överföra vippkopplingens information innan denna går vidare till nästföljande registerelement. Härtill används ett kondensatorminne, fig. 11 nedtill. I detta mottas informationen över en grind och en uppladd-



Olika typer av grindar. Tabellerna ("sanningstabellerna") vid sidan av resp. kopplingar anger; H hög eller L låg utspänning som resultat av H hög eller L låg spänning på de båda ingångarna.

Fig. 10



Minneselement; upptill vippkoppling, nedtill kondensatorminne.

Fig. 11

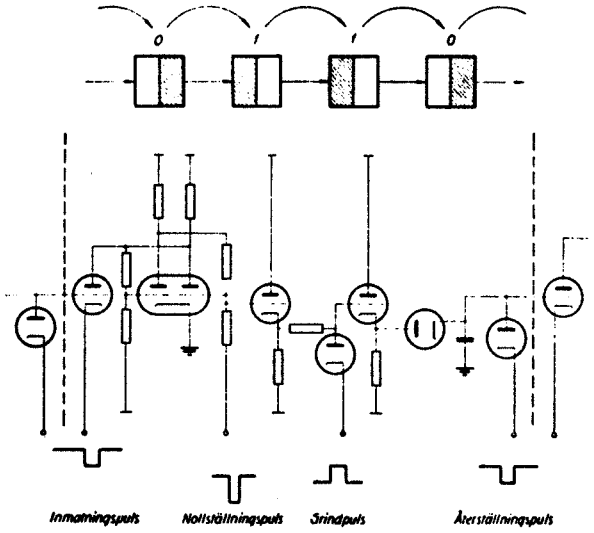
ningsdiod till en kondensator på 150 pF. Denna ligger som resultat av en tidigare återställningspuls på låg spänning. Om man har en nolla i det matande minneselementet så ändras ingenting vid inmatningen under det att en etta resulterar i en uppladdning av kondensatorn till hög spänning. Alla vippkopplingar nollställs därefter varefter kondensatorernas informationer överförs till registerelementen på beskrivet sätt, fig. 12.

Två av den aritmetiska enhetens register, nämligen ackumulatorregistret och multiplikatorregistret, kan utföra skiftoperationer. Det tredje registret, multiplikandregistret, tjänar som uppsamlingsställe för tal, som kommer från kärnminnet, se blockschema, fig. 4.

Vid början av en addition finns det ena av talen i ackumulatorregistret, det andra finns lagrat i kärnminnet. Det senare talet överförs till multiplikandregistret och i vart och ett av de 40 identiska stegen utförs addition i en enhet kallad adder. Denna har tre ingångar förutom från de två registren, även en överföring från näst föregående steg. Addern har också två utgångar, dels en resultatutgång som överför summan till ackumulatorregistret, dels en utgång för carry till nästföljande steg (jfr kap. 2).

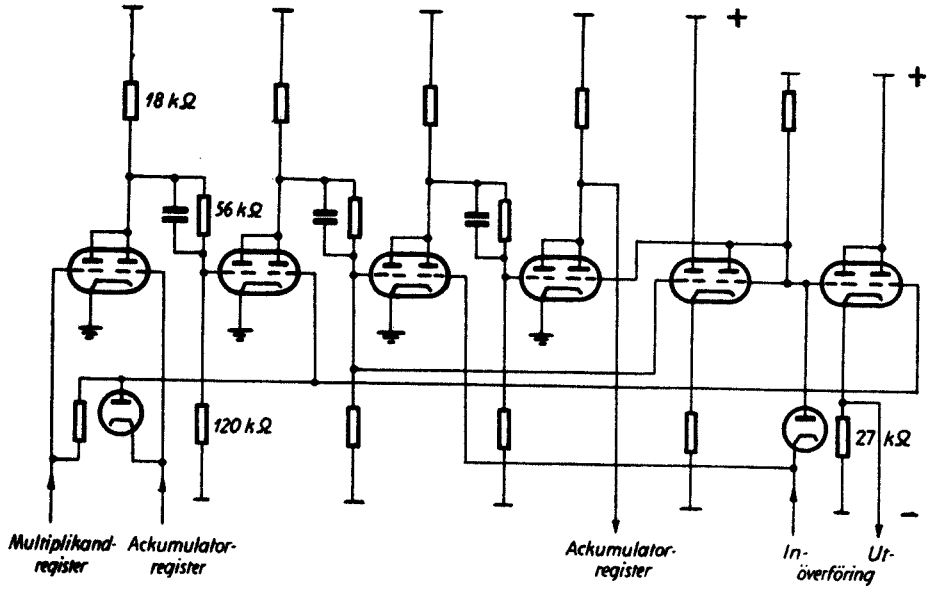
Addern särskiljer de åtta möjliga additionsfall, som uppstår vid addition av tre binära siffror, se tabell 1. Om samtliga ingångsspänningar är höga (ettor) eller om en men endast en ingångsspänning är hög skall addern som resultat ge hög spänning ut. Vidare skall addern lämna hög spänning för utgående carry om två eller flera av ingångsspänningarna är höga. För att särskilja dessa fall fordras ett system av grindar. Adderkretsen i Besk består av sex dubbeltrioder kopplade som triod- eller katodförljargrindar samt två dioder, fig. 13.

Parallelldriften i maskinen innebär, att additionen sker samtidigt i samtliga 40 positioner. Man måste emellertid i varje position ta hänsyn till carry från föregående position. Detta betyder, att additionen i själva verket utförs i serie. Om man t.ex. adderar två tal, där det ena består av idel ettor och det andra har etta endast i sista positionen får man en carry, som fortplantar sig hela vägen från den lägsta positionen till den högsta. Additionens snabbhet kommer därför i stort sett att bestämmas av den hastighet med vilken denna serieöverföring sker. I Besk har tiden för carry nedbringats till 1 µs. Totalt tar emellertid hela operationen inberäknat uthämtning av instruktion och tal 56 µs. Resultatet av additionen lagras i ackumulatorregistret.



Skiftregister.

Fig. 12



Logisk adder.

Fig. 13

Tabell 1. Resultat och överföring från åtta möjliga additionsfall.

Multiplikandregister	0	1	0	0	1	1	0	1
Ackumulatorregister	0	0	1	0	1	0	1	1
Carry in	0	0	0	1	0	1	1	1
Resultat	0	1	1	1	0	0	0	1
Carry ut	0	0	0	0	1	1	1	1

Subtraktion sker genom att komplementet till det tal, som lagrats i multiplikanregistret, adderas till talet i ackumulatorregistret. Komplementbildningen, som sker i en grind inbyggd i multiplikanregistret, innebär, att alla nollor i ett tal utbyts mot ettor och alla ettor utbyts mot nollor, varefter en etta adderas till lägsta positionen. Detta händer i samband med överföringen av multiplikandregistrets innehåll till addern, varför multiplikandregistret själv aldrig kommer att innehålla komplementet till det tal, som hämtades från minnet.

Hur multiplikation och division sker, beskrevs utförligt i kap. 4. Tiden för en multiplikation är 8 1/3 additionstider (A), därav för skift och addition 6 2/3A. Resten av tiden åtgår för eventuell korrektion med hänsyn till tecknen samt för uthämtning av instruktion och multiplikand. Divisionen (operation 12) tar också 8 1/3A, men därtill kommer 7 2/3A för vändning av kvoten (operation 13), som kommer bakvänd i multiplikatorregistret. Multiplikation och division styrs av inbyggda pulsgivare i styrorganet. Tiden för skift beror på antalet steg enligt följande: 0-3 steg tar 1 2/3A, 4-7 tar 2 1/3A, o.s.v., alltså en ökning med 2/3 A för varje påbörjad grupp om fyra steg.

Styrorganet. Samtliga operatipner inom maskinen initieras och styrs genom en följd av pulser från kontrollenheten. Denna innehåller två register, adressregister och operationsregister som vid början av varje operation mottar en instruktion i parallell form från kärnminnet. Adressregistret står även i dubbelriktad förbindelse med kontrollregistret, som lagrar den aktuella instruktionens nummer. Med undantag av adressregistret, som är byggt som en binärräknare, är kontrollenhetens register i princip konstruerade lika som den aritmetiska enhetens.

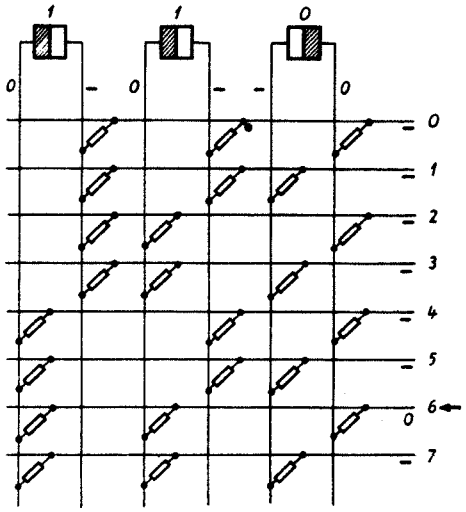
Under varje operation, som omfattar en additionstid, genomlöper styrorganet sex tempon (T_1, T_4, T_5-T_8), som vardera har en varaktighet av 7 μ s. Under T_1 utväljer adressregistret på sätt som närmare beskrivs i nästa

avsnitt en ny instruktion från kärnminnet. Denna fördelas under T_1 på operationsregistret och under T_4 på adressregistret. Under T_4 sker nollställning av AR, om instruktionen beordrar sådan. Den egentliga operationen utförs i regel under T_5 - T_8. Så sker t.ex. överföring av ett tal från minnet till multiplikandregistret eller av ett tal från ackumulatorregistret till minnet under T_5, varvid minnesplatsens nummer tas från adressregistret. Under T_6 överför därjämte kontrollregistret sitt innehåll till adressregistret. Till detta adderas en etta under T_7, varvid numret för den instruktion, som står i tur att utföras erhålls. Slutligen överförs under T_8 adressregistrets innehåll till kontrollregistret. Eftersom numret för nästa instruktion står kvar i adressregistret efter detta tempo, som är det sista i cykeln, är allt klart för nästa operation. Ett undantag från detta tidsschema utgör hoppinstruktionerna. Adressdelen i en sådan instruktion anger den minnesplats, där nästföljande instruktion lagras. Vid hopp uteslutes därför överföringen från kontrollregistret till adressregistret samt framstegningen av adressregistret och endast överföringen under T_8 utförs.

Operationsregistret lagrar instruktionens operationsdel, som omfattar åtta binära siffror. Fem av dessa, motsvarande 32 grundoperationer, dekodifieras i en motståndsmatris. Detta innebär i princip, att det binära talet omvandlas till en viss spänningsnivå på en av matrisens utgångar, fig. 14. De resterande tre siffrorna används för att tillsammans med de 32 grundoperationerna bilda varianter av dessa.

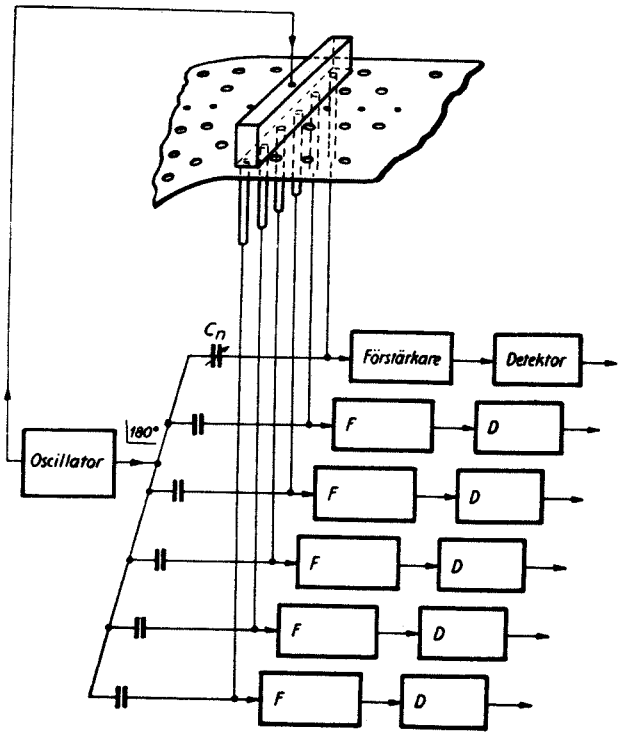
Motståndsmatrisen lämnar statiska spänningar till ett stort system av grindar och pulsförstärkare, som utsänder styrpulser till samtliga enheter i maskinen. Pulsernas fördelning i tiden bestäms i första hand av ett antal tidgeneratorer, som alstrar pulser motsvarande de nämnda tiderna T_1 - T_8. Dessa "T-pulser" har en varaktighet av 7 /u s, vilket ger en normal operationstid på 42 /u s. Vid operationer, som kräver längre tid t.ex. multiplikation och division alstras med hjälp av en särskild binärräknare en puls med en varaktighet motsvarande 39 T-pulser.

De utgående pulsernas längd och läge inom de av tidgeneratorerna definierade tidsintervallen fastläggs av ett antal centrala pulsgeneratorer. Sålunda finns för de pulser, som fordras i den aritmetiska enhetens register (grindpuls, nollställningspuls, inmatningspuls och återställningspuls), fyra pulsgeneratorer.



Motståndsmatris. Det tresiffriga binära talet 110 ger jordpotential på utgång 6 medan de övriga utgångarna får negativ potential.

Fig. 14



Principschema för dielektrisk remsläsare.

Fig. 15

Såväl tidgeneratorerna som de centrala pulsgeneratorerna styrs av en klockpulsgenerator med frekvensen 143 kHz. Vid sådana operationer som berör någon av de yttre enheterna remsläsare, trumminne eller elektrisk skrivmaskin, som går asynkront i förhållande till maskinen i övrigt, bortkopplas klockpulsgeneratorn genom en elektronisk omkopplare och i stället ansluts någon av de klockpulsgeneratorer, som finns i de yttre enheterna. När en sådan operation är färdig utsänder enheten i fråga en klarsignal till kontrollenheten, varvid den inre klockpulsgeneratorn återigen inkopplas.

Se vidare mikrooperationsschemat i slutet av detta kapitel.

Minnet. I en universell elektronisk siffermaskin fordras en minneskapacitet av storleksordningen $5 \cdot 10^5$ binära siffror, varför det av praktiska skäl är uteslutet att enbart använda minneselement som kräver en eller flera rör per binär siffra. I Besk används därför dels ett snabbt magnetiskt minne (kärnminne, ferritminne) med plats för 1024 fyrtiosiffriga tal, dels ett magnetiskt trumminne med åttafaldig kapacitet.

Kärnminnet. (Detta avsnitt är ett utdrag ur en rapport, författad av C.-I. Bergman. Inkonsekvensen i figurnumreringen är en följd av att originalets figurnummer bibehållits.)
Minneselementet i ett magnetiskt kärnminne eller ferritminne utgöres av en toroid av ett speciellt ferritmaterial med en ytterdiameter, som vanligen ligger omkring 2 mm. Utmärkande för detta ferritmaterial är att dess magnetiseringskurva har en utpräglad rektangulär form (fig. 1). Om kärnan magnetiseras med en strömimpuls +im får man en stark kvarstående remanens $+\Phi_r$ i ringen. Strömimpulsen -im ger remanensen $-\Phi_r$. Dessa två magnetiseringstillstånd kan definiera de binära siffrorna 0 och 1. Antag att kärnan befinner sig i läge 1 och att en strömimpuls +im med lämplig stigtid och varaktighet passerar ledaren genom kärnan. Samtidigt som magnetiseringstillståndet ändras från $-\Phi_r$ till $+\Phi_r$ induceras i ledaren en spänningsstöt. Om kärnan från början i stället hade befunnit sig i 0-läge, skulle strömimpulsen im ha orsakat en spänningsstöt av kortare varaktighet och med avsevärt lägre amplitud (fig. 2).

Som framgår av fig. 1 ändras magnetiseringen hos en kärna som befinner sig i 1-läge endast i ringa grad om den pulseras med strömstyrkan + $\frac{1}{2}$ im. Magnetiseringskurvans höga rektangularitet erbjuder en elegant metod att utvälja ett önskat minneselement för läsning eller skrivning. Fig. 3 visar hur detta kan göras. Kärnorna sitter upptrådda på ett antal mot varandra vinkelräta puls-

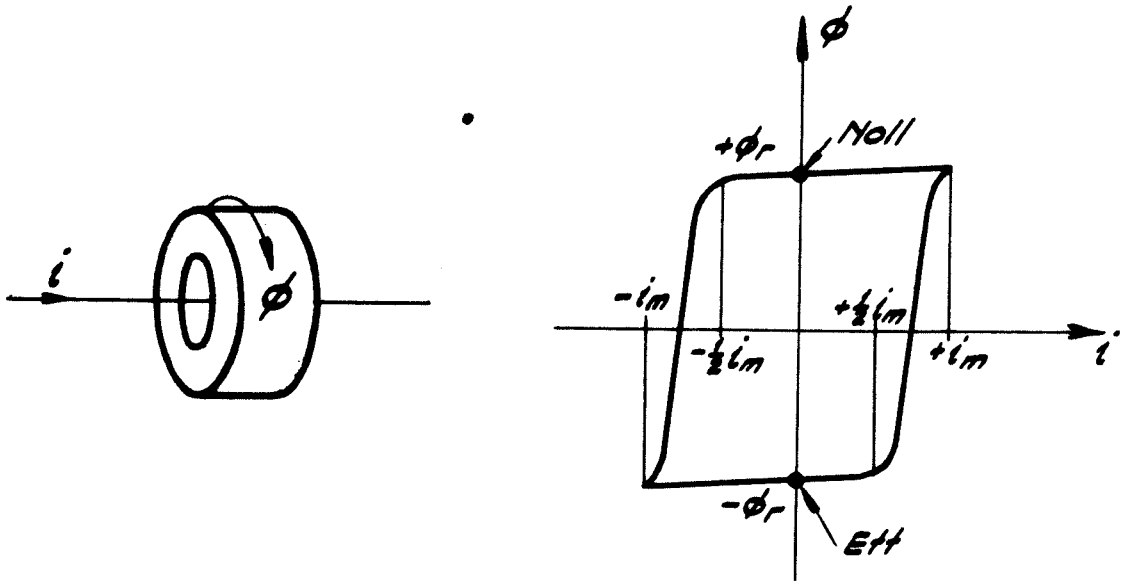


Fig. 1

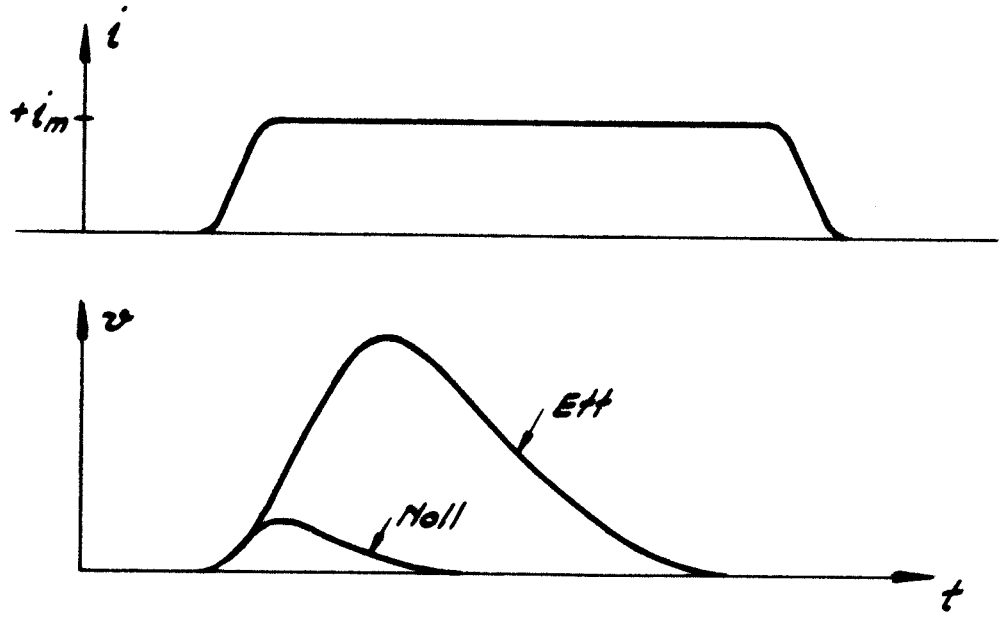


Fig. 2

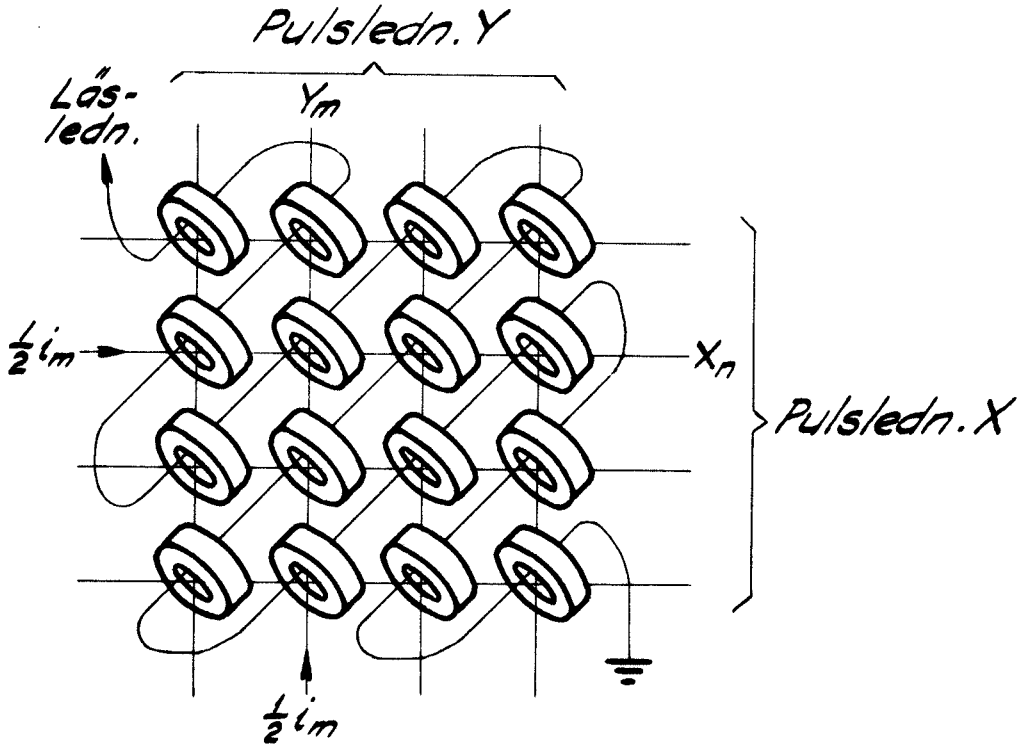

Pulsledn. Y
Läs-
ledn.
Y_m
$\frac{1}{2} i_m$
X_n
Pulsledn. X
$\frac{1}{2} i_m$

Fig. 3

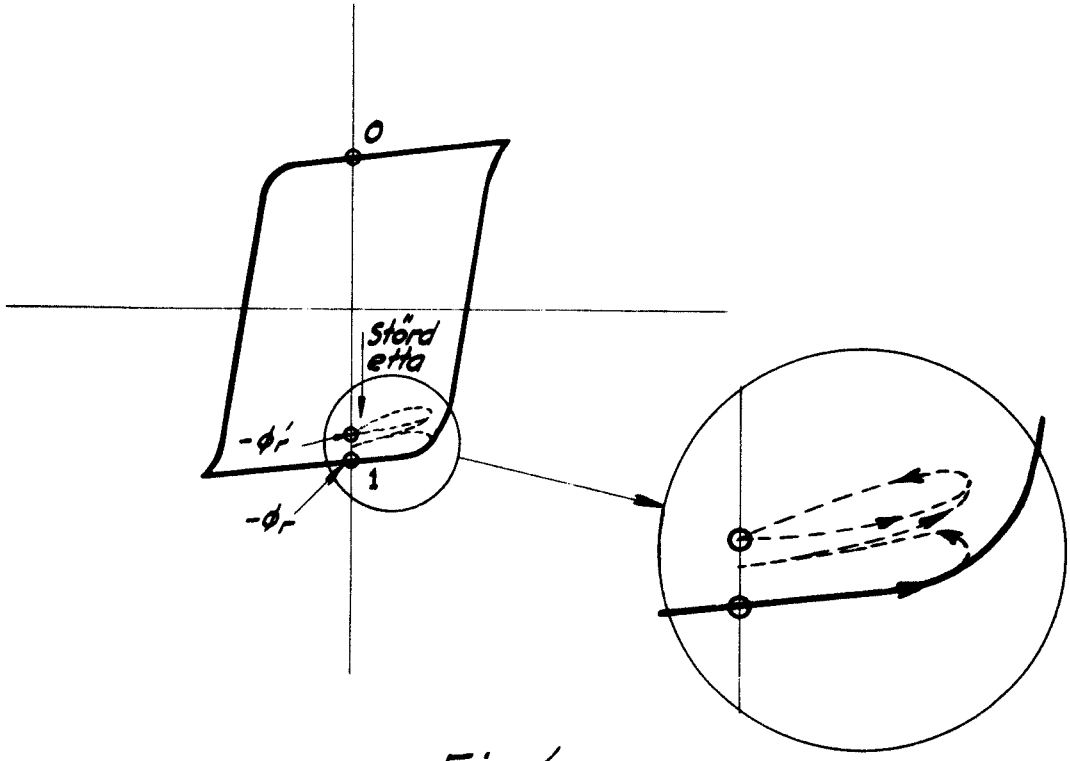

0
Störd
etta
$-\phi_r$
1
$-\phi_r$

Fig. 4

ledningar x och y. Varje kärna i denna matris kan tillordnas en adress, som anges av koordinater (n, m) definierade av x- och y-ledningarnas ordningsnummer. Samtliga kärnor i matrisen genomlöpas dessutom av en läsledning, som går diagonalt i förhållande till x- och y-ledningarna. Om en strömimpuls $+\frac{1}{2}$ im samtidigt skickas genom pulsledningarna x_n och y_m, blir endast kärnan (n, m) pulserad med en "helpuls" +im. Den spänning, som härvid induceras i läsledningen (se fig. 2), indikerar om en etta eller en nolla fanns lagrad i den utvalda kärnan. Om man önskar skriva en etta i adress (n, m) drives motsvarande pulsledningar med $-\frac{1}{2}$ im. Vid läsning enligt ovan i adress (n, m) blir alla kärnor på linjerna x_n och y_m halvpulserade (utom den utvalda). Varje halvpulserad kärna ger upphov till en störspänning på läsledningen. Som framgår av fig. 3 är läsledningen lagd så, att störspänningarna från en viss drivledning parvis kommer att motverka varandra. På grund av magnetiseringskurvans krökning ger en ett-ställd kärna upphov till en större störning än en nollställd. Den ur störningssynpunkt ogynnsammaste fördelningen av ettor och nollor i matrisen är den som uppkommer, om man skickar strömmen +im eller -im genom läsledningen. Ju större störningen är, desto svårare är det att vid läsning diskriminera mellan en etta och en nolla. Eftersom störspänningens maximum inträffar tidigare än maximum för en ettpuls, är det lämpligt att vid läsning använda en kombination av amplitud- och tidsdiskriminering. Kravet på goda marginaler vid denna diskriminering sätter en övre gräns för det antal kärnor, som kan hopkopplas i en och samma matris.

Om en kärna i ett-läge utsättes för upprepade halvpulseringar, $+\frac{1}{2}$ im, kommer magnetiseringen i kärnan att avta till ett värde $\emptyset_r' < \emptyset_r$, man får en "störd etta". Minskningen i $\emptyset_r$ bör vara så liten som möjligt, och det är vidare önskvärt, att ett stationärt tillstånd inträder efter så få halvpulseringar som möjligt. För en god kärna är detta tillstånd uppnått redan efter två halvpulser (fig. 4).

I en matris med N st. x-ledningar och M st. y-ledningar kan N·M binära siffror lagras. Om matrisen göres kvadratisk (N=M) blir adressväljningen enklast. Ett minne med en kapacitet av N^2 ord vartdera bestående av P binära siffror kan byggas upp av P st. matriser med N^2 kärnor i varje. Mot varje binär position svarar alltså en matris. Fig. 5 visar ett system med kvadratiska matriser. x- och y-ledningarna har kopplats i serie på sådant sätt att alla matriserna drivas samtidigt från gemensamma x- och y-aggregat. Utgångarna från maskinens adressregister delas upp i två grupper, som definierar en x- och en y-koordinat. Genom avkodning exempelvis med kristallmatriser utväljas ett x- och ett y-drivsteg, som pulseras från en central pulsgenerator styrd av blockpulsen

MATEMATIKMASKIN-NÄMNDENS ARBETSGRUPP	Ritad	7.8.56 Ra	KÄRNMINNE Rapport.	MNA
	Gransk.	J. Å.		KM-1017
	Godk.	ES		
	Reg.			

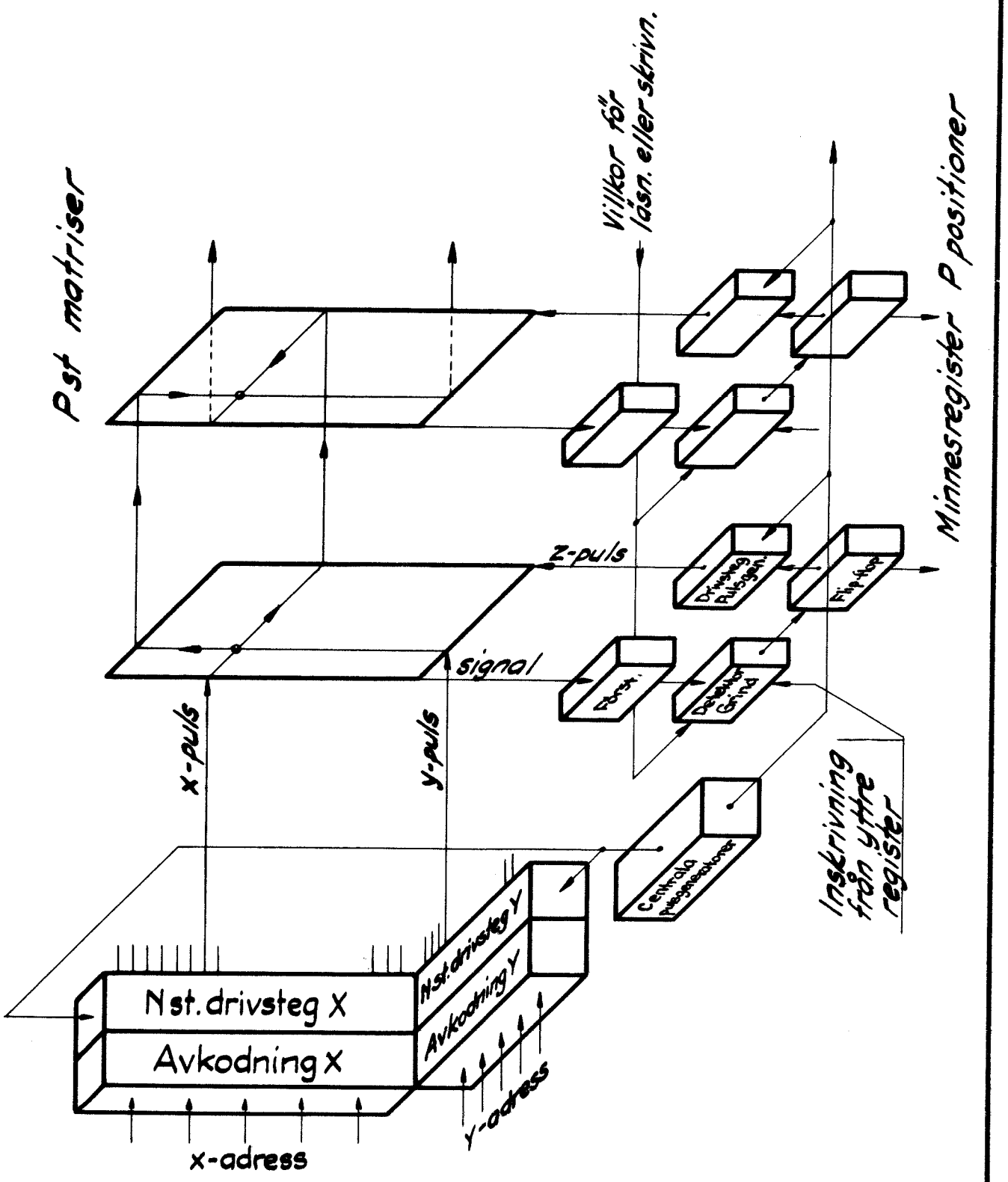


Fig. 5

i maskiner. Förutom x- och y-ledningarna är varje matris försedd med ytterligare en drivledning, z-ledningen eller sifferpulsledningen. Denna ledning löper liksom läsledningen genom alla kärnor i matrisen, men på sådant sätt att kärnorna magnetiseras med strömmen $+\frac{1}{2}$ im av en z-puls. Z-pulsen, som genereras villkorligt för varje matris, bestämmer om en etta eller en nolla skall inskrivas i matrisen. Pulseringen av minnet kan tänkas uppdelad i en läs- och en skrivfas (fig. 6). Under läsfasen drivas de utvalda x--och y-ledningarna med strömstyrkan $+\frac{1}{2}$ im. I varje matris ger den utvalda kärnan upphov till en spänningsstöt i läsledningen, som förstärkes och tolkas som en etta eller en nolla av detektorn. Information inmatas till minnesregistrets flip-flop-kretsar från grinden. Om minnet har fått order om läsning (vilket även måste innebära regenerering) grindas informationen från detektorerna ut till minnesregistret. Vid order om skrivning får minnesregistret i stället motta den utifrån kommande informationen. Det ord, som skall inskrivas, finns alltså vid skrivfasens början lagrat i minnesregistret. Under skrivfasen pulseras x- och y-ledningarna med $-\frac{1}{2}$ im. Villkoren för de olika matrisernas z-pulser tagas från motsvarande positioner i minnesregistret. z-pulsen genereras om siffran är noll men uteblir vid skrivning av en etta. Fig. 6 visar, att algebraiska summan av strömmarna genom den utvalda kärnan är $-\frac{1}{2}$ im vid skrivning av nolla. Villkoret z-pulsen kan alternativt utformas så, att pulsen för siffran ett fördröjes som antytts i fig. 6.

Snabbheten hos ett kärnminne bestäms i första hand av den tid, som erfordras för att ändra magnetiseringstillståndet i en kärna från $\emptyset_r$ till $+\emptyset_r$, den s.k. omslagstiden. För de material, som hittills kommit till användning, håller sig denna tid mellan 1 och 5 µs. För en pulseringsström av given styrka bestäms omslagstiden av materialets ledningsförmåga och koercitivkraft. God rektangularitet hos hysteresskurvan och låg koercitivkraft uppnås lättast för "långsamma" material (alltså med relativt hög ledningsförmåga). Snabba material har en koercitivkraft av ung. 1 oersted. För en ring med 2 mm ytterdiameter betyder detta att erforderlig magnetiseringsström im är ca 0,8 A. Med de snabbaste kärnor, som nu finns tillgängliga, kan en pulscykel (läs- och skrivfas) fullbordas på 5 µs. Hög pulsfrekvens i ett kärnminne ställer stora krav på elektroniken i såväl driv- som läskretsar.

Beskrivningen ovan gäller i stort sett för kärnminnet i Besk. I maskiner utomlands har även använts andra principer för kärnminnets utformning, inte bara beträffande pulseringssystemet utan också när det gäller själva matrisernas konstruktion och lindningssätt.

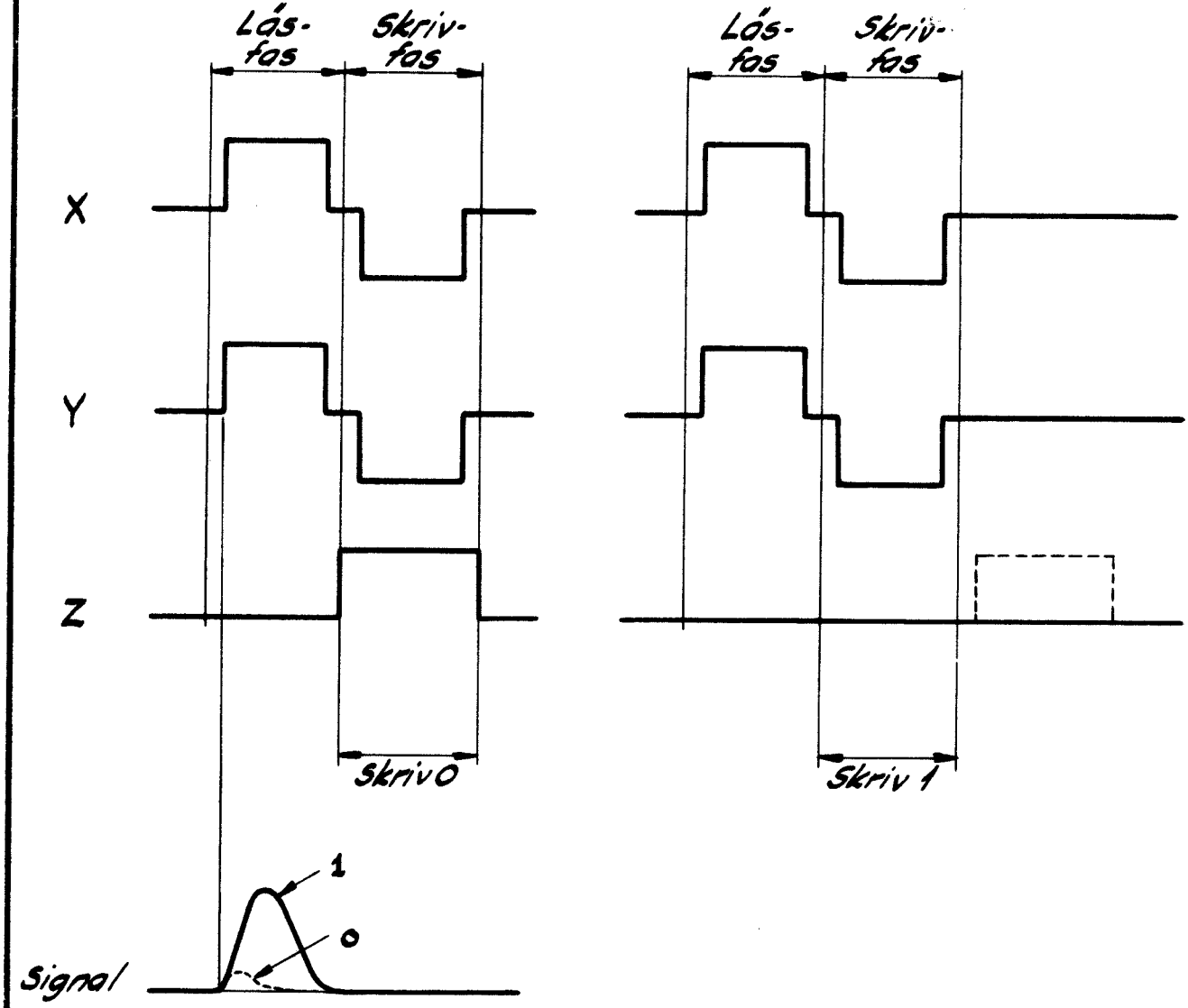

Läs-
fas
Skriv-
fas
Läs-
fas
Skriv-
fas
X
Y
Z
Skriv 0
Skriv 1
1
0
Signal

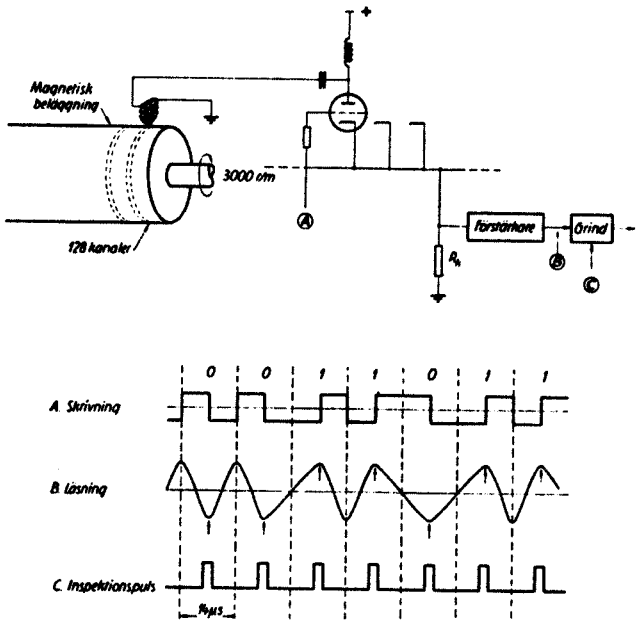
Fig. 6

Magnetiska trumminnet. Det magnetiska minnet består av två separata trummor med vardera 128 informationskanaler samt tre klockpulskanaler för generering av identifierings- och synkroniseringspulser. Till varje kanal hör ett magnethuvud med kärnor av ferrit. Trummorna är tillverkade av massiv mässing och belagda med tunt skikt av röd järnoxid. Trumdiamtern är 120 mm och luftgapet mellan trumma och huvud uppgår till 0,01 mm. Eftersom endast små variationer i luftgapet kan tillåtas, har trummorna slipats och balanserats med stor omsorg.

Varje kanal rymmer 32 fyrtiosiffriga binära tal, vilket motsvarar en lineär pulstäthet av 3,4 pulser per mm. Ett binärt tal i serieform inskrivs på trumman som en följd av positiva och negativa magnetiseringar, fig. 16. En nolla representeras sålunda av en positiv magnetiseringsperiod åtgöljd av en negativ av samma varaktighet och en etta av en negativ magnetiseringsperiod åtföljd av en positiv. För att en effektiv utradering av tidigare inskriven information skall erhållas bör den magnetiska beläggningen under magnetiseringsperioderna drivas till mättning i ena eller andra riktningen. Vid läsning induceras i magnethuvudets lindning en växelspänning, vars maxima och minima motsvarar magnetiseringarnas teckenväxlingar, fig. 16. En etta får således ett spänningsmaximum och en nolla ett spänningsminimum i sifferperiodens mitt. En inspektionspuls, som erhålls från en av klockpulskanalerna, avkänner signalens polaritet med hjälp av en pentodgrind.

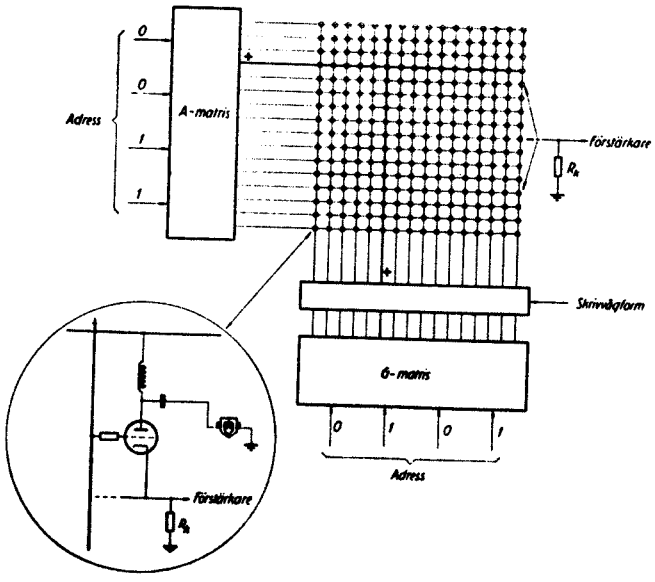
De elektroniska kretsarna till ett magnetiskt minne har till uppgift att välja en specificerad kanal samt att vid läsning eller skrivning motta eller avge signaler. Det faktum, att det vid skrivning fordras en amplitud på omkring 150 V, medan den vid läsning erhållna signalen endast uppgår till 0,05 V, innebär ett speciellt problem. Då härtill kommer, att ett elektronrör i princip är ett unilateralt element, har därför tidigare i regel till varje kanal fordrats separata kretsar för läsning och skrivning.

I den elektroniska väljaren i Besk i fig. 17 ingår en enkel triodkoppling, som möjliggör läsning och skrivning med en gemensam krets. De 256 magnethuvudena är anslutna till anoderna i ett system av trioder med gemensamt katodmotstånd. Trioderna är kopplade i matrisform med gallren radvis anslutna till 16 horisontala ledningar och triodernas anodinduktanser förbundna med 16 vertikala ledningar. Val av en kanal sker genom att en horisontal och en vertikal ledning bringas till positiv potential, medan de övriga hålls vid negativ potential. Det rör, som är inkopplat mellan de båda positiva ledningarna, blir härvid strömförande, medan strömmen i alla övriga rör är strypt genom att



Det magnetiska trumminnet, upptill trumma samt in- och avspelningshuvudet med tillhörande krets, nedtill representation av den binära informationen på trumman.

Fig. 16



Elektronisk väljare till magnetiskt trumminne.

Fig. 17

antingen galler eller anod eller såväl galler som anod har negativ potential. De enheter, som i figuren betecknas med A-matris och G-matris, har till uppgift att omvandla de båda adresserna, vilka anger den kanal som skall väljas från 4-siffrig binär form till "1 av 16-form". Vid skrivning tillförs G-matrisens positiva utgång skrivvågformen, som förstärks av den valda trioden. Katodmotståndet ger härvid en viss strömmotkoppling, vilket är en fördel, eftersom belastningen är rent induktiv.

Under läsning kan triodkopplingen betraktas som en katodföljare. Till skillnad från en konventionell katodföljarkoppling används här anoden som ingång, medan gallret hålls vid konstant potential. Över det gemensamma katodmotståndet erhålls en signal, som utgör ungefär $1/\mu$ (μ = förstärkningsfaktor) av signalen vid anoden. Denna relativt kraftiga dämpning kan emellertid reduceras genom att anodspänningen under läsning minskas till ett så lågt värde att gallerström uppstår. Ett motstånd i serie med gallret reducerar gallerströmmen till ett i förhållande till anodströmmen försumbart värde. Trioden blir då ekvivalent med ett motstånd, som i den aktuella kopplingen är av samma storleksordning som katodmoståndet.

Genom den beskrivna triodkopplingen har det varit möjligt att i den elektroniska väljaren nedbringa antalet rör till endast ett per kanal mot 4-5 i jämförbara system. Överföring av information mellan kärnminnet och trumminnet sker i block om 32 fyrtiosiffriga binära tal motsvarande en kanal på trumminnet.

Remsläsaren. Data och instruktioner matas in över aritmetiska enheten till kärnminnet med hjälp av en dielektrisk remsläsare. Sex avkänningselektroder, fig. 15, svarande mot remsans hålrader är kopplade till förstärkare med efterföljande detektorer. Remsan löper mellan avkänningselektroderna och en gemensam elektrod, som är kopplad till en utgång från en oscillator. En andra utgång med i förhållande till den första 180° fasförskjuten spänning är via neutraliseringskondensatorer förbunden med förstärkaringångarna. Dessa kondensatorer är så avpassade, att då ett hål i remsan befinner sig mitt för en elektrod spänningarna från denna och från neutraliseringskondensatorn tar ut varandra. Då man har papper i luftgapet får man däremot en ingångsspänning på förstärkaren och därmed en likspänning från detektorn.

Remsan läses med högst 400 rader per sekund och då läsningen avslutats stannar remsan på kortare tid än 1 ms genom att en magnetisk koppling mellan motor och framdrivningsrulle lösgöres samtidigt som en elektromagnetisk broms griper fast om remsan.

Utorgan. Resultaten skrivs ut på elektriska skrivmaskiner, som manövreras av ett antal solenoider, en för varje typarm. Utskriften sker i decimalsystem,+) vilket innebär, att maskinen översätter från binär till decimal form med hjälp av en översättningssekvens (en grupp av instruktioner), som matas in i maskinen samtidigt med instruktioner och övriga data. I den elektroniska enheten till skrivmaskinen ingår bl.a. en matris, som översätter den kodifierade decimalsiffran (i binär form) till en spänning, som med hjälp av solenoiderna utlöser ett av de 24 tecken som för närvarande används. Utom de 10 decimalsiffrorna måste man nämligen kunna ge order om mellanslag, vagnretur, tecken m.m.

Utskrivningshastigheten på skrivmaskin är i genomsnitt 12 tecken per sekund och alltså mycket låg jämfört med räknehastigheten. Ett avsevärt snabbare sätt för utmatning finns, nämligen med snabbstans, som f.n. stansar 145 tecken per sekund på remsa. Utskrift av remsan sker sedan i en skrivmaskinsenhet skild från Besk. Snabbstansen omtalas i kap. 9.

Manöverbordet. Manöverbordet innehåller start och stoppanordningar, indikeringslampor för de olika registren i maskinen samt knappar för manuell inmatning till dessa. För att underlätta felsökning är det möjligt att låta maskinen arbeta steg för steg eller med en variabel frekvens mellan 1-1000 Hz. Det är också möjligt att dela upp en instruktion i de olika stegen eller tempona och kontrollera resultatet av var deloperation. En multiplikation t.ex. kan delas upp i 46 deloperationer.

Manöverbordet innehåller också tidur, som anger hur länge olika spänningar stått på samt maskinens totala räknetid. En förstärkare och högtalare kan kopplas till ett register i aritmetiska enheten, och man kan höra rytmen av de olika sekvenserna i programmet. Man kan t.ex. höra om maskinen råkat fastna i en räkneprocess som ej konvergerar.

Likriktarenheten. Likriktarenheten matas med trefasspänning direkt från nätet. Från denna erhålls glöd- och anodspänning. Den förra höjs vid tillslag automatiskt i tre steg under 5 min. för att glödtrådarna skall skyddas. De övriga spänningarna är reglerade med hjälp av tyratroner, högvakuumrör eller vakuumstabiliserade aggregat. Vid höga strömmar används sexfasiga tyratronregulatorer med elektronisk reglering av tändningsögonblicket. Följande spänningar används:

+) eller sedecimalsystem

+400 V,	2,5 A	+100 V,	6,5 A
+300 V,	3,2 A	- 30 V,	3 A
+225 V,	1,5 A	-200 V,	11,5 A
+150 V,	15 A	-400 V,	0,3 A

Kraven på stabilitet är ca 0,2-0,5%.

Samtliga spänningar är variabla, vilket medger marginalundersökningar. Dessa går till så, att någon spänning sakta sänks, under det att maskinen bearbetar en särskild självkontrollerad sekvens. Då någon krets utför en felaktig operation på grund av den reducerade spänningen, skriver maskinen ut en sifferkombination, varur man i regel kan utläsa, var det svaga elementet finns.

Genom dylika prov är det möjligt att i förväg upptäcka svaga element i maskinen, innan de hunnit orsaka fel under normal drift. Sådana prov utförs dagligen eller vid fel.

Maskinen skyddas genom en omfattande automatik i likriktardelen för olika former av fel, kortslutning, bortfall av spänning på nätet, felmanövrering vid start o.s.v. Om t.ex. en kortslutning inträffar i någon del av maskinen, utlöses en säkring, som dels skiljer hela maskinen från likriktarenheten, dels signalerar felets läge till kontrollbordet.

Maskinens totala effektförbrukning är ca 15 kVA. Maskinen innehåller 2663 elektronrör och 1125 germaniumdioder.

Teckenförklaringar till tabellen över mikrooperationer.

Varje rad i tabellen över mikrooperationer (med undantag av rubrikraden) representerar ett mikrooperationstempo.

Första kolumnen innehåller beteckningar för olika tempon. Förutom ovannämnda tempon T_1-T_8 finns där ytterligare tempon med eller utan beteckningar, som förekommer i mera komplicerade operationer. Varje tempo tager en tid 7/u s = A/6 i anspråk med undantag av tempot T_s, som är a s y n k r o n t. Under detta tempo utföres, förutom i tabellen omnämnda mikrooperationer, även sådana, som berör inmatning, utmatning eller trumman. Tidsåtgången under T_s betingas av snabbheten hos ifrågavarande asynkrona yttre enhet.

Varje kolumn (med undantag av första och sista) representerar en hel operation eller klass av likartade operationer. Grundformerna för operationerna står i rubrikraden.

Sista kolumnen innehåller vissa tekniska mikrooperationer, som är gemensamma för alla operationerna. Dessa mikrooperationer utföres samtidigt och oberoende av eventuella andra mikrooperationer under samma tempo. Om ingenting står i en ruta i sista kolumnen, betyder det, att mikrooperationen endast har tekniskt intresse eller saknas helt (men tid går åt).

Bred linje under raden T_4 resp. över raden T_5 betecknar platsen i schemat, där kontrollutskrift, stopp (även vid körning stegvis) och start äger rum (i denna ordningsföljd). Det bör påpekas, att vid stopp resp. start utföres inte den aktuella ordern i sin helhet.

De särskilda tecknen betyder:

⟶	Innehållet i ett register överföres till ett annat och ersätter dess tidigare innehåll.
⟶+⟶	Innehållet i ett register eller en siffra adderas till innehållet i ett(annat) register.
▭	Tom ruta inne i schema betyder, att mikrooperationen inte utförs, men tid går åt
x	Mikrooperationen enligt sista kolumnen utförs
( )	kring en mikrooperation betyder, att operationen utförs icke alltid, men samma tid går åt.
↑	Mikrooperationen eller en grupp av sådana upprepas ett antal gånger.
↓	Mikrooperationen förbigås totalt; tid går inte åt.
V	Vänsterskift.
H	Högerskift.

Fig. 24 MIKROOPERATIONER I BESK

(teckenförklaring på föregående sida).

Tempo	10, 0D 0B, 0F	08, 09	11, 07	06, 46	02, 03	00	01	13	12	05, 45	04, 44	15, 55	0C, 0E 4E, 0A	1F	1B	19, 59	74	1C, 1D	18	
T_1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0 → BR w → OP
T_4	x (x)	x (x)	x (x)	x (2-19, 2-20 → AR) (x)	x (x)	x (x)	x (x)	x (x)	x (x)	x	x	x	x (x)	x (x)	x (x)	x (x)	x x	x (x)	x (x)	w → AS (0 → AR)
T_5														V ↑ AR	V ↑ AR	V ↑ AR	V ↑ AR		as → FS ar → FS	
T_m				w → MD										w → MD 0 → AR	ar → W					
T_n				md ← AR										md ← AR +2 ← AS	+2 ← AS					
T_k				V ↓ AR																
T_8	w → MD	w → MD	ar → W	ar → W	w → MD 24 → BR	w → MD 24 → BR	24 → BR	24 → BR	w → MD 24 → BR	as → BR	as → BR					ar → W				
T_6	md ← AR x	md ← AR x	x	x	x	md ← AR x	x	x	x	x	x	x	(x)	x	x	x	x	x	x	(kr → AS)
T_7	V ↓ AR x	V ↓ AR x	x	x	(1 → AR1) x	V ↓ AR x	x	x	x	x	x	x	(x)	x	x	x	x	x	x	(+1 → AS)
T_8	x	ar → MR x	x	x	x	H ↓ {AR, MR} x	H ↓ {AR, MR} x	V ↓ AR H ↓ MR	V ↓ AR x	x	x	0 → MR x	x	x	x	x	x	x	x	as → KR
B_1												(V ↓ AR, br → MR)								
B_{25}					md ← AR, H ↓ {AR, MR}	H ↓ {AR, MR}	H ↓ {AR, MR}	H ↓ MR, V ↓ AR	md ← AR, V ↓ AR, H ↓ MR			(V ↓ AR, br → MR)								
B_{44}					md ← AR, H ↓ {AR, MR}	H ↓ {AR, MR}	H ↓ {AR, MR}	H ↓ MR, V ↓ AR	md ← AR, V ↓ AR, H ↓ MR	(V ↑ AR)	(H ↑ AR)	(V ↑ AR, br → MR)								
B_{63}					md ← AR, H ↓ {AR, MR}	H ↓ {AR, MR}	H ↓ {AR, MR}	H ↓ MR, V ↓ AR	md ← AR, V ↓ AR, H ↓ MR	(V ↓ AR)	(H ↓ AR)									
M_{40}					md ← AR				H ↓ {AR, MR}											
M_{41}					H ↓ MR V ↓ AR															

Appendix I. Sakregister.

(Siffrorna anger sidnumreringen.)

A

ackumulator(register) = AR 1.3,3.1
addition 13.2
addera 3.6, 3.7, 4.1, 5.5
addera absolut 3.6, 5.1
adress 1.1
adressdel 1.2, 3.3
adressregister = AS 1.3, 3.3
analogirepresentation 1.7
analys 1.6
anpassning 7.5, 9.3
anpassningslängd 7.7
arbetskod 12.6
aritmetiska enheten 1.3, 3.1, 13.1
asterisk 3.9, 9.9
automatisk kodning 12.1
avrundningsfel 8.2

B

bas 2.1
belägenhetsord 12.6
beskaritmetik 2.6
beskkod 12.1, 12.3
beskkvot 4.5, 4.6
beskord 2.8
beskrest 4.5
besktal 2.8, 8.1
bibliotek för standardkoder 1.5, 11.1
binär 2.1
binär addition 2.4
binär aritmetik 2.4
binärkomma 2.1
binär multiplikation 2.5
binär representation 2.1
binärräknare = BR 3.3
binär subtraktion 2.5
binärt system 2.2
bit 2.1
blindorder 5.2
block 7.8
blockschema 3, fig. 4

C

card-to-tape-utrustning 9.13
carry 2.4
checksumma 9.3
cykel 4.7, 5.6

D

decimalkomma 2.1
decimalt system 2.1
delprogram 7.1
dividera 3.7, 4.5
dubbel noggrannhet 8.6 c

E

elektrisk skrivmaskin 1.3
elementära funktioner 11.17
etikett 9.1
E_2-stopp 9.6, 9.8, 10,3
E_2-utskrift 10.6
exponentialfunktioner 11.18
extrahera 3.7, 5.8

F

FA(5)program 12.11
FA-kod 12.6
FA-kodning 12.3
FA-order 12.4
fel 10.5
ferritminne 1.3
fiktiv adress 12.3
FLINTA 12.2
flytande binär form 8.3
flytande binärkomma 8.3
flytande räkning 8.3
flytande räkning (med R 10) 8.4
flödesschema 7.1
funktionsskrivare 1.3, 10.3

G

grundform 3.4, 3.5

H

halvcell 1.3
halvord (hao) 1.3, 2.6, 3.1
helcell (hec) 3.1
helord (heo) 2.6, 3.1
hopp, ovillkorligt 1.2, 3.8, 4.6
hopp, villkorligt 1.5, 3.8, 4.6
hoppa 3.8, 4.6
hoppa vid minus 3.8, 4.6
hoppa vid plus 3.8, 4.6
hoppa vid spill 3.8, 4.6
huvudprogram 7.3, 7.4, 7.8
hålkort 1.7
härledd form 3.4
högerhalvcell (hhac) 3.1
högerhalvord (hhao) 3.1

I

inkörning 1.6, 10.4
inläsning, decimal 6.4, 9.5, 11.7
inläsning, primitiv 6.2.b
inläsning, sedecimal 6.2.c, 9.1
inläsning av program 9.1
inläsning av remsa 10.1
inläsning av rättelser 6.1, 9.2, 9.4
inläsningssekvens 6.2.c, 9.1, 9.2, 11.7
interpretation 12.1
interpretationssekvens 12.1
irrelevant (irr) 3.6, 3.7, 3.8, 3.9

K

kanal 9.9
karakteristik 8.3
kod 1.2
kodbibliotek 1.5
kodning 1.6
kod(nings)mall 8.6.c
komparering av remsor 10.7
komplement 2.4, 2.5
kontrollnummer 1.2
kontrollregister = KR 1.3, 3.3
kontrollutskrift 3.4, 10.3
kopiering 9.12
konversion 2.2
kvadratrötter 11.18
kärnminne 1.3, 3.1, 13.5

L

lagra 3.7, 4.2, 5.4, 5.5
lexikon 12.3
lexikonord 12.6
likriktarenhet 13.10
logaritmfunktioner 11.19
längdord 9.3
läs från remsa 3.8, 6.1
läs från trumma 3.10, 6.6

M

magnetiskt trumminne 13.8
manuella operationer 10.4
manuell insättning i AR 10.2
manuell insättning i kärnminnet 10.2
manuell överföring från kärnminnet till AR 10.3
manöverbord 10.1, 13.10
matrisräkning 11.14
mellanslag 3.9, 9.9
mikrooperationsschema 13, fig. 18
minustecken 3.9, 9.9
modifiering 5.6, 5.7
multiplikand(register)= MD 1.3, 3.1
multiplikator(register) = MR 1.3, 3.1
märkning 12.4

N

namn 12.3
nollställning av trumminne 10.6
normalisera 3.8, 5.3
normaliserat tal 8.4
normalläge II 9.7, 10,8

O

omkastare 9.12
omstart 10.6
opackat tal 8.4, 11.7
operationsdel 1.2, 3.3
operationslista 3.6, 4.1, 5.1, 6.1
operationsregister = OP 1.3, 3.3
ordermodifikation 1.4, 1.5, 5.5

P

packad form 8.3
packat tal 8.3, 11.8, 11.15
parallellöverföring 3.3
permanenta konstanter 3.10
plustecken 3.9, 9.9
positionsvärde 2.8
potensfunktioner 11.19
programmering 1.6
program på trumman 7.7
pseudokod 12.1
pseudoorder 3.5
punkt 3.9, 9.9

R

rad 9.9
register 3.1
regulјär körning 1.6
relativ adress 12.2
relativ noggrannhet 8.2
remsapparatur 9.10
remslängd 7.7
remsläsare 1.3, 9.10, 13.9
remsor 9.9
rätta med parentes 4.8
rättelser 6.2.b, 12.9
rättelser (till trumman) 9.5

S

sedecimal addition 2.4
sedecimal aritmetik 2.4
sedecimal representation 2.1
sedecimal subtraktion 2.5
sedecimalt beskord 2.6
sedecimalt system 2.2
sekvens 4.8
sekvens, sluten (öppen) 7.5
serieöverföring 3.3
självinläsande hjälpsekvenser (elementära) 10.6

självinläsande hjälpsekvenser (speciella) 10.7
skala 8.1
skalfaktor, konstant 8.1
skalfaktor, variabel 8.3
skelettsekvens 7.9
skifta höger (vänster) 3.7, 5.1
skriv på trumma 3.9, 6.6
snabbstans 1.3, 9.9, 9.11
spill 2.14
spillindikation (si) 2.14, 3.1
standardprogram 1.5, 9.4, 11.1
stans 9.9
stansa siffra 3.9, 6.4
stansa tecken 3.9, 6.5
stansning 9.12
stansutrustning 9.11
start 10.1
start av remsa 9.1
start från remsa 6.3, 9.1, 9.3, 10.2
start med hopp 6.2.b, 10.2
startorder 9.2, 9.4
stopp 10.1
stoppkombination 3.5, 3.9, 9.9
styrorgan 1.3, 3.3, 13.3
subsekvens 7.1, 7.3
subtrahera 3.6, 4.1
subtrahera absolut 3.6, 5.1
subtraktion 13.3
summakontroll 9.4
sänd till funktionsskrivare 3.10, 6.7

T

tabulator 3.9, 9.9
taldel 8.3
talsystem, binärt 2.1
talsystem, decimalt 2.1
teckenposition 2.8
teknisk beskrivning av Besk 13.1
telextecken 6.2.a
totallängd 7.7
trigonometriska funktioner 11.20
trumetikett 9.4
trumkanal 7.8, 9.4
trumminne 1.6
trumprogram 7.7
tryckning, decimal 6.5, 7.3, 11.12
trycksekvens 11.12
tryck siffra 3.9, 6.4
tryck tecken 3.9, 6.5

U

understrykning 3.9, 9.9
utmatningsväljare 10.3
utorgan 13.10
utskrift, decimal 10.7
utskrift, sedecimal 10.6

V

vagnretur 3.9. 9.9
varvräkning 4.7. 5.7
verklig adress 12.4
Wheelerhopp, modifierat 7.4, 7.8
Wheelerhopp, normalt 7.2
vänsterhalvcell (vhac) 3.1
vänsterhalvord (vhao) 3.1

Y

yttre minne 1.3

Ö

öka adressdel 3.7, 5.5
överför mr till AR 3.7, 4.2
överför mr till AR bakvänt 3.7, 4.5
översättning 12.1

Appendix II. Tabeller.

Decimal - sedecimal konversion av heltal.

0	0	48	30	96	60	144	90	192	C0	240	F0
1	1	49	31	97	61	145	91	193	C1	241	F1
2	2	50	32	98	62	146	92	194	C2	242	F2
3	3	51	33	99	63	147	93	195	C3	243	F3
4	4	52	34	100	64	148	94	196	C4	244	F4
5	5	53	35	101	65	149	95	197	C5	245	F5
6	6	54	36	102	66	150	96	198	C6	246	F6
7	7	55	37	103	67	151	97	199	C7	247	F7
8	8	56	38	104	68	152	98	200	C8	248	F8
9	9	57	39	105	69	153	99	201	C9	249	F9
10	A	58	3A	106	6A	154	9A	202	CA	250	FA
11	B	59	3B	107	6B	155	9B	203	CB	251	FB
12	C	60	3C	108	6C	156	9C	204	CC	252	FC
13	D	61	3D	109	6D	157	9D	205	CD	253	FD
14	E	62	3E	110	6E	158	9E	206	CE	254	FE
15	F	63	3F	111	6F	159	9F	207	CF	255	FF
16	10	64	40	112	70	160	A0	208	D0		
17	11	65	41	113	71	161	A1	209	D1		
18	12	66	42	114	72	162	A2	210	D2		
19	13	67	43	115	73	163	A3	211	D3		
20	14	68	44	116	74	164	A4	212	D4		
21	15	69	45	117	75	165	A5	213	D5		
22	16	70	46	118	76	166	A6	214	D6		
23	17	71	47	119	77	167	A7	215	D7		
24	18	72	48	120	78	168	A8	216	D8		
25	19	73	49	121	79	169	A9	217	D9		
26	1A	74	4A	122	7A	170	AA	218	DA		
27	1B	75	4B	123	7B	171	AB	219	DB		
28	1C	76	4C	124	7C	172	AC	220	DC		
29	1D	77	4D	125	7D	173	AD	221	DD		
30	1E	78	4E	126	7E	174	AE	222	DE		
31	1F	79	4F	127	7F	175	AF	223	DF		
32	20	80	50	128	80	176	B0	224	E0		
33	21	81	51	129	81	177	B1	225	E1		
34	22	82	52	130	82	178	B2	226	E2		
35	23	83	53	131	83	179	B3	227	E3		
36	24	84	54	132	84	180	B4	228	E4		
37	25	85	55	133	85	181	B5	229	E5		
38	26	86	56	134	86	182	B6	230	E6		
39	27	87	57	135	87	183	B7	231	E7		
40	28	88	58	136	88	184	B8	232	E8		
41	29	89	59	137	89	185	B9	233	E9		
42	2A	90	5A	138	8A	186	BA	234	EA		
43	2B	91	5B	139	8B	187	BB	235	EB		
44	2C	92	5C	140	8C	188	BC	236	EC		
45	2D	93	5D	141	8D	189	BD	237	ED		
46	2E	94	5E	142	8E	190	BE	238	EE		
47	2F	95	5F	143	8F	191	BF	239	EF		

Decimal - sedecimal konversion av heltal.

256	100	1024	400	1792	700
272	110	1040	410	1808	710
288	120	1056	420	1824	720
304	130	1072	430	1840	730
320	140	1088	440	1856	740
336	150	1104	450	1872	750
352	160	1120	460	1888	760
368	170	1136	470	1904	770
384	180	1152	480	1920	780
400	190	1168	490	1936	790
416	1A0	1184	4A0	1952	7A0
432	1B0	1200	4B0	1968	7B0
448	1C0	1216	4C0	1984	7C0
464	1D0	1232	4D0	2000	7D0
480	1E0	1248	4E0	2016	7E0
496	1F0	1264	4F0	2032	7F0
512	200	1280	500	2048	800
528	210	1296	510		
544	220	1312	520		
560	230	1328	530		
576	240	1344	540		
592	250	1360	550		
608	260	1376	560		
624	270	1392	570		
640	280	1408	580		
656	290	1424	590		
672	2A0	1440	5A0		
688	2B0	1456	5B0		
704	2C0	1472	5C0		
720	2D0	1488	5D0		
736	2E0	1504	5E0		
752	2F0	1520	5F0		
768	300	1536	600		
784	310	1552	610		
800	320	1568	620		
816	330	1584	630		
832	340	1600	640		
848	350	1616	650		
864	360	1632	660		
880	370	1648	670		
896	380	1664	680		
912	390	1680	690		
928	3A0	1696	6A0		
944	3B0	1712	6B0		
960	3C0	1728	6C0		
976	3D0	1744	6D0		
992	3E0	1760	6E0		
1008	3F0	1776	6F0		

För några ofta återkommande konstanter ges här även en normaliserad form med tio siffrors noggrannhet med motsvarande karakteristik. Den packade formen erhålls lätt ur de två sista kolumnerna.

Exempel: Packad form för 1/7 är 49249 2497E
Packad form för 3/7 är 6DB6D B6E7F

	Besktal	Normaliserat	Karakteristik
1/2	40000 00000	40000 00000	80
1/3	2AAAA AAAAB	55555 55555	7F
2/3	55555 55555	55555 55555	80
1/4	20000 00000	40000 00000	7F
3/4	60000 00000	60000 00000	80
1/5	19999 9999A	66666 66666	7E
2/5	33333 33333	66666 66666	7F
3/5	4CCCC CCCCD	4CCCC CCCCD	80
4/5	66666 66666	66666 66666	80
1/6	15555 55555	55555 55555	7E
5/6	6AAAA AAAAB	6AAAA AAAAB	80
1/7	12492 49249	49249 24925	7E
2/7	24924 92492	49249 24925	7F
3/7	36DB6 DB6DB	6DB6D B6DB7	7F
4/7	49249 24925	49249 24925	80
5/7	5B6DB 6DB6E	5B6DB 6DB6E	80
6/7	6DB6D B6DB7	6DB6D B6DB7	80
1/8	10000 00000	40000 00000	7E
3/8	30000 00000	60000 00000	7F
5/8	50000 00000	50000 00000	80
7/8	70000 00000	70000 00000	80
1/9	0E38E 38E39	71C71 C71C7	7D
2/9	1C71C 71C72	71C71 C71C7	7E
4/9	38E38 E38E4	71C71 C71C7	7F
5/9	471C7 1C71C	471C7 1C71C	80
7/9	638E3 8E38E	638E3 8E38E	80
8/9	71C71 C71C7	71C71 C71C7	80
1/10	0CCCC CCCCD	66666 66666	7D
3/10	26666 66666	4CCCC CCCCD	7F
7/10	59999 9999A	59999 9999A	80
9/10	73333 33333	73333 33333	80

	Besktal	Normaliserat	Karakteristik
π	-	6487E D5111	82
$\frac{1}{\pi}$	28BE6 0DB94	517CC 1B727	7F
$\frac{1}{2\pi}$	145F3 06DCA	517CC 1B727	7E
$\frac{1}{\sqrt{2}}$	5A827 999FD	5A827 999FD	80
$^{10}\log 2$	26882 6A13F	4D104 D427E	7F
$^{10}\log e$	3796F 62A4E	6F2DE C549C	7F
$^{e}\log 2$	58B90 BFBE9	58B90 BFBE9	80
$^{e}\log 10$	-	49AEC 6EED5	82
$^{2}\log e$	-	5C551 D94AE	81

Appendix II. Tabeller.

Digniteter 10^h i sedecimal form och i normaliserad form med karakteristik.

h(dec.)	Heltal	Normaliserat	Karakteristik
0	1	40000 00000	81
1	A	50000 00000	84
2	64	64000 00000	87
3	3E8	7D000 00000	8A
4	2710	4E200 00000	8E
5	186A0	61A80 00000	91
6	F4240	7A120 00000	94
7	989680	4C4B4 00000	98
8	5F5E100	5F5E1 00000	9B
9	3B9ACA00	77359 40000	9E
10	2540BE400	4A817 C8000	A2
11	174876E800	5D21D BA000	A5
12	E8D4A51000	746A5 28800	A8
13	-	48C27 39500	AC
14	-	5AF31 07A40	AF
15	-	71AFD 498D0	B2
16	-	470DE 4DF82	B6
	Besktal		
-1	0CCCC CCCCD	66666 66666	7D
-2	0147A E147B	51EB8 51EB8	7A
-3	0020C 49BA6	41893 74BC7	77
-4	00034 6DC5D	68DB8 BAC71	73
-5	00005 3E2D6	53E2D 6238E	70
-6	00000 8637C	431BD E82D8	6D
-7	00000 0D6C0	6B5FC A6AF3	69
-8	00000 0157A	55E63 B88C2	66
-9	00000 00226	44B82 FA09B	63
-10	00000 00037	6DF37 F675F	5F
-11	00000 00005	57F5F F85E6	5C
-12	00000 00001	465E6 604B8	59
-13	-	70970 9A126	55
-14	-	5A126 E1A85	52
-15	-	480EB E7B9D	4F
-16	-	734AC A5F62	4B