STATISTISK SENTRALBYRÅS HÅNDBØKER

Nr. 15 Oslo, 25. april 1958

PROGRAMMERING FOR DEUCE

Første hefte

P R O G R A M M E R I N G F O R D E U C E

F ø r s t e h e f t e

Statistisk Sentralbyrå
Oslo, 1958

F o r o r d

Statistisk Sentralbyrå har tidligere sendt ut Innføring i DEUCE som tok sikte på å gi en generell orientering om bruken av EDB-maskiner. Med dette sender en ut første hefte av Programmering for DEUCE, som omhandler programmering av bearbeidingen. Annet hefte vil omfatte inn- og utlesning, bruk av kontrollbordet og testing av programmer m.m.

Programmering for DEUCE er utarbeidd av førstesekretær Thor Aastorp i tilknytning til de kurser som Gruppen for metodespørsmål og statistiske standarder holder for Byråets funksjonærer. Ellers har Norsk Regnesentral bidratt med verdifulle merknader til manuskriptet.

Statistisk Sentralbyrå, Oslo, 25. april 1958

Petter Jakob Bjerve

Svein Nordbotten

I n n h o l d

I. INNLEDNING

D E U C E (Digital Electronic Universal Computing Engine) er en elektronisk regnemaskin som automatisk og med stor hastighet kan utføre de serier av elementæroperasjoner som er nødvendig for å løse de forskjelligste oppgaver av numerisk art. Den første DEUCE ble levert fra produsenten, The English Electric Company Limited, i 1955.

DEUCE er e l e k t r o n i s k. All lagring, transport, aritmetiske og logiske operasjoner foregår ad elektronisk veg, derfor er den interne hastigheten betydelig større enn i de regnemaskinene som benytter mekaniske eller elektro-mekaniske elementer for disse operasjonene.

Elektroniske og andre regnemaskiner kan deles i to hovedgrupper etter måten hvorpå tall blir representert i maskinene. DEUCE tilhører den gruppen som går under benevnelsen s i f f e r m a s k i n e r, fordi alle tall blir representert internt i disse maskinene ved hjelp av tegn eller symboler. I den andre hovedgruppen, a n a l o g m a s k i n e n e, representeres tallene av en kontinuerlig variabel fysisk størrelse: en lengde, en fjærspenning, en elektrisk strøm eller liknende. Typiske eksempler på en slik representasjonsmetode finner en i regnestaven og ampéremeteret.

Navnet forteller dessuten at DEUCE er en u n i v e r s a l m a s k i n. Med dette menes at den er konstruert for løsning av de forskjelligste oppgaver. Andre maskiner som er bygget for et bestemt formål og hvor det ikke er mulig å forandre det arbeidsprogrammet som maskinen følger, kalles spesialmaskiner.

DEUCE kan, som nevnt, løse numeriske oppgaver av ulik art. For hver oppgave som DEUCE skal løse, må først utarbeides en arbeidsrutine eller et arbeidsprogram. Dette arbeidsprogrammet som spesifiserer de elementæroperasjoner som er nødvendig for løsning av oppgaven, må lages i et slikt språk og form at maskinen kan forstå det. Før DEUCE kan starte med operasjonene, må programmet lagres i maskinen; dette lagrede programmet vil så dirigere og kontrollere alle operasjonene. DEUCE betegnes derfor ofte som programstyrt eller selvstyrt.

Rent skjematisk består DEUCE av fire komponenter (fig. 1):

1. I n p u t / O u t p u t - o r g a n. Dette er en IBM 528 (Accumulating Reproducer) med en normal innlesningshastighet på 200 kort pr. minutt og punchehastighet på 100 kort pr. minutt.

2. M i n n e t lagrer og oppbevarer det program, data og øvrige opplysninger som er nødvendig for løsning av den enkelte oppgave.

3. R e g n e - o r g a n e t kan utføre de vanlige regneartene på lagrede tall.

4. K o n t r o l l - o r g a n e t sørger for at de enkelte operasjonene som er spesifisert i programmet, blir utført til riktig tid og i riktig rekkefølge.

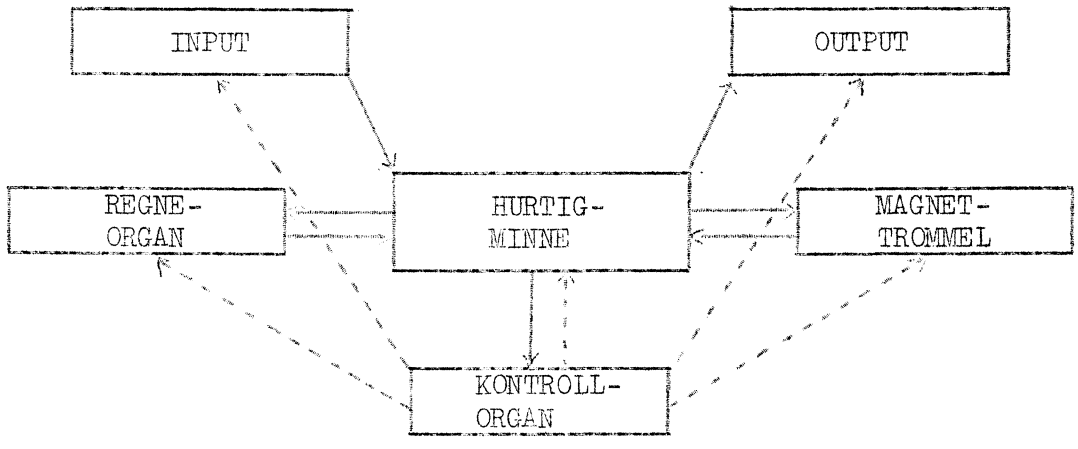


Fig. 1

En må være oppmerksom på at den skjematiske framstillingen i fig. 1 bare viser den rent logiske oppbyggingen av DEUCE. Fysisk vil en ikke kunne trekke noe tilsvarende skille mellom alle disse komponentene.

Som en ser av figuren er hurtigminnet den sentrale delen av systemet. Alle opplysninger som skal inn i maskinen, både tall, instruksjoner og andre informasjoner, går via input-organet til hurtigminnet for lagring; likeså må alle resultater osv. som skal ut av maskinen, overføres fra hurtigminnet til output-organet for punching. På tilsvarende måte må all overføring mellom de andre komponenter gå via hurtigminnet. De brutte strekene i figuren betyr at det er kontrollorganet som dirigerer operasjonene i de øvrige komponentene ut fra et arbeidsprogram.

En vil i dette hefte forsøke å føre leseren inn i grunnprinsippene for utarbeiding av arbeidsprogrammer for DEUCE. Før en kan gjøre dette, er det imidlertid nødvendig å kjenne en del til hvordan tallene blir representert internt i DEUCE.

II. BINÆR REPRESENTASJON

1. Det binære tallsystem

Når en leser tallet 1485, vet en med en gang hva det betyr, nemlig:

$$1000 + 400 + 80 + 5$$

eller

$$1 \cdot 10^3 + 4 \cdot 10^2 + 8 \cdot 10^1 + 5 \cdot 10^0$$

På tilsvarende måte vet vi at 14,85 står for

$$10 + 4 + 0{,}8 + 0{,}05$$

eller

$$1 \cdot 10^1 + 4 \cdot 10^0 + 8 \cdot 10^{-1} + 5 \cdot 10^{-2}$$

En kaller dette tallsystemet for desimal- eller 10-tallsystemet, fordi 10 er grunntallet. En trenger 10 forskjellige talltegn eller symboler, 0, 1, 2, ,,,, 9. I desimalsystemet uttrykker en alle tall som en sum av forskjellige potenser av 10. For letthets skyld skriver en imidlertid ikke disse potensene, men nøyer seg med å skrive de koeffisientene som potensene skal multipliseres med. Plaseringen av koeffisientene i forhold til desimalkommaet vil i hvert enkelt tilfelle avgjøre hvilken potens av 10 det gjelder. I stedet for å skrive hele dette lange uttrykket:

$$5 \cdot 10^3 + 8 \cdot 10^2 + 0 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 7 \cdot 10^{-2}$$

skriver vi bare:

$$5803{,}47$$

Regnemaskiner som internt arbeider med tallene i desimalform, benytter seg av elementer som kan innta 10 forskjellige tilstander, én tilstand for hvert talltegn. Et eksempel på et slikt element er regneverkshjulet. Noe liknende utstyr vil ikke være praktisk i en elektronisk maskin. Det er imidlertid intet til hinder for at dette kan lages, men sammenliknet med elementer som kan innta bare to tilstander, vil slikt utstyr være dårligere både økonomisk sett og med hensyn til sikkerheten. Elektroniske regnemaskiner benytter derfor elementer med bare to tilstander; disse elementene anvendes imidlertid ikke på samme måte i alle maskinene, men etter to helt forskjellige prinsipper. I den ene gruppen av maskiner blir hvert enkelt desimalsiffer representert ved hjelp av en gruppe slike elementer; alle tall blir da behandlet som desimale tall, slik at aritmetiske operasjoner blir utført i det desimale tallsystemet. I binærmaskinene, som danner den andre gruppen, blir alle tall derimot behandlet som binære tall, og alle regneoperasjoner utføres i det binære tallsystemet. I disse maskinene blir hvert enkelt binærsiffer representert ved hjelp av et element. Binærmaskinene krever ikke så mye teknisk utstyr som desimalmaskinene fordi tallrepresentasjonen og regnereglene er betydelig enklere.

DEUCE er i likhet med mange andre elektroniske siffermaskiner en binærmaskin. Arbeidsprogrammene for DEUCE vil imidlertid normalt bli laget slik at data som skal inn i maskinen, kan leses inn i desimalform; maskinen vil da, ved hjelp av et program, sørge for å få regnet om tallene til binærform. På samme måte vil resultater og opplysninger som skal ut av maskinen, bli regnet om fra binærform til desimalform før de blir sendt til output-organet for punching.

I binærsystemet er 2 grunntallet, og en trenger derfor bare to forskjellige talltegn eller symboler, 0 og 1. Alle tall skrives da som en sum av potenser av 2:

2^0	(definert som)	1
2^1	=	2
2^2	=	4
2^3	=	8
osv.		osv.

Desimaltallet 27 vil kunne skrives som:

$$16 + 8 + 0 + 2 + 1$$

eller

$$1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

På tilsvarende måte vil desimaltallet 5,625 skrives som:

$$4 + 0 + 1 + 0{,}5 + 0 + 0{,}125$$

eller

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

På tilsvarende måte som i desimalsystemet skriver vi ikke disse lange uttrykkene, men utelater potensene av grunntallet og nøyer oss bare med å skrive koeffisientene. Disse koeffisientenes plasering i forhold til binær-kommaet forteller hvilken potens av 2 de enkelte koeffisientene skal multipliseres med. En får da:

Desimal	Binær
26	11010
8,625	1000,101

De første tallene i desimalsystemet ser slik ut i binærform:

Desimal	Binær	Desimal	Binær
0	0	9	1001
1	1	10	1010
2	10	11	1011
3	11	12	1100
4	100	13	1101
5	101	14	1110
6	110	15	1111
7	111	16	10000
8	1000	17	10001

Som en ser trengs vanligvis flere sifre for å uttrykke et tall binært enn desimalt. For store tall vil forholdet være omtrent som log 10 : log 2, eller omkring 10 : 3.

Til alle hele desimale tall svarer et binært med nøyaktig samme verdi. En desimal brøk kan imidlertid bare i visse tilfelle regnes om til en binær brøk av nøyaktig samme verdi; en kan komme så nær den desimale verdien en vil ved å ta med tilstrekkelig mange binær-siffer etter binær-kommaet.

Eksempel:

Desimalbrøken 1,3 omregnet til binærform:

Med 4 binærposisjoner etter kommaet:

1,0101 = 1,3125

Med 8 binærposisjoner etter kommaet:

1,01001101 = 1,30078125

Med 12 binærposisjoner etter kommaet:

1,010011001101 = 1,300048828125

Med 16 binærposisjoner etter kommaet:

1,0100110011001101 = 1,3000030517578125

2. Binær tallrepresentasjon i DEUCE

Lagring og transport av tall internt i elektronmaskinene foregår etter to hovedprinsipper:

a. P a r a l l e l l. Etter dette prinsippet vil alle sifrene i et tall være tilgjengelige til samme tid, og ved overføring internt i maskinen blir alle sifrene overført samtidig, dvs. parallelt. Maskiner som benytter dette prinsippet, trenger et element for lagring av hvert siffer og en kanal for overføring av hvert siffer. Hullkortmaskinene er slike parallell-maskiner.

b. S e r i e. I seriemaskinene vil ikke alle sifrene i et tall være tilgjengelig på samme tid, men bare ett og ett siffer om gangen med et visst tidsintervall mellom sifrene. DEUCE benytter dette serie-systemet; alle tall blir lagret og transportert inne i DEUCE som en serie av elektriske impulser (impuls-tog). Ved serie-lagring trengs bare et element for lagring av et tall; det vil også være mulig å lagre flere tall i samme element, og det trengs bare en kanal ved transport av et tall eller en serie tall.

I DEUCE blir altså alle tall lagret i maskinen som en serie av elektriske impulser, impuls = 1, ingen impuls = 0. Det laveste siffer vil alltid

komme først i en slik serie av binære siffer, fordi en alltid begynner med det laveste siffer ved addisjon. Desimaltallet 26 som i binærform skrives 11010, vil se slik ut i en impuls-serie:

Først ingen impuls	(0)	= 0
Så impuls	(1)	= 2
Så ingen impuls	(0)	= 0
Så impuls	(1)	= 8
Så impuls	(1)	= 16
		26

Som kjent er den vanligste måten å tegne en tidsrekke på fra venstre til høyre. Dette har bevirket at når det gjelder DEUCE, blir alle binærtall skrevet med laveste (første) siffer til venstre og høyeste (siste) siffer til høyre. Da dette er motsatt av vanlig praksis, kalles dette "omvendt" eller "kinesisk" binærform.

For å unngå forvirring og misforståelser, vil en i dette hefte konsekvent gjennomføre denne omvendte binærform.

Noen eksempler:

Desimal	Vanlig binær	Omvendt binær
8	1000	0001
13	1101	1011
19	10011	11001
32	100000	000001
17,25	10001,01	10,10001

Vi kjenner til fra vanlige bordmaskiner og hullkortmaskiner at de forskjellige regneverk og registre har en viss kapasitetsgrense, dvs. de kan ikke inneholde mer enn et bestemt antall siffer. Det samme er tilfelle for DEUCE; alle tall og opplysninger lagres i DEUCE som en serie av 32 bits (bit = Binary Digit). En kaller en slik gruppe á 32 bits for " e t o r d ", og en sier at ordlengden er 32 bits. En har valgt betegnelsen "ord" istedenfor "tall" fordi som en senere skal se, behøver nødvendigvis ikke disse 32 bits være et tall, men kan også være en instruksjon, alfabetiske opplysninger osv.

I en maskin med 32 bits ordlengde er det mulig å arbeide med 2^{32} forskjellige tall. Hvis en antar at disse tallene ligger i området 0 til $2^{32} - 1$, eller binært fra (000 ....000) til (111 .... 111), vil en ikke ha noen mulighet til å kunne behandle negative størrelser. En sier at dette er en metode uten fortegn. Det er nemlig ikke noe spesialutstyr i DEUCE for indikering av fortegn. Når både positive og negative tall forekommer, må en derfor sørge for at det tillatte tallområde strekker seg like langt på begge sider av null. Det betyr at det største positive tall bare vil

være halvparten av det største tallet etter metoden uten fortegn, men til gjengjeld vil en også kunne behandle negative tall med tilsvarende tallverdi. Denne siste måten å betrakte tallområdet på kan vi kalle fortegnsmetoden.

Det er flere fortegnsmetoder. Den som brukes i DEUCE, har den fordelen at binær addisjon kan utføres på samme måten både etter metoden uten fortegn og med fortegn. DEUCE kan derfor benyttes til regning etter begge metodene; resultatet av en regneoperasjon blir da å fortolke i forhold til den metoden som er brukt i hvert enkelt arbeidsprogram.

En har følgende regler for hvordan tall blir representert i DEUCE etter fortegnsmetoden:

a) Positive tall i området 0 til $(2^{31} - 1)$ blir representert på vanlig binær måte.

b) Negative tall (-n) fra -2^{31} til -1 blir i DEUCE representert ved binærtallet $(2^{32} - n)$.

I DEUCE har en altså følgende tillatte tallområde:

$$-2^{31} \leqslant x \leqslant 2^{31} - 1$$

eller

$$-2\ 147\ 483\ 648 \leqslant x \leqslant 2\ 147\ 483\ 647$$

Dette er tilstrekkelig til å dekke alle tall med inntil 9 desimalsiffer.

En skal se noen eksempler på hvordan en del binærtall blir fortolket etter de to metodene:

Ordlengden = 6 bits.

Binær	Desimal u/fortegn	Desimal m/fortegn
000001	32	-32
100001	33	-31
010001	34	-30
101111	61	- 3
011111	62	- 2
111111	63	- 1
000000	0	0
111110	31	31

Som en ser vil det høyeste av de 32 bits (dvs. høyre bit) indikere fortegnet: 0 for positive og 1 for negative tall. Enkelt kan en si tallrepresentasjonen er slik: positive tall representeres ved sin normale binær-form, mens negative tall blir representert ved sitt "toer"-komplement. Dette "toer"-komplementet er analogt til "tier"-komplementet i desimalsystemet.

For å finne "toer"-komplementet til et tall, kan en bruke følgende regel: Begynn med laveste siffer og kopier alle siffer til og med første

"ener", deretter forandres alle enere til nuller og alle nuller til enere.

Eksempel:

Ordlengden = 6 bits.

14	=	011100	1	=	100000
-14	=	010011	-1	=	111111
10	=	010100	-32	=	000001
-10	=	011011			

En har hittil behandlet alle tall som hele tall. Når en arbeider med brøker, vil en kunne benytte akkurat de samme fortegnsmetoder, men en må under utarbeidingen av arbeidsprogrammet hele tiden holde rede på og ta hensyn til hvor binær-kommaet befinner seg.

3. Litt om binær regning og andre operasjoner i DEUCE

En skal i dette avsnittet se litt på de regnereglene som gjelder for binær-systemet, og forklare litt om hvordan regningen foregår i DEUCE. For å forenkle skrivingen antas at ordlengden er 6 bits; prinsippene vil bli akkurat de samme for en ordlengde på 32 bits.

A d d i s j o n

Addisjonsreglene er svært enkle:

0 + 0 = 0	1 + 0 = 1
0 + 1 = 1	1 + 1 = 0 og 1 i mente

Eksempler:

1)	010100	(10)	3)	101000	(5)
	+111000	(7)		+110111	(-5)
	=100010	(17)		=000000	(0)
2)	011100	(14)	4)	111100	(15)
	+100001	(-31)		+100010	(17)
	=111101	(-17)		=000001	(-32)

Hvis en ser nærmere på eksemplene ovenfor, ser en at i eksempel 1 og 2 blir addisjonene riktig etter begge metodene, både med og uten fortegn. I eksempel 3 derimot vil en få et galt svar etter metoden uten fortegn, mens en i eksempel 4 vil få et galt svar etter fortegnsmetoden. I begge disse eksemplene har summen blitt så stor at den ikke faller innenfor det tillatte tallområde. Dette kalles "over-flow".

På grunn av denne mulighet for "over-flow" må en være varsom under utarbeidingen av arbeidsprogram. En må enten sørge for at tallene holder seg innenfor det tillatte tallområdet, eller en må la DEUCE selv varsle når

"over-flow" inntreffer. En måte å minske sjansene for "over-flow" er å arbeide med tallene i dobbelt lengde, dvs. at ordlengden er 64 bits istedenfor 32 bits.

S u b t r a k s j o n

Ved subtraksjon benytter en seg av "toer"-komplementet til tallet. Som nevnt før, finner en "toer"-komplementet til et tall ved å kopiere alle bits til og med første ener; deretter erstattes alle enere med nuller og nuller med enere.

Eksempel:

Komplementet til 101000 er 110111
" " 011001 " 010110

Subtraksjon av et tall foregår altså ved å addere "toer"-komplementet til tallet.

Eksempler:

1) 010100 (10) = 010100 + 100111 = 110000 (3)
-111000 (7)

2) 101010 (21) = 101010 + 101000 = 010110 (26)
-110111 (-5)

3) 001100 (12) = 001100 + 110001 = 111101 (-17)
-101110 (29)

4) 000111 (-8) = 000111 + 110100 = 110000 (3)
-101011 (-11)

Også ved subtraksjon må en være oppmerksom på mulighet for "over-flow".

M u l t i p l i k a s j o n

"Den lille multiplikasjonstabellen" er meget kort og enkel i binær-systemet:

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

Selve multiplikasjonen foregår ved addisjon; for hvert multiplikatorsiffer vil multiplikanden bli addert eller ikke addert avhengig av verdien av multiplikatorsifferet (0 eller 1). For hver gang vil multiplikanden bli flyttet en plass til høyre. Begge faktorene i en multiplikasjon må være av standard ordlengde, 32 bits; produktet derimot vil opptre med dobbelt ordlengde, 64 bits.

Eksempler:

1) (7) (5)
111000 x 101000
111000
+ 0
+ 111000
= 110001000000 (35)

2) (1,5) (2,5)
1,10000 x 1,01000
110000
+ 0
+ 110000
=11,1100000000 (3,75)

Som en ser er produktet riktig; produktet vil alltid være riktig etter metoden uten fortegn. En kan også multiplisere brøker (eks. 2); binærkommaets plasering i produktet finnes da etter følgende regel:

Antall siffer til venstre for kommaet i produktet er lik summen av antall siffer til venstre for kommaet i de to faktorene.

Resultatet av en multiplikasjon av to positive tall vil alltid bli korrekt i DEUCE. Hvis derimot den ene eller begge faktorene er negative, må en foreta visse korreksjoner til resultatet for å få et korrekt produkt med fortegn. Et negativt tall "-a" vil i DEUCE være representert som "$2^{32} - a$". Når dette tallet multipliseres med et positivt tall "b", vil produktet bli $(2^{32} - a)b$ eller $2^{32}b - ab$. Det riktige produktet etter fortegnsmetoden og med dobbelt ordlengde skal imidlertid være $2^{64} - ab$. Det første resultatet må derfor korrigeres med uttrykket $(2^{64} - 2^{32}b)$ eller $2^{32}(2^{32} - b)$. $(2^{32} - b)$ er DEUCE-representasjonen av "-b". Dette betyr at "b" må subtraheres i høyeste halvpart for å få det riktige produktet.

Hvis begge faktorene er negative "-a" og "-b", vil disse være representert henholdsvis ved $(2^{32} - a)$ og $(2^{32} - b)$; produktet av disse tallene er $2^{64} - 2^{32}(a + b) + ab$. Det riktige produktet er $(+ab)$, altså må det første produktet korrigeres med følgende uttrykk "$2^{32}(a + b)$" for å få det riktige produktet. 2^{64} behøver ikke subtraheres, da dette tall allikevel faller utenfor kapasiteten.

En har da følgende regel: Hvis en av faktorene i en multiplikasjon er negativ, må den andre faktoren subtraheres i den høyeste halvpart av produktet. Hvis begge faktorene er negative, må begge subtraheres i høyeste halvpart av produktet.

Eksempel:

```
    (-3)      (+5)                 (-5)      (-3)
  101111 x 101000                110111 x 101111

  101111                         110111
   0                           + 0
+ 101111                      +  110111
  100011001000                +   110111
+      110111   (-5)           +    110111
  ------------                +     110111
  100011111111   (-15)          =111100000111
                              +       101000    (+5)
                              +      110000     (+3)
                                =111100000000   (+ 15)
```

D i v i s j o n

DEUCE har innebygd divisjonsutstyr. Dette er konstruert slik at kvotienten alltid vil få riktig fortegn. Divisjonen foregår ved gjentatte subtraksjoner.

Når en benevner de forskjellige størrelsene som inngår i en divisjon slik:

Dividenden	A
Divisor	B
Kvotienten	Q
Resten	R
Modifisert rest	r

kan en sette opp forholdet mellom disse på følgende måte:

$$2^{31}A = Q\,B + R$$

hvor $0 \leq R < B$ hvis $B > 0$

$B \leq R < 0$ hvis $B < 0$

Alle disse størrelsene må være av enkel ordlengde, dvs. 32 bits. Ut fra relasjonen foran ser en umiddelbart at hvis A er forskjellig fra null, må også B være forskjellig fra null; ellers vil Q være uendelig stor.

Videre får en fordi både A, B og Q er av enkel ordlengde, at:

$$|A| \leq |B| \leq 2^{31}$$

dvs. tallverdien av A må være lik eller mindre enn tallverdien av B, og begge tallverdiene må være mindre enn 2^{31}. Tallverdiene til A og B kan bare være like stor i de tilfellene A er negativ og B er positiv.

Ovenfor ser en at R alltid har samme fortegn som B, dvs. den er positiv når B er positiv og negativ når B er negativ. Kvotienten vil derfor i de tilfellene R ikke er lik null, alltid være algebraisk mindre enn den riktige kvotienten. En ser dessuten at en eksakt kvotient (R = 0) bare kan være mulig når B er positiv. I de øvrige tilfellene vil kvotienten være algebraisk mindre, dvs. tallverdien vil, sammenliknet med den eksakte kvotient, være 1 mindre i laveste bit.

En vil komme nærmere tilbake til divisjon i kap. V.

S k i f t

Innebygd i DEUCE er det anordning for multiplikasjon og divisjon med 2. Da maskinen arbeider i binær-systemet, betyr dette ikke annet enn skifting eller flytting av tallet henholdsvis opp, dvs. til høyre, eller ned, dvs. til venstre. En må imidlertid være oppmerksom på at denne skiftingen foregår etter fortegnsmetoden.

Eksempler:

	Original		x 2		: 2	
1)	011000	(6)	001100	(12)	110000	(3)
2)	110000	(3)	011000	(6)	100000	(1)
3)	010111	(-6)	001011	(-12)	101111	(-3)
4)	101111	(-3)	010111	(-6)	011111	(-2)
5)	111001	(-25)	011100	(14)	110011	(-13)
6)	111010	(23)	011101	(-18)	110100	(11)

Merknader til eksemplene:

Multiplikasjon med 2:

a) I eksemplene 3 og 4 vil resultatene være riktige etter fortegnsmetoden, men gale etter metoden uten fortegn.

b) I eks. 5 vil resultatet være galt etter begge metodene, fordi originaltallet er så stort at det inntreffer "over-flow".

c) Resultatet i eks. 6 er galt etter fortegnsmetoden, men riktig uten fortegn.

Divisjon med 2:

d) I eks. 2, 4, 5 og 6 ser en at når et ulike tall divideres med 2, mistes det siste sifret slik at det blir en feil på 1/2.

e) I eks. 3, 4 og 5 ser en at når et negativt tall divideres med 2, blir en ener automatisk plasert som höyeste siffer, slik at resultatet blir riktig etter fortegnsmetoden, men galt etter metoden uten fortegn.

L o g i s k e o p e r a s j o n e r

1. Rudimenter addisjon

Rudimenter addisjon foregår ved at sifrene i to tall adderes, posisjon for posisjon, uten mente. Symbolet for denne operasjonen er " $\neq$ ".

Eksempel:

```
  101011   (-11)
≠ 011110   (30)
= 110101   (-21)
```

En enkel regel: Resultattallet etter en " $\neq$ "-operasjon har 1 i de posisjonene hvor de to tallene er ulike, men det er 0 i de posisjonene hvor tallene er like.

Denne operasjonen kan være svært nyttig, f.eks. hvis en önsker å undersöke om to tall er like; hvis nemlig resultatet av en rudimenter addisjon av to tall er lik null, må tallene være like store.

Ofte kan det være svært nyttig å være oppmerksom på at når a ≠ b = x, så er a ≠ x = b og b ≠ x = a.

Eksempel:

a = (101101), b = (011000) og x blir (110101)

1)		2)		3)	
101101	(a)	101101	(a)	011000	(b)
≠ 011000	(b)	≠ 110101	(x)	≠ 110101	(x)
= 110101	(x)	= 011000	(b)	= 101101	(a)

2. Logisk multiplikasjon

Logisk multiplikasjon foregår ved at sifrene i to tall multipliseres, posisjon for posisjon. Tegnet for denne operasjonen er " & ".

Eksempel:

101011	(-11)
& 011110	(30)
= 001010	(20)

En enkel regel: Resultattallet etter en " & "-operasjon har 1 i de posisjonene hvor begge tallene har 1, men nuller i de övrige

Denne operasjonen brukes særlig ved ekstrahering av visse sifre i et ord. Hvis en bare er interessert i sifrene 9-13 i et ord f.eks., kan en skille ut disse ved å foreta logisk multiplikasjon av tallet med et annet tall som har enere i sifrene 9-13, men forövrig nuller. Resultatet vil da være nettopp de önskede sifre.

III. MINNET

1. Hurtig-minnet

I figuren i kap. I ser en hvordan hurtig-minnet er den sentrale delen innen DEUCE's logiske oppbygning. Som för nevnt lagres alle tall og instruksjoner i DEUCE som en serie, hver med 32 elektriske impulser; en slik serie kalles et ord. Lagringen av disse ordene i hurtig-minnet er basert på d e l a y - l i n e s, dvs. en strömkrets med et forsinkelses-element. Dette forsinkelses-elementet i DEUCE består av et kvikksölvrör med én inngang og én utgang. Figur 2 viser en skjematisk tegning av en slik delay-linje.

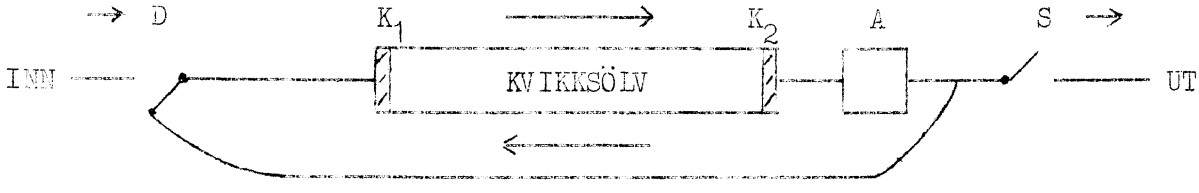


Fig. 2

I hver ende av kvikksölvröret er det en kvartskrystall, K_1 og K_2, som virker mot en membran. Når en elektrisk impuls blir sendt til K_1 vil dette resultere i en mekanisk svingning (ikke hörbar lydbölge) som forplanter seg som en bölge i kvikksölvet. Etter en viss tid, avhengig av kvikksölvrörets lengde, vil denne bölgen nå K_2 hvor den blir omsatt til elektrisk impuls igjen. Impulsen vil imidlertid bli svekket noe i kvikksölvet, derfor blir den forsterket i A. Hvis så denne impulsen sendes direkte inn i K_1 igjen, vil den sirkulere i kretsen og komme ut fra K_2 med faste mellomrom.

I DEUCE er tidsintervallet mellom impulsene i en serie 1 mikro-sekund (1 mikro-sek. = 1/1000000 sek.). Med en lengde på kvikksölvröret som gjör at forsinkelsen eller tiden en bölge trenger fra K_1 til K_2 er 32 mikro-sek., vil et ord à 32 bits kunne lagres i form av et bölge-tog i kvikksölvröret.

Når et ord skal lagres i en slik delay-linje, må den elektroniske bryteren D bringes i en slik stilling at impuls-serien kan komme til K_1. 31 mikro-sekunder etter at den förste impulsen når K_1 vil den siste impulsen i serien komme dit. Like etter at denne siste impulsen er kommet fram, legges bryteren D om igjen, slik at når den förste impulsen i serien kommer til K_2 ett mikro-sek. senere, vil den öyeblikkelig bli sendt til K_1 igjen.

På denne måten vil de 32 impulsene i serien "sirkulere" i delay-linjen så lenge bryteren D er i samme stilling som på figuren.

Skal innholdet i delay-linjen overföres til et annet sted i maskinen, må bryteren S settes på like för den förste impulsen i serien ventes fra K_2. Når siste impuls i serien har passert K_2, legges bryteren tilbake i normal-stilling igjen. Slik kan innholdet i en delay-linje "leses ut", men fortsatt vil det samme impuls-toget sirkulere i delay-linjen.

Så lenge det er ström på maskinen og bryterstillingen i D ikke forandres, vil et ord sirkulere i det uendelige i en delay-linje. Den siste impulsen i et ord vil alltid bli etterfulgt av den förste impulsen i ordet uten noen indikasjon for hvor ordet begynner og slutter. For å kunne holde rede på dette, er det i DEUCE en spesiell impuls-serie hvor tidsintervallet mellom impulsene er 32 mikro-sek. Disse impulsene kan en tenke seg kommer like för den förste impulsen i et ord, dvs. i tidsrommet mellom siste og förste bit. Da alle ordene innen DEUCE opptrer "i takt" dvs. det förste bit opptrer på samme tid for alle ordene, brukes denne spesielle impuls-serien til å kontrollere operasjonene i DEUCE på den måten at alle operasjoner starter og slutter på tidspunkt som markeres av en av disse kontroll-impulsene. Tidsintervallet på 32 mikro-sekunder mellom kontroll-impulsene kalles en "minor-cycle" eller "m.c.".

Det er mange delay-linjer i hurtig-minnet i DEUCE, hver med sitt eget forsinkelses-element. Disse forsinkelses-elementene er ikke like for alle, på samme måte som vi kan lagre 32 bits eller et ord i en delay-linje med en forsinkelse på 32 mikro-sekunder, vil 2 ord eller 64 bits kunne lagres hvis forsinkelsen er 64 mikro-sek. På tilsvarende måte vil en delay-linje med 1024 mikro-sekunders forsinkelse kunne inneholde 32 ord eller 1024 bits. I en slik delay-linje vil de 32 ordene sirkulere etter hverandre og komme ut fra K_2 i rekkefölge. Önsker en å erstatte ett av de 32 ordene med et nytt, må en ved hjelp av bryteren D sörge for at forbindelsen mellom K_2 og K_1 blir brutt i den minor-cycle det ordet som skal erstattes kommer ut av K_2. Samtidig vil det nye ordet ha adgang til K_1. Etter at hele ordet er kommet til K_1, etter en minor-cycle, forbindes K_2 og K_1 igjen. Denne operasjonen vil ikke beröre de övrige 31 ordene i delay-linjen.

Hurtigminnet i DEUCE består av 22 delay-linjer som er delt opp i 4 grupper.

a) Korte registre (Temporary Stores)

Det er fem av disse delay-linjene, som hver kan lagre et ord:

TS COUNT er en del av kontroll-organet, som fortolker og adlyder

instruksjonene. Denne delay-linjen er ikke tilgjengelig for lagring av tall.

De övrige er TS 13, TS 14, TS 15 og TS 16.

b) Dobbelte registre (Double-length Stores)

Det er tre av disse, som hver kan inneholde to ord. De er DS 19, DS 20 og DS 21.

c) 4-ords registre (Quadruple Stores)

Det er to av disse, som hver kan inneholde fire ord. De er QS 17 og QS 18.

d) Delay-linjer (Delay Lines)

Det er tolv av disse, hver kan lagre 32 ord. De er nummerert fra DL 1 til og med DL 12.

Alle tall vil vanligvis lagres i delay-linjene (DL) og flyttes over til de kortere registrene når de skal være med i en regneoperasjon.

Så lenge det er ström på DEUCE, vil serien av kontrollimpulser stadig være tilstede med et tidsintervall på 32 mikro-sekunder mellom impulsene. En kan si at disse kontrollimpulsene "deler" tiden opp i minor-cycles. For enkelthets skyld er disse minor-cycles nummerert fra 0 til 31 og så begynner det på 0 igjen. Vi antar at vi sender et ord til DL 1 i m.c. 5, idet vi bryter forbindelsen mellom K_2 og K_1 i nettopp denne minor-cycle. Hvert siffer i dette ordet vil da komme ut av K_2 igjen 1024 mikro-sek. etter at det ble sendt til K_1. I andre ord kan en si at hele ordet vil komme ut av K_2 i den 32 $\frac{te}{}$ minor-cycle etter den minor-cycle som ordet ble sendt inn i. Helt til ordet blir erstattet vil det komme ut av K_2 hver 32 $\frac{te}{}$ minor-cycle. Alle disse minor-cycles som ordet kommer ut av K_2, har nummeret 5. Vi sier at ordet er lagret i "DL 1, m.c. 5" eller ganske enkelt i "1_5". Vi kaller dette adressen til ordet. De 32 ordene som er lagret i DL 1 vil da ha adressene 1_0, 1_1, 1_2 osv. inntil 1_{31}.

Adresseringen av ordene i QS og DS er ikke så enkel. QS 17 kan som nevnt inneholde 4 ord, hvis vi derfor sender et ord til QS 17 i m.c. 21 vil dette komme ut igjen 4 minor-cycles senere, i m.c. 25, deretter i m.c. 29, så i m.c. 1 osv. Den laveste m.c. (m.c. 1) i listen blir brukt som adresse til dette ordet i QS 17, en sier altså at adressen er "QS 17, m.c. 1" eller enkelt "17_1". Ordene i QS 17 får altså disse adressene: 17_0, 17_1, 17_2 og 17_3. Ordene i QS 18 adresseres på tilsvarende måte.

De to ordene i en DS sier en er lagret i minor-cycle 2 og 3, adressene vil da bli 19_2 og 19_3, 20_2 og 20_3, 21_2 og 21_3. Ordene i 19_2, 20_2 og 21_2 vil være tilgjengelige i m.c. 0, m.c. 2, m.c. 4 osv. kort sagt alle

like minor-cycles. Ordene i 19_3, 20_3 og 21_3 vil være tilgjengelige i alle ulike minor-cycles.

I de periodene hvor alle delay-linjene er lukket, dvs. forbindelsen mellom K_2 og K_1 er tilstede, vil alle delay-linjene være i nöyaktig lik tilstand i intervaller på 32 minor-cycles, eller forsinkelses-tiden i en lang delay-linje. En slik periode på 32 minor-cycles eller 1024 mikro-sekunder kalles en "major-cycle" eller "M.c."

En benytter seg gjerne av et begrep som kalles v e n t e t i d ("latency") for å karakterisere hvor hurtig minnet i en elektronisk maskin er. Denne ventetiden kan defineres som den tiden en må vente fra en beordrer overföring av et bestemt ord i minnet og til denne overföringen kan starte. De korte registrene i DEUCE er som nevnt tilgjengelige i alle minor-cycles slik at overföring av ord innen disse registrene går uten ventetid. I de dobbelte registrene derimot vil bare det ene ordet være tilgjengelig umiddelbart (ventetid = 0), det andre ordet vil först bære tilgjengelig en minor-cycle senere (ventetid = 1 m.c.). For slike registre hvor ventetiden varierer fra ord til ord, benytter en som regel d e n g j e n n o m s n i t t l i g e v e n t e t i d e n . Denne vil være 0,5 m.c. eller 16 mikro-sek. for de dobbelte registrene. I 4-ords registrene vil ventetiden til ordene være: 0, 1, 2 og 3 minor-cycles, altså en gjennomsnittlig ventetid på 1,5 minor-cycles eller 48 mikro-sek. På tilsvarende måte er gjennomsnitts-ventetiden for delay-linjene (DL) 15,5 minor-cycles eller 496 mikro-sekunder.

I följende tabell har en fört opp de viktigste data for hurtig-minnet:

		Antall ord	Gjennomsnittlig ventetid i mikro-sek.
4 korte registre	(TS)	4	0
3 dobbelte registre	(DS)	6	16
2 4-ords registre	(QS)	8	48
12 delay-linjer	(DL)	384	496
		402	

2. Magnet-trommel

I tillegg til hurtig-minnet er DEUCE utstyrt med en magnetisk trommel. Denne er som navnet tilsier, en metall-sylinder med magnetiserbar overflate. Denne magnet-trommelen er i stadig rotasjon, og roterer med omdreiningshastighet på èn omdreining pr. 9 major-cycles, som tilsvarer ca. 6500 omdreininger pr. minutt. Magnet-trommelen er delt opp i 256 spor, hvert spor kan lagre 32 ord à 32 bits. Et spor kan altså lagre akkurat like mange ord som en DL.

Selve lagringen av ord på trommelen gjöres ved hjelp av et skrive-hode som er plasert meget nær trommelen, og som magnetiserer "flekker" på overflaten etter hvert som trommelen roterer. En binær ener vil således resultere i en magnetisk "flekk" hvor nord-polen peker i én retning, mens en binær null vil magnetisere en "flekk" med nord-polen i den andre retningen. På samme måte hentes ordene fra trommelen igjen ved et lese-hode, som avleser den ene magnetiske retningen som enere og den andre som nuller. Også for magnet-trommelen er det slik at ved "skriving" av et ord på trommelen blir det gamle resultatet borte, mens ved lesing vil ordet fortsatt stå på trommelen uforandret.

Hvis et lese-hode og et skrive-hode skulle betjene alle de 256 sporene, ville mye tid gå med til flytting av hodene fra spor til spor. Det beste ville naturligvis være å ha et lesehode og et skrivehode til hvert spor. I DEUCE har en tatt "midt på treet" idet det er 16 skrive-hoder og like mange lese-hoder. Disse 16 lese- hodene eller 16 skrive-hodene er ikke individuelt flyttbare, men alle 16 må flyttes på en gang. Det er imidlertid ingen forbindelse mellom lese- og skrive-hodene. På grunn av at hodene kan dekke 16 spor om gangen, deler en gjerne magnet-trommelen inn i 16 blokker à 16 spor. Lese- og skrive-hodene vil da kunne dekke en blokk om gangen (det er imidlertid ikke nödvendig at lese- og skrive-hodene dekker samme blokk).

Som nevnt er det 256 spor på trommelen, disse kan da nummereres fra 0 til 255, men på grunn av denne blokkinndelingen vil det ofte være vel så greit å nummerere blokkene fra 0 til 15 og sporene innen blokkene fra 0 til 15, slik at adressen kan angis som "a/b", hvor "a" representerer blokk nummer og "b" står for spor-nummer innen blokken.

Eksempel:

spor	nr.	0	vil	ha	adressen	0/0
"	"	16	"	"	"	1/0
"	"	22	"	"	"	1/6
"	"	255	"	"	"	15/15

Alle overföringer av ord til og fra magnet-trommelen skjer alltid i grupper av 32 ord, det er altså ikke mulig å "lese" bare ett ord eller "skrive" ett ord på trommelen. Det er dessuten alltid DL 11 i hurtigminnet som er forbindelsesleddet til trommelen, alle opplysninger som skal inn på eller ut fra trommelen må gå via DL 11.

Det er derfor fire operasjoner i maskinen som gjelder magnet-trommelen:

a) Flytt lese-hodene til en bestemt blokk
b) Flytt skrive-hodene til en bestemt blokk
c) Overför innholdet av et bestemt spor til DL 11
d) Overför innholdet av DL 11 til et bestemt spor på magnet-trommelen.

De to förste operasjonene behöver ikke å gjöres i de tilfellene en vet at lese-hodene eller skrive-hodene står i riktig posisjon. Alle fire operasjonene trenger relativt lang tid. Flytting av lese- og skrive-hodene tar 35 millisek. og overföring av et spor til eller fra trommelen tar 13 millisekunder. Disse operasjonene kan imidlertid foregå samtidig med andre operasjoner, bare disse ikke berörer trommelen eller DL 11.

IV. KONTROLL-ORGAN

For å löse en oppgave på DEUCE må oppgaven splittes opp i en kortere eller lengere serie av elementæroperasjoner som maskinen kan utföre automatisk. Disse elementæroperasjonene (ordrene) kan deles i följende grupper:

a) Intern data-overföring
b) Addisjon, subtraksjon, multiplikasjon og divisjon
c) Skift
d) Visse logiske operasjoner
e) Test-operasjoner
f) Innlesing og punching av kort

Arbeidet med å tilrettelegge oppgaven og bryte den ned til elementær-operasjoner, kaller vi p r o g r a m m e r i n g . Resultatet av programmeringen er et program.

Programmet må lagres inne i maskinen. Det forekommer ikke noen spesiell lagerplass for programmet, derfor må dette lagres på samme sted og på samme måte som de data DEUCE skal bearbeide. Vi kjenner til at alle tall som lagres på disse stedene, i hurtig-minnet eller på magnet-trommelen, opptrer som ord à 32 bits. De enkelte elementær-operasjoner i programmet må skrives ned i en form som maskinen kan forstå, dvs. ved hjelp av tallkoder. En slik linje (elementæroperasjon) kaller vi en i n s t r u k s j o n eller o r d r e . Hver ordre består av 32 bits og okkuperer derfor et ord i DEUCE. Arbeidet med å omforme programmet til et språk maskinen vil kunne forstå, kalles k o d i n g .

Den delen av DEUCE som sörger for å utföre programmet korrekt, kalles kontroll-organet. En del av dette er TS COUNT, som er et kort register og derfor kan lagre et ord. Ordrene i programmet blir hentet inn i dette registret. Kontroll-organet vil analysere ordren i TS COUNT og sörge for at den blir utfört til riktig tid. Kontroll-organet sörger også for at neste ordre i programmet blir hentet inn i TS COUNT.

En ordre i maskinen består som nevnt av et ord à 32 bits. Disse bits er delt opp i 8 forskjellige ordre-deler, som har sin bestemte plass i ordren og sin egen bestemte funksjon. Noen ganske få bits i ordren er vanligvis ikke i bruk. En vil i det följende ganske kort forklare litt om de enkelte ordre-delers funksjoner, i senere kapitler vil en komme mer detaljert inn på disse.

N e x t I n s t r u c t i o n S o u r c e (NIS)

Denne ordredelen angir i hvilken DL neste ordre er å finne. NIS er på 3 bits, derfor vil den kunne anta verdiene 0 - 7. NIS "0" betyr DL 8, mens NIS 1-7 betyr DL 1-7, slik at dette begrenser det området hvor neste ordre er å finne, nemlig til de 8 første DL.

S o u r c e (S) o g D e s t i n a t i o n (D)

Disse to ordre-delene sammen spesifiserer den bestemte operasjon som skal utföres. Både S og D kan ha verdier fra 0 til 31, da de begge er på 5 bits. 21 av S-verdiene og like mange D-verdier har tilknytning til de registrene i hurtig-minnet som har samme nummer (DL 1-12, TS 13-16, QS 17-18, DS 19-21). De övrige 11 S-verdier og 11 D-verdier benyttes til forskjellige andre formål, f.eks. addisjon, subtraksjon, lesing og punching i kort, overföring til og fra magnet-trommel osv. Det er vanligvis ingen spesiell relasjon mellom en S og D av samme verdi.

Bortsett fra de 21 förste verdiene er det ingen relasjon mellom S og D av samme verdi. S-verdiene 27-31 brukes til å skaffe 5 nyttige konstanter, en kan betrakte disse 5 S-verdiene som et tillegg til hurtig-minnet på 5 korte registre som en ikke kan forandre innholdet av.

C h a r a c t e r i s t i c s (C)

Denne ordre-delen avgjör hvor lenge operasjonen skal vare på följende måte:

0 = operasjonen skal vare i en minor-cycle
1 = operasjonen kan vare flere enn to minor-cycles, inntil en major-cycle (32 minor-cycles)
2 = operasjonen skal vare i to minor-cycles
3 = ikke brukt

En kan si at C betegner kort (C=0), dobbelt (C=2) eller lang (C=1) operasjon.

W a i t n u m b e r (W)

W spesifiserer når operasjonen skal begynne, eller rettere sagt hvor mange minor-cycles kontroll-organet må vente för operasjonen kan starte.

J o e n u m b e r

Denne delen av ordren ignoreres av kontroll-organet og har ingen betydning for operasjonen. Det vil senere bli forklart når og hvordan denne ordredelen blir brukt.

T i m e n u m b e r (T)

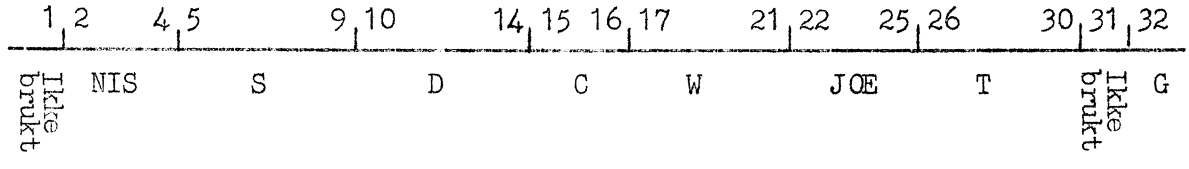
T spesifiserer hvor mange minor-cycles kontroll-organet skal vente för neste ordre blir hentet inn i TS COUNT. Noen ganger vil T dessuten angi siste minor-cycle i en operasjon, dvs. når operasjonen skal slutte.

G o d i g i t (G)

Hvis G = 0, er ordren en stopp-ordre, slik at maskinen vil stoppe og vente på en bestemt impuls för ordren blir utfört, hvis G = 1, vil operasjonen bli utfört på vanlig måte uten stopp.

Bit nr. 1 og 31 blir vanligvis ikke brukt og ignoreres av kontroll-organet.

Plaseringen av ordre-delene innen ordren er tegnet opp i figur 3. Sifrene i et ord i DEUCE nummereres fra 1 til 32, og betegnes P 1, P 2 P 3, osv. P 1 er laveste bit, dvs. venstre, mens P 32 er höyeste bit (höyre).



Ordre-ordet

Fig. 3

Ordre-delene NIS (ofte kalt bare N), S og D kalles ofte a d r e s s e - d e l e n i ordren. Så snart en ny ordre hentes inn i TS COUNT, vil kontroll-organet undersöke adressedelen og velge ut de riktige NIS, S og D forbindelser. Forbindelsen mellom de valgte S og D, og mellom den utvalgte NIS og TS COUNT, vil imidlertid ikke bli opprettet för i de minor-cycles som blir bestemt av de övrige ordre-deler. Ordren vil bli analysert og forbindelsene valgt i förste halvpart av en minor-cycle, som kalles "set-up minor-cycle". I denne minor-cycle, som följer umiddelbart etter den minor-cycle som ordren blir lest inn i TS COUNT, må der ikke forekomme andre operasjoner. Ordren må naturligvis hentes inn i TS COUNT i samme minor-cycle som den er lagret i en DL.

Ordre-delene C, W og T kalles for ordrenes " t i m i n g s e c t i o n ", og virker på följende måte:

W a i t : W angir som för nevnt hvor mange minor-cycles kontroll-organet skal vente för operasjonen skal starte. Den tidligste minor-cycle en operasjon kan starte i er den andre minor-cycle etter den minor-cycle som ordren blir lest inn i TS COUNT. Hvis en ordre blir lest inn i TS COUNT i minor-cycle nr. m, vil m.c. nr. (m+1) være "set-up minor-cycle",

slik at m.c. (m+2) vil være den første minor-cycle operasjonen kan starte i. Hvis W = 0, dvs. ingen venting, vil operasjonen starte i m.c. (m+2); hvis W = 1, vil operasjonen starte i m.c. (m+3) osv. Generelt kan en si at operasjonen vil starte i m.c. (m+2+W).

T i m e : T angir alltid når neste ordre skal hentes inn i TS COUNT, i noen tilfelle (C = 1) vil T også bestemme når operasjonen skal slutte. Vi antar at en ordre blir hentet inn i TS COUNT i m.c. (m), neste minor-cycle, m.c. (m+1), er "set-up minor-cycle", minor-cyclen deretter, m.c. (m+2) vil altså være den første minor-cycle som operasjonen kan starte i og samtidig, hvis denne minor-cycle også er den siste i operasjonen, vil denne minor-cycle være den første som neste ordre kan hentes inn i TS COUNT. Det er nemlig intet til hinder for at neste ordre kan hentes inn i TS COUNT og at operasjonen i foregående ordre blir utført i samme minor-cycle, forutsatt at denne minor-cycle er den eneste eller siste i operasjonen. Hvis T = 0, vil neste ordre bli hentet inn i TS COUNT i m.c. (m+2); hvis T ≥ 1, vil neste ordre bli hentet inn i m.c. (m+2+T).

Normalt vil T være lik eller større enn W, i de tilfelle da W er større enn T må en være oppmerksom på følgende: Det er klart at en ordre ikke henter inn den neste ordren før operasjonen i ordren er utført, derfor er det ordnet slik at når W > T, vil verdien av T bli økt med 32, slik at neste ordre blir hentet inn i TS COUNT i m.c. (m+34+T) istedetfor som vanlig i m.c. (m+2+T).

Det samme vil være tilfelle hvis W=T og C=2, som vi senere skal se, også da blir T økt med 32, slik at neste ordre blir hentet inn i m.c. (m+34+T).

C h a r a c t e r i s t i c s (C): Som nevnt før angir denne ordre-delen operasjonens lengde, på denne måten:

Kort operasjon (C = 0): I dette tilfelle vil operasjonen bli utført i en minor-cycle, nemlig m.c. (m+2+W). Hvis T = W, vil neste ordre bli hentet inn i TS COUNT i samme minor-cycle, dvs. det blir ingen ventetid fra utførelsen av operasjonen og til neste ordre blir hentet inn. I ordre med C = 0 får en kortest mulig ventetid til operand og neste ordre hvis en kan sette W = T = 0, da vil hele ordren og innlesingen av neste til TS COUNT ta bare 2 minor-cycles.

Dobbelt operasjon (C = 2): Ved en dobbelt operasjon vil operasjonen bli utført i m.c. (m+2+W) og (m+3+W). For å få minst mulig ventetid til innlesingen av neste ordre må en ha T = W + 1. Hvis W = T, vil som før nevnt, T bli økt med 32 slik at neste ordre blir hentet inn i TS COUNT i m.c. (m+34+T). I ordre med C = 2 får en kortest mulig ventetid til operand og neste ordre hvis en kan sette W = 0 og T = 1.

Lang operasjon (C = 1): I dette tilfelle vil operasjonen begynne i m.c. (m+2+W) og vare til og med m.c. (m+2+T). Hvis W = T, varer operasjonen i en minor-cycle, er W+1 = T vil operasjonen vare i to osv. Generelt vil operasjonen vare i (T+1-W)minor-cycles, hvis W≤T og i (T+33-W)minor-cycles hvis W>T. En operasjon kan altså aldri vare lenger enn 32 minor-cycles.
C = 3: Dette er ikke brukt, Hvis en imidlertid skulle komme til å bruke denne C verdien, bör en være oppmerksom på virkningene. Hvis W er forskjellig fra T, vil C = 3 ha akkurat samme virkning som C = 1 (lang operasjon). Hvis W = T, vil T bli økt med 32. Operasjonen vil da vare i 33 minor-cycles fra m.c. (m+2+W) til og med m.c. (m+34+W) og neste ordre vil også bli hentet inn i m.c. (m+2+W).

V. "SOURCES" OG "DESTINATIONS"

Som nevnt arbeider DEUCE ved å utföre en serie av ordrer i en spesiell rekkefölge. I det foregående kapitel er de enkelte delene innen ordren kort beskrevet. En skal i dette kapitel gå litt nærmere inn på de to ordredelene som spesifiserer operasjonen, nemlig Source (S) og Destination (D).

Både S og D har 5 bits og kan derfor anta alle verdier fra 0 til 31. 21 S-verdier og like mange D-verdier angir lagerposisjonene i hurtig-minnet, nummeret på lagerposisjonen tilsvarer dens S- og D-verdier. En har for eksempel at S 4 og D 4 begge angår DL 4, S 14 og D 14 angår TS 14 osv. De övrige 11 S-verdier og tilsvarende D-verdier benyttes til andre formål, og for disse er det ingen spesiell forbindelse mellom S og D av samme verdi.

Ved programmering skriver en som regel ordrene i denne form, "S - D", altså med en strek mellom S og D. En skal senere se hvilke symboler som benyttes for å angi operasjonens varighet osv. En skal i det följende se litt på de forskjellige verdiene av S og D, og ved enkle eksempler forsöke å forklare hvordan forskjellige kombinasjoner av S og D kan brukes.

I n t e r n o v e r f ö r i n g

I sin enkleste form spesifiserer en ordre bare en transport eller overföring av et eller flere ord innen hurtig-minnet. Det er mulig å foreta overföring fra en hvilken som helst og til en hvilken som helst delay-linje, og det er mulig å overföre inntil 32 ord ved hjelp av en ordre. I slike ordrer vil S og D bare kunne ha verdier fra 1 til 21, fordi dette er nummerne på de lagerposisjonene i hurtig-minnet som er tilgjengelig for programmereren. Ordren "13 - 15" vil erstatte innholdet i TS 15 med innholdet i TS 13, TS 13 forblir uforandret.

Eksempler:

a) $14 - 21_2$
b) $11_4 - 19_2$
c) $8_{17} - 18_1$
d) $18_2 - 12_{14}$

Merknader:

a) Innholdet i TS 14 vil kunne overföres i begge minor-cycles i DS 21, fordi TS 14 er tilgjengelig i alle minor-cycles. En må derfor spesifisere hvilket ord i DS 21 det gjelder ved å sette D = 21_2.

b) Ordet i DL 11_4 kan bare overföres til den like minor-cycle i DS 19, derfor er det ikke strengt tatt nödvendig å sette D = 19_2, men det gjöres gjerne for oversiktens skyld.

c) Ordet i DL 8_{17} vil bare kunne erstatte det ordet som befinner seg i m.c. 1 i QS 18, fordi det er QS 18, som er tilgjengelig i m.c. 17.

d) Ordet i QS 18_2 kan erstatte 8 av ordene i DL 12, nemlig DL 12_2, DL 12_6, DL 12_{10} osv., derfor må en spesifisere hvilken minor-cycle operasjonen skal utföres i.

Som en ser av eksemplene foran, må en være oppmerksom på når de forskjellige ordene i hurtig-minnet er tilgjengelig. For å lette oversikten er det i fig. 4 en skjematisk framstilling av dette.

Minor-cycles: |0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,
20,21,22,23,24,25,26,27,28,29,30,31|

DL : |0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,
20,21,22,23,24,25,26,27,28,29,30,31|

QS : |0,1,2,3|0,1,2,3|0,1,2,3|0,1,2,3|0,1,2,3|0,1,2,3|0,1,2,3|0,1,2,3|
|0,1,2,3|0,1,2,3|0,1,2,3|0,1,2,3|

DS : |2,3|2,3|2,3|2,3|2,3|2,3|2,3|2,3|2,3|2,3|2,3|2,3|2,3|2,3|2,3|2,3|
|2,3|2,3|2,3|2,3|2,3|2,3|2,3|2,3|

Fig. 4

En har ikke tatt med de korte registrene i denne figuren da disse, som kjent, er tilgjengelig i alle minor-cycles.

Ved eksemplene foran var det bare overföringer av enkle ord, det er imidlertid også mulig å overföre flere ord (inntil 32) ved en ordre. (I ett enkelt tilfelle er det endog mulig å overföre 33 ord ved en ordre).

Eksempler:

a) 19 - 21 d

b) 9_{12} - 17_0 d

c) 17 - 18 l (4 m.c.)

d) 2 - 4 l (32 m.c.)

e) 8_{13} - 17_1 l (3 m.c.)

f) 8_1 - 21 d

21 - 8_{10} d

Merknader:

a) Symbolet "d" i denne ordren markerer at det er en dobbelt operasjon (C = 2). Operasjonen varer i to minor-cycles slik at begge ordene i DS 19 vil bli overført til DS 21.

b) Også dette er en dobbelt operasjon, innholdet av DL 9_{12} vil erstatte innholdet i QS 17_0 og innholdet i DL 9_{13} vil erstatte innholdet i QS 17_1.

c) "1" i en ordre betegner at operasjonen er lang, varigheten angis i parenteser. I dette eksemplet skal operasjonen vare i 4 minor-cycles, dvs. at innholdet i hele QS 17 vil erstatte innholdet i QS 18. Det spiller ingen rolle i hvilken minor-cycle denne operasjonen starter.

d) Dette er en lang operasjon og varer i hele 32 m.c. eller 1 major-cycle, slik at innholdet i hele DL 2 vil erstatte innholdet i DL 4. Her spiller det heller ikke noen rolle når operasjonen starter.

e) Denne ordren erstatter innholdet i QS 17_{1-3} med innholdet i DL 8_{13-15}.

f) Hvis en ønsker å overföre et ord fra en minor-cycle i en delay-linje til en annen minor-cycle i samme eller en annen delay-linje, må dette gjöres i to operasjoner med mellomlagring i en kortere delay-linje. Legg merke til at her blir DL 8_1 overført til DS 21_3 og derfra til DL 8_{11}, mens DL 8_2 går via DS 21_2 til DL 8_{10}. Ordene i DL 8_{1+2} blir altså lagret i omvendt orden i DL 8_{10+11}.

K o n s t a n t e r

Noen av S-verdiene angir nyttige konstanter:

S 27 - P_1 (ener i laveste bit)
S 28 - P_{17} (ener i laveste bit i "Wait")
S 29 - P_{32} (ener i höyeste bit)
S 30 - nuller
S 31 - enere (etter fortegnsmetoden representerer dette -1)

Dette kan betraktes som en utvidelse av hurtig-minnet med 5 ord. Innholdet i disse ordene kan overföres til hvor som helst innen hurtig-minnet, men det er ikke mulig å forandre innholdet i disse ordene da en ikke kan overföre noe til dem.

Eksempler:

a) Töm TS 14:

30 - 14

b) Töm hele DL 9:

30 - 9 1 (32 m.c.)

A d d i s j o n o g s u b t r a k s j o n

Addisjon og subtraksjon kan bare utföres på to steder i hurtigminnet, nemlig i TS 13 og DS 21. Det er fire D-verdier som betegner disse operasjonene:

D 25 - addisjon i TS 13

D 26 - subtraksjon i TS 13

D 22 - addisjon i DS 21

D 23 - subtraksjon i DS 21

Disse fire D-verdiene angir ikke i förste rekke en adresse, men en bestemt funksjon. Ordren "16 - 25" adderer innholdet i TS 16 til innholdet i TS 13 og plaserer summen i TS 13. TS 16 forblir uforandret.

Eksempel:

To tall som befinner seg i TS 14 og TS 15 skal adderes og summen skal plaseres i TS 16. En får:

14 - 13

15 - 25

13 - 16

Disse tre ordrene sammen danner et program.

Det er intet i veien for å ha "13 - 25", dvs. at innholdet i TS 13 adderes til seg selv, slik at det blir fordoblet. I binær-systemet betyr dette at innholdet i TS 13 skiftes eller flyttes en posisjon til höyre eller opp. Lar en denne ordren vare i to minor-cycles, slik: "13 - 25 d", får en följende resultat: I förste minor-cycle blir innholdet i TS 13 addert til seg selv, dvs. fordoblet, i neste minor-cycle blir så dette nye innholdet i TS 13 fordoblet igjen slik at det endelige resultat av ordren blir at det opprinnelige innholdet i TS 13 firedobles, dvs. multipliseres med 2^2 eller skiftes 2 posisjoner opp. Lar en ordren vare i tre minor-cycles, "13 - 25 1 (3 m.c.)", blir innholdet i TS 13 multiplisert med 2^3 eller skiftet 3 posisjoner opp. Generelt har en denne regel:

Når en önsker å skifte innholdet i TS 13 "n" posisjoner opp (til höyre), oppnås dette ved ordren "13 - 25 1 ("n"m.c.)".

Ordren "13 - 26" bevirker at innholdet i TS 13 blir subtrahert fra seg selv, dvs. TS 13 blir nullstillet eller "tömt". Denne ordren er det ingen grunn til å la vare flere minor-cycles, fordi resultatet blir det

samme. Ordren "13 - 26 d" f.eks., har denne virkning: TS 13 blir nullstillet i förste minor-cycle, og det samme gjentar seg i neste minor-cycle.

Addisjon og subtraksjon i DS 21 er avhengig av hvorvidt innholdet i DS 21 betraktes som ett langt ord å 64 bits eller to atskilte ord å 32 bits. Dette dirigeres av en elektronisk bryter som kalles "TCB". Normalt, dvs. når TCB er "av", blir innholdet i DS 21 behandlet som et langt ord å 64 bits med höyeste halvpart i DS 21_3. Når TCB er "på", derimot, behandles ordene i DS 21 som to atskilte ord, hver med 32 bits.

Eksempler:

a) Adder de to lange tallene i DL 8_{0+1} og DL 8_{10+11} og plaser resultatet i DL 12_{4+5}.

TCB "av"

8_0 - 21 d

8_{10} - 22 d

21 - 12_4 d

b) Subtraher innholdet i DL 2_9 fra innholdet i DL 2_3 og plaserer resultatet i DL 2_{15}.

TCB "på"

2_3 - 21_3

2_9 - 23_3

21_3 - 2_{15}

En kunne naturligvis her ha benyttet TS 13 istedetfor DS 21_3, men ikke DS 21_2.

Ordren "21 - 22" vil addere innholdet av et av ordene i DS 21 til seg selv; utföres operasjonen i en like minor-cycle, blir DS 21_2 fordoblet, er det en ulike minor-cycle, er det DS 21_3 som fordobles. Lar en ordren vare i to minor-cycles "21 - 22 d", blir begge ordene i DS 21 fordoblet. En kan da sette opp en tilsvarende regel for skifting opp i DS 21 som den som er satt opp for TS 13:

> Når en önsker å skifte innholdet i DS 21 "n" posisjoner opp (til höyre), oppnås dette ved ordren "21 - 22 l ("2n"m.c.).

Har en ordren "21 - 22 l ("2n+1" m.c.)", dvs. et ulike antall minor-cycles, resulterer dette i at bare ett av ordene i DS 21 blir skiftet "n" posisjoner opp, mens det andre ordet blir skiftet "n+1" posisjoner opp. Hvorvidt dette siste ordet er DS 21_2 eller DS 21_3, avhenger av om operasjonen starter i en like eller ulike minor-cycle.

DS 21 nullstilles ved ordren "21 - 23". Varer denne ordren bare en minor-cycle, blir et av ordene i DS 21 nullstillet. Hvis ordren varer to eller flere minor-cycles, blir hele DS 21 nullstillet.

Eksempler:

1) Summer sammen tallene i hvert fjerde ord i DL 10 og plaser resultatet i DL 12_0

a)
```
10_0  - 13
10_4  - 25
10_8  - 25
10_12 - 25
10_16 - 25
10_20 - 25
10_24 - 25
10_28 - 25
13    - 12_0
```

b)
```
10_0  - 21_2
10_4  - 22_2
10_8  - 22_2
10_12 - 22_2
10_16 - 22_2
10_20 - 22_2
10_24 - 22_2
10_28 - 22_2
21_2  - 12_0
```

Som en ser kan en summere de 8 tallene i TS 13 eller DS 21_2, men derimot ikke DS 21_3, fordi denne og de 8 tallene ikke er tilgjengelige i samme minor-cycle.

2) Regn ut "x = a+2b+3c", hvor "a" står i DL 8_0, "b" står i DL 8_1, "c" står i DL 8_2 og "x" skal lagres i DL 8_3.

a)
```
8_0 - 13
8_1 - 25
8_1 - 25
8_2 - 25
8_2 - 25
8_2 - 25
13  - 8_3
```

b)
```
8_0 - 13
8_1 - 25 d
8_1 - 25 d
8_2 - 25
13  - 8_3
```

c)
```
8_0  - 13
8_1  - 21 d
21_2 - 25 l (5 m.c.)
13   - 8_3
```

Det er mange flere måter å löse denne oppgaven på. Den enkle "rett-fram" lösningen i a) trenger hele 7 ordre og vil, som vi skal se nærmere på under kodingen, være en langsom lösning. Lösning b) har den fordelen framfor c) at bare TS 13 benyttes ved addisjonen. Lösning c) er imidlertid den raskeste. Ordren "8_1 - 21 d" plaseres "b" i DS 21_3 og "c" i DS 21_2. Neste ordre varer i 5 m.c., dette resulterer i at innholdet i DS 21_2 blir addert tre ganger og innholdet i DS 21_3 blir addert to ganger inn i TS 13.

3) Regn ut "x = 2b - d", hvor "b" finnes i DL 7_{14}, "d" i DL 7_{15} og "x" skal plaseres i DL 7_{16}.

```
7_14 - 13
13   - 25
7_15 - 26
13   - 7_16
```

S k i f t

Ved å bruke noen spesielle S-verdier er det mulig å skifte eller flytte innholdet av visse lagerposisjoner.

S 22 - innholdet i DS 21 skiftes til venstre, dvs. divideres med 2.

S 23 - innholdet i TS 14 skiftes til venstre, dvs. divideres med 2.

S 24 - innholdet i TS 14 skiftes til höyre eller multipliseres med 2.

S 22, S 23 og S 24 sörger bare for at skiftet finner sted, men bestemmer ikke hvor resultatet skal plaseres, dette avgjöres av D-verdien i ordren. Resultatet av et skift kan plaseres hvor som helst i hurtigminnet.

I ordrer med S 23 og S 24, dvs. skifting av innholdet i TS 14, blir ikke TS 14 forandret, unntatt i de tilfeller da ordrene har D = 14. Varer en ordre med S 23 eller S 24 en minor-cycle, blir innholdet i TS 14 skiftet en posisjon til venstre eller höyre, og resultatet etter dette skiftet plaseres på det sted som D-verdien i ordren angir. I de ordrene hvor D ikke er lik 14 og hvor operasjonen ikke er addisjon eller subtraksjon, er det normalt ingen hensikt i å la ordren vare lenger enn en minor-cycle, fordi innholdet i TS 14 som nevnt ikke blir forandret og derfor blir også resultatet etter skiftet det samme uansett hvor lenge operasjonen varer. Önsker en å skifte innholdet i TS 14 flere posisjoner, må derfor D være lik 14 slik at i hver minor-cycle som operasjonen varer, blir innholdet i TS 14 erstattet med det samme innholdet skiftet en posisjon til venstre eller til höyre. Som eksempel vil ordren "23 - 14 d" resultere i at innholdet i TS 14 blir skiftet to plasser ned (til venstre) i TS 14. Generelt har en följende regel:

Innholdet i TS 14 skiftes "n" plasser opp (til höyre)
eller ned (til venstre) med disse ordrene:
24 - 14 1 ("n" m.c.) eller 23 - 14 1 ("n" m.c.)

På tilsvarende måte vil ikke innholdet i DS 21 bli forandret av en ordre inneholdende S 22 hvis den ikke samtidig også inneholder en av disse D-verdiene: 21, 22 og 23. Varer en ordre med S 22 en minor-cycle, blir bare et av ordene i DS 21 skiftet, nemlig DS 21_2 når operasjonen utföres i en like minor-cycle eller DS 21_3 når operasjonen skjer i en ulike minor-cycle. For å få skiftet begge ordene i DS 21 en posisjon må ordren vare minst to minor-cycles. Normalt er det ingen grunn til å la en ordre med S = 22 og hvor D ikke er 21, 22 eller 23, vare lenger enn to minor-cycles, fordi innholdet i DS 21 ikke blir forandret og dermed heller ikke resultatet etter skiftet. Hvis en önsker å skifte innholdet i DS 21

flere posisjoner til venstre, må ordren inneholde D = 21. En trenger da to minor-cycles for hver posisjon innholdet i DS 21 skal skiftes, slik at hvis skiftet skal strekke seg over "n" posisjoner, får en denne ordren:

22 - 21 l ("2n" m.c.)

Vi har nevnt för (under "Addisjon") at skifting av innholdet i DS 21 "n" posisjoner til höyre, oppnås ved:

21 - 22 l ("2n" m.c.)

Skiftoperasjonene kan kombineres med addisjon eller subtraksjon; ordren "23 - 25" f.eks., vil bevirke at innholdet i TS 14 blir skiftet en posisjon til venstre og addert til innholdet i TS 13, innholdet i TS 14 forblir uforandret. Lar en denne ordren vare i flere minor-cycles, f.eks. "23 - 25 l (5 m.c.)", vil innholdet i TS 14 bli skiftet en posisjon til venstre og addert til innholdet i TS 13 fem ganger, dvs. en får:

$$(TS\ 13) + \left(\frac{(TS\ 14)}{2} \times 5\right)$$

I DS 21 kan både adderes og skiftes på samme tid ved ordren "22 - 22 d". Denne ordren gir följende operasjon: innholdet i DS 21 skiftes en posisjon til venstre, og resultatet etter dette skiftet adderes til det opprinnelige innholdet i DS 21, dvs. at resultatet etter operasjonen blir $\frac{3}{2} \cdot$ (DS 21) i DS 21. Lar en denne ordren vare "2n" minor-cycles, blir resultatet $(\frac{3}{2})^n \cdot$ (DS 21) i DS 21.

Det kan også utföres subtraksjon i DS 21 samtidig med venstre-skift, nemlig "22 - 23 d". Denne ordre vil gi följende resultat: Hvis laveste bit (P 1) er null, blir resultatet det samme som etter ordren "22 - 21 d", dvs. et normalt venstre skift, hvis laveste bit er 1, blir derimot resultatet 1 höyere i laveste bit enn resultatet av ordren "22 - 21 d". En kan derfor kalle operasjonen i ordren "22 - 23 d" for skift til venstre med avrunding eller forhöyelse.

Disse operasjonene vil alltid bli gjort etter fortegnsregelen, dvs. at ved skifting til venstre blir höyeste bit kopiert (se "Skift" side 14).

Eksempler:

1) Skift innholdet i TS 14 én plass til höyre:

   24 - 14

2) Skift innholdet i TS 15 én plass til höyre:

   15 - 14

   24 - 15

3) Halver innholdet i TS 13:

   13 - 14

   23 - 13

4) Skift innholdet i TS 14 6 plasser til höyre:

24 - 14 l (6 m.c.)

5) Skift innholdet i DS 21_3 én plass til venstre:

$22 - 21_3$

6) Skift hele DS 21 én plass til venstre:

22 - 21 d

7) Skift DS 19 4 plasser til venstre:

19 - 21 d
22 - 21 l (8 m.c.)
21 - 19 d

8) Skift DS 21_3 én plass til höyre:

$21_3 - 22_3$

9) Skift DS 21 5 plasser til höyre:

21 - 22 l (10 m.c.)

10) Regn ut "x = 10 a" hvor "a" står i QS 18_0 og "x" skal plaseres i QS 18_1.

a) 18_0 - 14
24 - 14 d
24 - 13
23 - 25
13 - 18_1

b) 18_0 - 14
14 - 13
24 - 25 d
13 - 25
13 - 18_1

11) Regn ut "x = 10a + 10b", hvor "a" og "b" står i DS 19_{2+3} og "x" skal plaseres i TS 16.

a) 19_2 - 14
14 - 13
24 - 25 d
19_3 - 14
14 - 25
24 - 25 d
13 - 25
13 - 16

b) 19 - 21 d
21 - 22l(4 m.c.)
19 - 22 d
21_3 - 13
21_2 - 25l(3 m.c.)
13 - 16

c) 19_2 - 13
19_3 - 25
13 - 14
24 - 25 d
13 - 25
13 - 16

12) Regn ut "x = 9a", hvor "a" står i DS 21_{2+3} dvs. et langt tall.

22 - 22 l (4 m.c.)
21 - 22 l (4 m.c.)

L o g i s k e o p e r a s j o n e r

To av S-verdiene spesifiserer operasjoner på innholdet i TS 14 og TS 15.

S 25 - TS 14 & TS 15, dette er en såkalt logisk multiplikasjon, dvs. at resultatet blir enere i de posisjonene hvor både TS 14 og TS 15 har enere, resten blir nuller.

S 26 - TS 14 ≠ TS 15, dette er såkalt rudimentær addisjon, dvs. at resultatet blir "1" i de posisjonene hvor TS 14 og TS 15 har ulik verdi, og "0" i de posisjonene hvor de to er like.

Eksempler:

1) Det kan ofte være av interesse å skille ut en del av et ord, hvis en f.eks. önsker å skille ut (ekstrahere) de 4 laveste sifrene i DL 10_4 og sette disse i TS 13, kan en klare dette på följende måte:
På forhånd er lagret en konstant, (11110000 ..... 00) eller (P_1 - P_4,) i DL 12_0. En får da följende program:

```
10₄ - 14
12₀ - 15
25  - 13
```

2) Ordet som står i DL 8_0 består av 4 deler à 8 bits. En önsker å isolere disse 4 delene og plasere disse i de laveste 8 bits i ordene: DL 8_{1-4} (én del à 8 bits i hvert ord). Hvis en bruker 4 konstanter: (P_1 - P_8) i DL 12_0, (P_9 - P_{16}) i DL 12_1, (P_{17} - P_{24}) i DL 12_2 og (P_{25} - P_{32}) i DL 12_3, kan en få följende program:

```
8₀  - 15
12₀ - 14
25  - 8₁
12₁ - 14
25  - 14
23  - 14 1 (8 n.c.)
14  - 8₂
12₂ - 14
25  - 14
23  - 14 1 (16 n.c.)
14  - 8₃
12₃ - 14
25  - 14
23  - 14 1 (24 n.c.)
14  - 8₄
```

(hvis det höyeste sifret i DL 8_0 er 1, vil ordet i DL 8_4 bli galt.)
En kan imidlertid lage dette programmet mye raskere og kortere. Hvis en benytter bare én konstant, (P_1 - P_8) i DL 12_0, kan programmet se slik ut:

8_0 - 14
12_0 - 15
25 - 8_1
23 - 14 1 (8 m.c.)
25 - 8_2
23 - 14 1 (8 m.c.)
25 - 8_3
23 - 14 1 (8 m.c.)
25 - 8_4

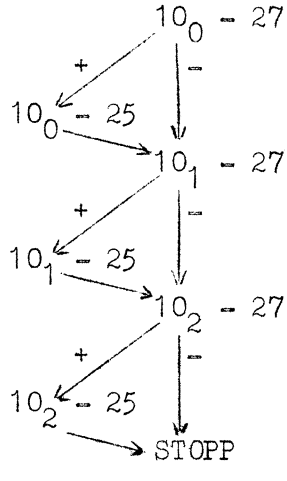
(her spiller det ingen rolle hvilken verdi det höyeste sifret i DL 8_0 har.)

T e s t - o r d r e r

Normalt vil adressen til neste ordre være entydig bestemt; i noen tilfelle er det imidlertid mulig å foreta et valg av neste ordre. En ordre som sender et ord til D 27 eller D 28, vil hente inn neste ordre i en adresse som er avhengig av verdien av det ordet som ble sendt til D-verdien. For D 27 vil valget av neste adresse være avhengig av om det bestemte ordet er positivt eller negativt, dvs. kontroll-enheten undersöker det höyeste sifret (P_{32}) i ordet, er dette sifret 1, er ordet negativt; hvis det höyeste sifret er 0, er ordet positivt (0 blir i dette tilfelle betraktet som et positivt tall). For D 28 vil vegvalget være avhengig av om det bestemte ordet inneholdet bare nuller eller ikke.

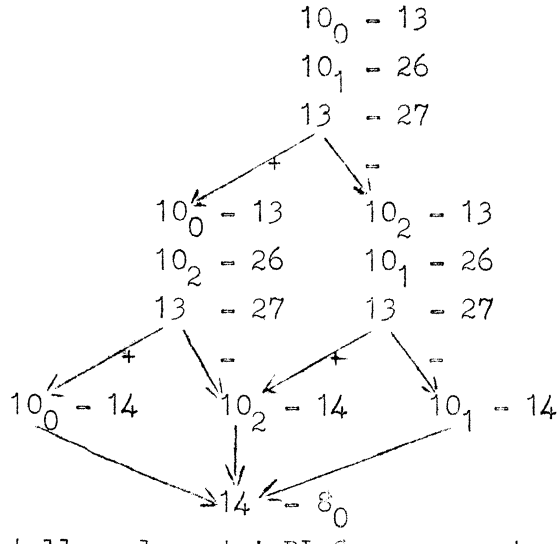
Eksempler:

1) Tallene i DL 10_0 - $_2$ kan være positive eller negative, en önsker summen av de positive av disse tallene

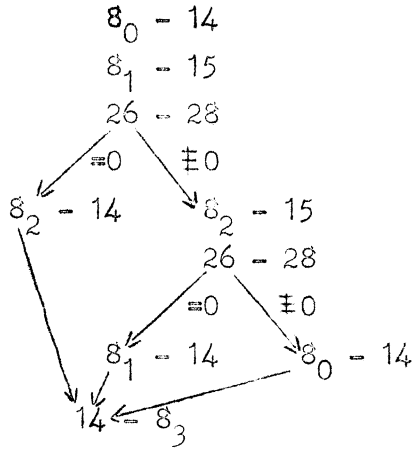


En har antatt at TS 13 er nullstillet på forhånd.

2) Tre tall er lagret i DL 10_{0-2}, plaser det höyeste av disse tallene i DL 8_0



3) Tre tall er lagret i DL 8_{0-2}, en vet at to av disse tallene er like store, mens det tredje er forskjellig. Plaser dette tredje tallet i DL 8_3



M a g n e t - t r o m m e l e n

Operasjoner som angår magnet-trommelen blir satt i gang ved hjelp av ordrer som inneholder D 30 og D 31. Som nevnt för, er magnet-trommelen inndelt i 256 spor, hvert spor kan inneholde 32 ord à 32 bits, dvs. akkurat det samme som en lang delay-linje. For å kunne hente opplysninger fra og sende dem til trommelen er det lese- og skrivehoder. Det er 16 lesehoder i DEUCE som er fast forbundet til hverandre, slik at flytting av bare et hode ikke er mulig, alle 16 må flyttes samtidig. Skrivehodene, som det også er 16 av, er forbundet på tilsvarende måte, derfor kan heller ikke disse flyttes enkeltvis. Det er imidlertid ingen forbindelse mellom lese- og skrivehodene. En kan si at magnet-trommelen er oppdelt i 16 blokker à 16 spor. Lese- og skrivehodene kan dekke èn blokk om gangen,

men som nevnt er det ikke nödvendig at lesehodene er innstilt på samme blokk som skrivehodene. Overföringer av opplysninger til og fra magnet-trommelen går alltid via DL 11, og alltid i hele spor, dvs. 32 ord.

För en overföring skal finne sted må en först sörge for å bringe lese- ev. skrivehodene i riktig posisjon ved hjelp av en ordre med D 31. Deretter settes selve overföringen i gang med D 30. En må imidlertid merke av i ordren om det er overföring til (skriving) eller fra trommelen (lesing). Dette bestemmes ved hjelp av C. Ved lesing og skifting av lesehodene må C være et like tall, dvs. C = 0 eller 2. Ved skriving og skifting av skrivehodene må C ha en ulik verdi, dvs. C = 1 eller 3.

Eksempler:

1) 5 - 31 s - lesehodene flyttes til blokk 5, "s" betyr kort operasjon
   9 - 31 1 - skrivehodene flyttes til blokk 9
   14 - 30 1 - overför innholdet av DL 11 til det sporet som nå er under skrivehode nr. 14
   0 - 30 s - overför innholdet av det sporet som nå er under lesehode nr. 0 til DL 11
2) Overför innholdet av DL 11 til spor nr. 121. Spor 121 er det samme som spor 9 i blokk 7 eller 7/9. En får da:
   7 - 31 1
   9 - 30 1
3) Overför innholdet av spor nr. 200 til DL 11. Spor 200 er det samme som spor 8 i blokk 12 (12/8). En får:
   12 - 31 s
   8 - 30 s

I n p u t

Den normale måten å lese inn opplysninger i DEUCE er fra hullkort i IBM 528. Denne innlesingen vil bli nærmere omtalt i neste hefte. En kan imidlertid også få lest inn opplysninger på en annen måte, nemlig ved å overföre et ord om gangen fra et sett på 32 "Input Dynamiciser" - lamper på kontrollbordet. För overföringen av et ord skal finne sted, må dette ordet være "satt opp" på ID-lampene. Hver enkelt lampe er utstyrt med en bryter, med denne kan en tenne eller slukke lampen. Et binærord blir representert på ID-lampene ved at lamper som er "på" representerer binære enere og lamper som er "av" angir binære nuller. Når et ord er satt opp på lampene ved hjelp av bryterne, kan dette overföres

ved en ordre som inneholder "Source 0" og hvor D-verdien angir den lagerplassen i hurtig-minnet som ordet skal overföres til.

Ved hjelp av ID-lampene på kontrollbordet kan operatören lese så mange ord som önskelig inn i DEUCE, men dette er naturligvis en svært langsom operasjon med mange muligheter for feil fra operatörens side.

Eksempel:

"0-14", betyr at ordet på ID-lampene blir overfört til TS 14.

O u t p u t

Resultater og opplysninger som en vil ha ut av DEUCE, blir normalt punchet i hullkort i IBM 528. Dette vil bli nærmere omtalt i neste hefte. En kan imidlertid også få opplysninger ut av maskinen på en annen måte, nemlig ved å overföre et ord om gangen til en rekke på 32 "Output Staticiser"-lamper på kontrollbordet. Ord som skal overföres til OPS-lampene, må overföres ved ordrer som inneholdet D 29 og hvor S-verdien angir lagerplassen i hurtig-minnet for det ordet som skal overföres.

På tilsvarende måte som for ID-lampene, blir et ord representert på OPS-lampene ved at lamper som er "på" betegner binære enere og lamper som er "av" angir binære nuller. På denne måten er det mulig for operatören å avlese et ord som blir sendt til D 29, men dette er selvfölgelig en meget tungvint og langsom måte å få opplysninger ut av DEUCE på.

Eksempel:

"10_{14} - 29", angir at innholdet i DL 10_{14} skal overföres til OPS-lampene.

T S C o u n t

Som nevnt under kapitlet om kontrollorganet, er det normalt de to ordredelene, NIS og T, som bestemmer hvor neste ordre skal hentes fra. Dette valget av neste ordre kan imidlertid ignoreres ved å benytte Destination 0. D 0 gjör det mulig å overföre hvilket som helst ord innen hurtig-minnet til TS COUNT. Innholdet av dette ordet blir da neste ordre, uansett NIS i den ordren som inneholder D 0.

En overföring til D 0 må vanligvis være lang (C = 1) fordi ordene fra den valgte NIS bare blir erstattet fra den valgte Source så lenge overföringen finner sted. En må altså kode den ordren som overförer et ord til D 0, slik at overföringen blir utfört i den samme minor-cycle som

ordren forsöker å hente inn sin neste ordre, det ordet som sendes til D 0, vil da gå inn i TS COUNT som neste ordre og bli utfört på normal måte. En bör derfor gjöre til regel at alle ordrer med D 0 skal være lange (C = 1).

Eksempel:

"15 - 0 1", denne ordren resulterer i at neste ordre blir hentet fra TS 15. NIS blir altså ignorert.

T r i g g e r s

I nesten alle de ordrene en har sett hittil, har det vært overföring av en serie av siffer fra et sted til et annet innen DEUCE. En del ordrer medförer ikke noen overföring av siffer-serier, men sörger bare for at noe bestemt skal skje eller begynner å skje. Dette kan være divisjon, multiplikasjon osv. De fleste av disse operasjonene er samlet i en gruppe med felles D-verdi, D 24, og kalles "Destination Triggers". Den operasjon som blir satt i gang for hver enkelt "Trigger" bestemmes av S-verdien. I det följende skal en se nærmere på de enkelte "Destination Triggers".

M u l t i p l i k a s j o n - (0 - 24)

DEUCE har innebygd en automatisk multiplikasjonsanordning som står i forbindelse med DS 21 og TS 16. Produktet blir på 64 bits og er riktig etter metoden uten fortegn. För multiplikasjonen kan starte, må de to faktorene lagres i DS 21_3 og TS 16. En må også sörge for at DS 21_2 er tom för multipliseringen. Multiplikasjonen startes med "0 - 24", denne ordren må adlydes i en ulik minor-cycle, det kan en dirigere enkelt med "Wait"-nr. Selve multiplikasjonen trenger 65 minor-cycles eller litt over 2 major-cycles. I denne tiden må hverken D 16, D 21, D 22 eller D 23 brukes; S 21 og S 22 vil gi en sifferserie som representerer et delvis ferdig produkt. S 16 vil kunne brukes og gir naturligvis den faktoren som er lagret i TS 16. Det ferdige produkt vil stå i DS 21 og bli oppfattet som et langt tall å 64 bits.

Eksempel:

En önsker å multiplisere de to positive tallene som står lagret i DL 10_0 og DL 10_1, og resultatet skal lagres i DL 10_{2+3}.

a) $30 - 21_2$
b) $10_0 - 16$
c) $10_1 - 21_3$
d) $0 - 24$ (ulik minor-cycle)
e) $30 - 29$
f) $21 - 10_2$ d

Merknader:

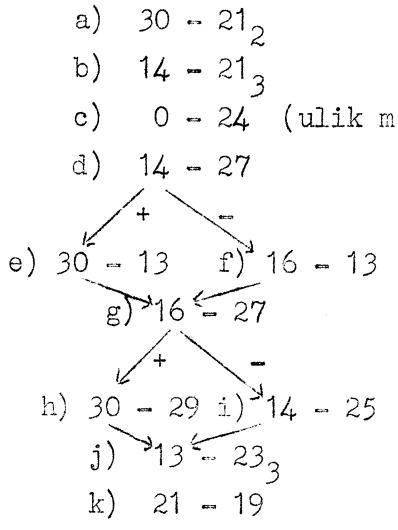
a) För multiplikasjonen må DS 21_2 tömmes.
b,c) De to faktorene lagres i TS 16 og DS 21_3.
d) Multiplikasjonen kontrolleres av en elektronisk bryter som kalles "MULT". Denne blir satt "på" når ordren "0 - 24" blir satt i verk. Dette må imidlertid være en ulik minor-cycle. "MULT" vil være aktiv i 65 minor-cycles, deretter vil den automatisk bli "slått av". I denne tiden kan en utföre andre operasjoner, men ikke operasjoner som angår TS 16 eller DS 21 (S 16 er dog unntatt).
e) Ordren "0 - 24" vil ikke kunne vare lengre enn 32 minor-cycles, men det må gå minst 65 m.c. för ordren (f) kan utföres, derfor er (e) bare en "dummy"-ordre dvs. en ordre som bare har som oppgave å fylle ut tiden.
f) Det ferdige produktet i DS 21 overföres til DL 10_{2-3}.

Som kjent blir et negativt tall "-a" representert i DEUCE som "$2^{32} - a$". Hvis en multipliserer et slikt negativt tall med et positivt tall "b", får en $(2^{32} - a)b$ eller 2^{32} b-ab. Det riktige dobbelt-lengde produktet etter fortegnsmetoden er $(2^{64} - ab)$, derfor må det forelöpige resultatet korrigeres ved at en adderer til $(2^{64} - 2^{32}b)$ eller $2^{32}(2^{32} - b)$. Dette er det samme som å subtrahere "b" i överste halvpart av produktet.

Hvis to negative tall multipliseres, blir resultatet $(2^{32} - a)(2^{32} - b)$ eller $(2^{64} - 2^{32}a - 2^{32}b + ab)$. Den nödvendige korreksjon vil da være å addere (a + b) til överste halvpart av produktet. Nå er det bare "-a" og "-b" som er öyeblikkelig tilgjengelig, derfor må en korrigere slik (-(-a)-(-b)).

Eksempel:

En önsker å multiplisere de to tallene (positive eller negative) som er lagret i TS 14 og TS 16 og resultatet skal lagres i DS 19:



Merknader:

a,b,c) DS 21_2 tömmes, faktoren i TS 14 overföres til DS 21_3 og multiplikasjonen settes i gang.

d,e,f) Den ene faktoren (i TS 14) undersökes om negativ eller positiv. Hvis denne faktoren er negativ, overföres den andre faktoren til TS 13. Hvis positiv nullstilles bare TS 13.

g,h,i) På samme måte undersökes fortegnet i den andre faktoren (i TS 16). Hvis negativ, adderes den förste faktoren i TS 13, hvis positiv, gjöres ikke noe.

j) Korreksjonsleddet som nå står i TS 13, subtraheres i höyeste halvpart i produktet (DS 21_3).

k) Det korrigerte produktet overföres til DS 19. Denne operasjonen må ikke gjöres för tidligst 65 minor-cycles etter multiplikasjonen settes i gang.

Etter fortegnskorreksjon ved multiplikasjon vil resultatet alltid være at de to höyeste bits er like, 00 eller 11. Da en bare trenger en bit for å angi fortegnet kan produktet, om en önsker det, alltid skiftes en plass opp uten at "over-flow" inntreffer.

Det er intet til hinder for at multiplikator og/eller multiplikand kan være binærbröker. Produktet etter multiplikasjonen vil da også være en binærbrök. Binærkommaets plasering i produktet er bestemt etter denne enkle regel:

> Antall bits til venstre for kommaet i produktet er lik summen av antall bits til venstre for kommaet i multiplikator og multiplikand.

Hvis f.eks. både multiplikator og multiplikand har 16 bits på venstresiden av kommaet, får produktet 32 bits til venstre for kommaet, dvs. kommaet kan tenkes plasert mellom DS 21_2 og DS 21_3.

En er ikke alltid interessert i å beholde alle binærposisjonene til venstre for kommaet i produktet, en må da kutte vekk de som ikke önskes. I eksemplet ovenfor hvor begge faktorene hadde 16 bits til venstre for komma, kan det tenkes at en også önsker samme kommaplasering i produktet. Innholdet i DS 21 må da skiftes 16 posisjoner opp eller ned slik at kommaet blir plasert mellom P 16 og P 17 i DS 21_3 eller DS 21_2. Er en fortsatt interessert i produktet med dobbelt ord-lengde (64 bits), må en skifte 16 posisjoner ned slik at kommaet kommer midt i DS 21_2. Önsker en derimot produktet med bare 32 bits, kan en velge hvilken veg en vil skifte DS 21, skifter en 16 posisjoner opp, vil produktet bli å finne i DS 21_3, skifter en 16 posisjoner ned, finner en produktet i DS 21_2.

Når en på denne måten har kuttet vekk et eller flere av de laveste binærsifrene, vil naturligvis ikke produktet være eksakt lenger. Det er introdusert en feil i laveste bit, denne feilen kan variere fra 0 til -1. En kan ikke korrigere denne feilen, men den kan utjamnes ved å addere 1/2 til den laveste bit slik at feilen vil variere fra + 1/2 til - 1/2. I DEUCE gjör en denne avrundingen eller forhöyelsen ved å addere 1 til det höyeste bit av de som kuttes vekk.

Selve programmeringen og kodingen av denne avrundingen kan gjöres på flere måter, nedenfor skal en vise et eksempel på en metode for avrunding.

Eksempel:

Multipliser innholdet i DS 19_2 med innholdet i DS 19_3, det förste tallet har 27 bits til venstre for kommaet mens det andre har 20 bits. Produktet skal lagres i TS 15 med 20 bits på venstresiden av kommaet.

a) 30 - 21_2
b) 19_2 - 16
c) 19_3 - 21_3
d) 0 - 24
e) 16 - 27

\+ / -

f) 30 - 13 g) 19_3 - 13

h) 19_3 - 27

\+ / -

i) 1 - 1 j) 16 - 25

k) 13 - 23_3
l) 21 - 221 (10 m.c.)
m) 29 - 23_2
n) 21_3 - 15

Merknader:

a,b,c) Töm DS 21_2, og sett multiplikator og multiplikand på plass.

d) Start multiplikasjonen i en ulike minor-cycle.

e-k) Korriger med hensyn på fortegnene.

l) Skift innholdet i DS 21 5 posisjoner til höyre slik at kommaet kommer mellom P 20 og P 21 i DS 21_3, derved blir det 20 bits til venstre for kommaet i DS 21_3.

m) Her ser det ut som om 1 blir subtrahert fra höyeste bit i DS 21_2, i virkeligheten er det imidlertid följende som foregår: Etter en multiplikasjon vil alltid innholdet i DS 21, dvs. produktet, bli behandlet som et langt tall å 64 bits. Pröver en å addere et kort tall (32 bits) til laveste halvpart av et langt tall, vil det korte tallet automatisk bli forlenget til et langt tall og denne forlengelsen, som er 32 kopier av [illegible] eller fortegnet i det korte tallet, blir [illegible] eller subtrahert i höyeste halvpart av det lange tallet. (Se mermere om dette under "TCB") I dette tilfellet blir S 29 som er (000 ... 001) eller P 32. forlenget til (000 ... 001111 ... 111) eller P 32 - P 64. Både för og etter forlengelsen representerer dette tallet -2^{31}. Når så dette forlengede tallet subtraheres fra DS 21, har det samme virkning som å addere 1 til höyeste bit i DS 21_2.

n) Det avrundete produktet overföres fra DS 21_3 til TS 15.

Hvis en angir antall binærposisjoner til venstre for kommaet i multiplikator, multiplikand og produkt for henholdsvis n_1, n_2 og N har en at:

$$n_1 + n_2 = N$$

Hvis en videre angir det antall bits som önskes på venstresiden av kommaet i produktet, for N^1, har en videre at:

$$N - N^1 = s$$

hvor "s" er det antall bits som skal kuttes vekk, dvs. det er bare tilfelle når $s > 0$. Når $s = 0$, betyr det at produktet skal beholdes i sin opprinnelige form; hvis $s < 0$, vil det si at en önsker flere binærplasser til venstre for kommaet.

"s" forteller hvordan og hvor langt innholdet i DS 21 må skiftes for å få produktet i den form som önskes. Framgangsmåten varierer etter hvor mange bits som önskes i produktet, 64 bits eller 32.

D o b b e l t - l e n g d e p r o d u k t (64 b i t s)

s < 0: skift innholdet i DS 21 "s" posisjoner til höyre

s = 0: intet skift

s > 0: skift innholdet i DS 21 "s" posisjoner til venstre

I alle disse tilfellene kan produktet hentes fra DS 21 som et langt tall.

E n k e l - l e n g d e p r o d u k t (32 b i t s)

s < 0: skift innholdet i DS 21 "s" posisjoner til höyre

s = 0: intet skift

$16 \geq s > 0$: skift innholdet i DS 21 "s" posisjoner til venstre

$32 > s \geq 16$: skift innholdet i DS 21 "32-s" posisjoner til höyre

s = 32: intet skift

$64 > s > 32$: skift innholdet i DS 21 "s-32" posisjoner til venstre

I de tre förste tilfellene vil produktet etter skiftet bli stående i DS 21_2, i de tre siste tilfellene blir det stående i DS 21_3.

I eksemplet foran var $n_1 = 27$, $n_2 = 20$ og $N^1 = 20$, altså får en $s = 27 + 20 - 20 = 27$. Da "s" i dette tilfelle ligger mellom 16 og 32, må innholdet i DS 21 skiftes "32-s", dvs. 5 posisjoner til höyre for at produktet skal bli stående i DS 21_3 i den form som önskes.

Eksempel:

Etter en multiplikasjon står produktet i DS 21 som et langt tall med 8 binærposisjoner til venstre for kommaet. En önsker imidlertid produktet med enkel ord-lengde og bare 4 bits på venstresiden av kommaet. En får da følgende program:

a) 22 - 21 1 (6 m.c.)

b) 22 - 23 d

Merknader:

a) I dette eksemplet er s = 4, dvs. innholdet i DS 21 må skiftes 4 posisjoner til venstre. I dette tilfellet önsker en imidlertid å avrunde produktet, derfor skiftes först 3 (s-1) posisjoner til venstre.

b) I denne ordren skiftes innholdet i DS 21 en posisjon til venstre samtidig med avrunding slik at tilsammen har disse to ordrene sörget for at innholdet er skiftet 4 posisjoner til venstre og rundet av. Produktet står nå i DS 21_2.

Som regel er en interessert i å runde av produktet etter at siffer er kuttet vekk på venstresiden av kommaet. En har sett to eksempler på hvordan avrundingen kan utföres i DEUCE. Generelt kan en si at det er følgende to metoder for avrunding.

a) Dobbelt-lengde produkt eller enkelt-lengde produkt i DS 21_2.
Innholdet i DS 21 skiftes først "s-1" posisjoner til venstre med ordren "22-21 l (2(s-1)m.c.)", deretter skiftes det en posisjon til venstre med avrunding ved ordren "22-23 d".

b) Enkel-lengde produkt i DS 21_3.
Selve skiftet av produktet utföres etter de reglene som er gitt foran, mens avrundingen foregår med f.eks. ordren "29-23_2" eller "21-23_2".

D i v i s j o n - (1 - 24)

I DEUCE er innebygd et automatisk divisjonsutstyr, som står i forbindelse med TS 16 og DS 21. Dette utstyret kan behandle både positive og negative divisorer og dividender, og produsere en kvotient med riktig fortegn. Divisor sendes til TS 16 og dividenden til DS 21_3. Divisjon startes med ordren "1 - 24", og må adlydes i en ulik minor-cycle. Hele divisjonsoperasjonen trenger 66 minor-cycles, altså èn minor-cycle mer enn multiplikasjonen. Når divisjonen er ferdig vil kvotienten stå i DS 21_2, mens en noe modifisert form av resten finnes i DS 21_3.

Hvis vi kaller dividenden A, divisor B, kvotienten Q og resten R, forekommer det följende forhold mellom disse störrelsene i divisjon i DEUCE:

$$2^{31} A = QB + R$$
$$B > 0 : 0 \leq R < B$$
$$B < 0 : B \leq R < 0$$

Forutsetningen ved divisjon er at både A, B og Q ikke trenger mer enn 32 bits. Alle störrelsene i divisjonen vil bli behandlet etter fortegnsmetoden. Når Q skal være et kort ord (32 bits), betyr dette at $/A/ \leq /B/$, en kan videre se at $/A/ = /B/$ bare i de tilfellene A er negativ og B er positiv. En har da således at

$$/A/ \leq /B/ \leq 2^{31}$$

En finner videre at Q alltid er algebraisk mindre eller lik den riktige kvotienten. I de tilfellene det er mulig å oppnå en eksakt kvotient, vil Q bli lik denne bare hvis B er positiv, i de övrige tilfellene vil Q være en enhet mindre i laveste posisjon. Hvis en önsker å finne den riktige resten i divisjonen, er det nödvendig å gjöre en ytterligere begrensing, slik:

$$/A/ \leq /B/ \leq 2^{30}$$

Den modifiserte resten (r), som står i DS 21_3 etter divisjonen, forholder seg til den korrekte resten (R) slik:

$$r = 2 (2R - B)$$

slik at

$$R = \tfrac{1}{2}(\tfrac{1}{2}r + B)$$

Det er da innlysende at hvis $2^{30} \leq B \leq 2^{31}$, kan det hende at "r" vil være större enn et kort ord, og da vil det være umulig å kunne frambringe den riktige resten. Dette er årsaken til den ekstra begrensingen ovenfor.

Ut fra forholdet "$2^{31}A = QB + R$" kan en forestille seg at dividenden A blir forlenget med 31 binærposisjoner, nemlig 31 nuller til venstre for tallet. Hvis både dividend og divisor er hele tall, har ikke divisor bits på venstresiden av kommaet, mens dividenden vil ha 31 bits til venstre for kommaet, nemlig de 31 nullene som er föyet til. Antall bits til venstre for kommaet i kvotienten finner en etter denne enkle regelen:

> Antall posisjoner til venstre for kommaet i kvotienten er lik differensen mellom antall posisjoner til venstre for kommaet i dividend og divisor.

I divisjon med hele tall har derfor kvotienten 31 posisjoner på venstresiden av kommaet, dvs. en kan tenke seg dette mellom P 31 og P 32.

Det er naturligvis mulig at divisor og/eller dividend er binærbröker. Hvis dividenden har 5 bits og divisor 8 bits på venstresiden av kommaet, får kvotienten 28, dvs. 31 + 5 - 8, bits til venstre for kommaet. Angir vi generelt antall bits til venstre for kommaet i dividend og divisor for m_1 og m_2, kan en finne antall bits til venstre for kommaet i kvotienten (M) etter denne formelen:

$$M = 31 + m_1 - m_2$$

Ofte vil en ikke være interessert i å beholde alle binærposisjonene til venstre for kommaet. Hvis en benytter betegnelsen M^1 for det antall bits som önskes på venstresiden av kommaet i kvotienten, har en at:

$$s = M - M^1$$

hvor "s" angir det antall bits som må kuttes vekk for at kvotienten skal få den önskede form. Hvis $s < 0$, betyr det at en önsker flere posisjoner til venstre for kommaet.

Hvordan og hvor langt innholdet av DS 21 må skiftes, bestemmes av "s" på fölgende måte:

$s < 0$: skift innholdet i DS 21_2 "s" posisjoner til höyre

$s = 0$: intet skift

$s > 0$: skift innholdet i DS 21_2 "s" posisjoner til venstre

En må her være oppmerksom på at innholdet av DS 21_2 ikke kan skiftes mer enn en posisjon uten at også innholdet av DS 21_3 blir skiftet. Er en derfor interessert i innholdet i DS 21_3 slik som det foreligger, må enten dette overföres til et annet sted i hurtigminnet eller kvotienten

må overföres til TS 14 og blir skiftet der. Hvis derimot innholdet i DS 21_3 ikke er av interesse, kan en gjerne utföre skiftingen i DS 21. Forutsetningen er imidlertid da at bryteren "TCB" ikke er satt "av", dvs. gjort om innholdet i DS 21 til et langt tall å 64 bits.

Når en ved skift har kuttet vekk endel bits til venstre for kommaet, har en også fört inn en feil i kvotienten som kan variere fra 0 til -1 i det laveste av de bits som beholdes. Denne feilen kan utjamnes ved at hver gang det höyeste av de bits som kuttes vekk er 1, adderes 1 til det laveste av de bits som beholdes.

Avrundingen utföres naturligvis bare når det er venstre-skift, og da kan skiftet ("s" posisjoner) og avrundingen utföres av disse to ordrene:

22 - 21 1 (2(s-1)m.c.)

22 - 23_2

Det kan altså være nödvendig med skift etter divisjonen for å bringe binær-kommaet på riktig plass og för divisjonen for å tilfredsstille betingelsen /B/ $\geq$ /A/.

Konstruksjonen av divisjonsutstyret gjör også at en ikke kan sende divisor til TS 16 i den ulike minor-cycle som kommer umiddelbart foran den ulike minor-cycle som divisjonen starter i.

Eksempel:

1. Divider tallet i TS 14 med det i TS 16, idet en vet at tallverdien til dette siste tallet er störst. Overför kvotienten med 31 bits til venstre for komma til TS 16.
   a) 14 - 21_3
   b) 1 - 24 (ulik m.c.)
   c) 30 - 29
   d) 21_2 - 16

   Merknader:
   a) dividenden til DS 21_3.
   b) sett i gang divisjonen. Dette må være i en ulik minor-cycle.
   c) en "dummy"-ordre. Divisjonen trenger som nevnt 66 minor-cycles, derfor trengs en ordre som bare har til oppgave å "bruke" tid.
   d) kvotienten overföres til TS 16. Kvotienten vil ha 31 plasser etter komma.
2. En önsker å dividere tallet i TS 16 med det i TS 14. En vet at tallet i TS 14 er större i tallverdi. Kvotienten med 31 binær-plasser til venstre for komma plaseres i TS 14 og den eksakte resten i TS 15.

a) 16 - 21_3
b) 14 - 16
c) 1 - 24 (ulik m.c.)
d) dummy
e) 22 - 21_3
f) 16 - 22_3
g) 22_3 - 15
h) 21_2 - 14

Merknader:

a,b) dividenden til DS 21_3 og divisor til TS 16
c) divisjonen settes i gang i en ulik minor-cycle
d) "dummy"-ordre for å "bruke" opp tid
e) halverer den modifiserte resten i DS 21_3. En har altså $\frac{r}{2}$
f) divisor adderes til slik at i DS 21_3 står $(\frac{r}{2} + B)$
g) DS 21_3 halveres slik at en får $\frac{1}{2}(\frac{r}{2} + B)$, dette er det samme som R og overföres til TS 15
h) kvotienten med 31 binær-plasser etter komma lagres i TS 14

3. Divider tallet i TS 14 med tallet i TS 16, idet en vet at det siste tallet er det störste i tallverdi. Dividenden har 2 bits til venstre for kommaet og divisor har 4 bits. Kvotienten med 25 bits på venstresiden av kommaet (avrundet) skal lagres i DS 19_2.

a) 14 - 21_3
b) 1 - 24 (ulik m.c.)
c) 1 - 1
d) 22 - 21 1 (6 m.c.)
e) 22 - 23_2
f) 21_2 - 19_2

T I L (T w e l f t h I m p u l s e L i n e)

TIL er en impuls som er tilstede i DEUCE når kort passerer gjennom IBM 528. Denne impulsen starter like för siste rad (nier-raden) kommer til börstene eller punchemagnetene og varer til en stund etter. Ved å benytte ordren "2 - 24" kan en undersöke om TIL er tilstede eller ikke, og foreta et valg av neste ordre avhengig av dette. En vil komme nærmere tilbake til TIL og denne bestemte ordre i et senere kapitel, når en skal gå detaljert gjennom IBM 528.

T C A

Normalt vil S 16 hente innholdet i TS 16, men det kan også ha en annen funksjon. Istedetfor å gi innholdet i TS 16, kan det gi innholdet i DL 10 en minor-cycle forsinket. Når S 16 har denne funksjonen, vil ordene i DL 10 være tilgjengelig på to steder, men i forskjellige minor-cycles. Det ordet som er lagret i minor-cycle 16 i DL 10 for eksempel, vil også være tilgjengelig fra S 16 i m.c. 17. I dette tilfelle virker ikke TS 16 som en lagerplass lenger og det innholdet som har vært lagret der, forsvinner.

Denne andre funksjonen av S 16 kontrolleres av en elektronisk bryter som kalles "TCA". En sier at denne bryteren er "av" når S 16 gir det ordet som er lagret i TS 16, og "på" når S 16 gir ordrene i DL 10 en minor-cycle forsinket. TCA settes "på" med ordren "3 - 24". Det er ingen grunn til å slå TCA "av" för TS 16 igjen skal brukes som vanlig lagerplass. Ved en overföring til TS 16 (D 16) blir derfor TCA automatisk slått "av", det er derfor ingen spesiell ordre for dette.

Eksempel:

a) 3 - 24

b) 16 - 9 1 (32 m.c.)

Merknader:

a) TCA settes "på"

b) Resultatet av denne ordren blir at innholdet av DL 10 blir overfört til DL 9 en minor-cycle forsinket, slik at DL 10_0 går til DL 9_1, DL 10_1 går til DL 9_2 osv.

Ved multiplikasjon og divisjon blir "TCA" automatisk slått "av" når multiplikanden respektive divisor sendes til TS 16, og må ikke settes "på" igjen ved ordren "3-24" för multiplikasjonen eller divisjonen er ferdig.

T C B

De 64 binær-sifrene i en DS kan enten behandles som to enkle ord à 32 bits eller et langt ord à 64 bits. For å kunne tillate dette dirigeres de automatiske operasjonene som er forbundet med DS 21 av en elektronisk bryter som kalles "TCB". Denne kan settes "på" eller "av" ved ordrene "5 - 24" og "4 - 24". Når TCB er "av" utföres operasjonene i DS 21 med dobbelt lengde (64 bits), når den er "på", utföres operasjonene på enkle ord.

Ved vanlig overföring, f.eks. "20 - 21 d" spiller det ingen rolle om de 64 binær-sifrene i DS representerer ett langt eller to korte tall. Ved aritmetiske operasjoner, derimot, er det en forskjell, og da bare i hvordan det $32^{\underline{te}}$ eller höyeste sifret i de to minor-cycles blir behandlet. Når en arbeider med lange ord, er det alltid den like minor-cycle som inneholder den laveste halvpart av ordet.

Når en adderer tall i DS 21, gjör en som kjent det med D 22. Hvis TCB er "på", opptrer DS 21 som to atskilte regneverk og menten fra det $32^{\underline{te}}$ sifret i begge ordene forsvinner. Hvis TCB er "av", vil eventuell mente fra det höyeste siffer i det ordet som er lagret i den like minor-cycle, bli addert til det laveste siffer i den ulike minor-cycle. Mente fra det höyeste siffer i dette ordet blir borte i begge tilfeller. På tilsvarende måte vil det være ved subtraksjon i DS 21 (D 23).

Både for addisjon og subtraksjon er det en spesiell anordning for operasjon på den laveste halvpart av et langt tall. Denne anordningen virker slik at hvis et kort tall blir addert eller subtrahert i DS 21 i en like minor-cycle og TCB er "av", vil dette korte tallet bli forlenget til et langt tall. Slik at når en overförer et kort tall til D 22 eller D 23 i en like minor-cycle og TCB er "av", vil også 32 kopier av fortegnet (höyeste siffer) i det korte tallet bli addert eller subtrahert i den följende ulike minor-cycle. Dette vil også være tilfelle hvis en slik operasjon varer i flere minor-cycles og slutter i en like minor-cycle.

Det samme gjelder for S 22, dvs. at innholdet i DS 21 blir dividert med 2 eller skiftet til venstre. Med TCB "av" blir DS 21 betraktet som å inneholde et langt ord, mens TCB "på" vil få DEUCE til å oppfatte innholdet i DS 21 som to enkle ord.

Ved multiplikasjon blir TCB automatisk slått "av" når multiplikasjonen starter, og vil fortsatt være "av" når multiplikasjonen er ferdig.

TCB blir, derimot, automatisk satt på ved starten av en divisjon, og vil være i denne stillingen ("på") når divisjonen er ferdig.

Eksempler:

a) 20 - 22 d Hvis TCB er "på", vil de to tallene i DS 20 bli addert til de to i DS 21. Hvis TCB er "av", er det meningen at innholdet i DS 20 skal adderes til det lange tallet i DS 21. Hvis det höyeste sifret i den like minor-cycle i DS 20 er "1" og overföringen foregår i en ulik minor-cycle fulgt av en like, vil den spesielle anordningen som er nevnt, gjöre at en uönsket "-1" blir addert i höyeste halvpart. For å

være sikker på at det skal bli korrekt resultat, må derfor overföringen skje i en like minor-cycle etterfulgt av en ulike. Dette indikeres slik: "20 - 22 d (e,o)", hvor "e,o" betyr "even - odd" (lik - ulik).

b) 15 - 23_3 Det spiller ingen rolle i hvilken tilstand TCB befinner seg, da subtraksjonen foregår i en ulik minor-cycle.

c) 15 - 22_2 I dette tilfelle vil TCB påvirke operasjonen slik: hvis TCB er "på", vil DS 21 inneholde to korte tall og addisjon i DS 21_2 vil ikke påvirke innholdet i DS 21_3 på noen måte, hvis TCB er "av" vil DS 21 inneholde et langt ord og under addisjonen vil ordet i TS 15 bli forlenget til et langt ord.

d) 19 - 22 l (6 m.c. e,o) (TCB "av") Denne ordren adderer det lange ordet i DS 19 tre ganger inn i DS 21.

e) 21 - 22 d (e,o) (TCB "av") Dette fordobler tallet i DS 21.

f) 21 - 22 l (4 m.c. e,o) (TCB "av") Dette firedobler tallet i DS 21.

g) 22 - 21 d (e,o) (TCB "av") halverer tallet i DS 21.

h) 22 - 21 l (2n m.c. e,o) (TCB "av") dividerer tallet i DS 21 med 2^n.

i) 22 - 22 d (e,o) (TCB "av") multipliserer det lange ordet i DS 21 med 1,5.

j) 29 - 23_2 eller 21 - 23_2 Begge disse ordrene kan benyttes til avrunding av produktet etter en multiplikasjon hvis en på forhånd har skiftet produktet slik at de bits som en ikke er interessert i befinner seg i DS 21_2. Disse to ordrene vil da bevirke at i de tilfellene höyeste bit i DS 21_2 er 1, blir 1 addert til laveste bit i DS 21_3. Det avrundede produktet vil stå i DS 21_3 (dvs. enkel ord-lengde).

k) 22 - 21 l (2(s-1)m.c. e o) I de tilfellene en fortsatt vil beholde 22 - 23 d (e,o) produktet med dobbelt-lengde eller produktet i DS 21_2 med enkel-lengde og "s" bits skal kuttes vekk bak kommaet, vil disse to ordrene sörge for at disse "s" bits blir kuttet vekk, dvs. skiftet ut til venstre, samtidig med avrunding.

A l a r m

For å kunne indikere eventuelle feil i programmet, finnes der på kontrollbordet en innretning som alarmerer operatören ved hjelp av lyd og lys. Denne alarmen startes ved ordren "7 - 24" og kan stoppes enten ved ordren "6 - 24" eller ved å trykke ned en spesiell knapp på kontrollbordet.

" C l e a r O P S "

Ved hjelp av en spesiell ordre "8 - 24" kan en nullstille eller slukke alle Output Staticiser-lampene.

L e s i n g o g p u n c h i n g

En skal komme nærmere inn på dette i neste hefte og vil her bare begrense seg til å nevne at ordrene "10 - 24" og "12 - 24" setter i gang kortene i henholdsvis punche- og lesebanen i IBM 528. Ordren "9 - 24" stopper kortene i begge banene.

VI. KODING

I foregående kapitel er laget en del korte programmer for å vise bruken av S- og D-verdiene. Selv om en vesentlig del av kodingen allerede er gjort i disse programmene, vil ikke ordrene i programmet være forståelige for DEUCE i den form de der står. Kodingen omfatter derfor mer enn å kode S- og D-verdiene i ordrene, de övrige ordre-delene må også kodes for at ordrene skal kunne bli forstått og utfört av DEUCE.

Hver enkelt ordre i et program opptar som kjent et ord i DEUCE. De 32 binærsifrene i et ord er delt opp i följende ordre-deler (se fig. 3 på side 25).

P_1	(laveste bit)	- brukes vanligvis ikke
$P_2 - P_4$		- NIS (Next Instruction Source)
$P_5 - P_9$		- S (Source)
$P_{10} - P_{14}$		- D (Destination)
$P_{15} - P_{16}$		- C (Characteristic)
$P_{17} - P_{21}$		- W (Wait)
$P_{22} - P_{25}$		- Joe (Joe number)
$P_{26} - P_{30}$		- T (Time)
P_{31}		- brukes vanligvis ikke
P_{32}	(höyeste bit)	- G (Go digit)

En ordre må spesifisere operasjonens art, når operasjonen skal begynne, hvor lenge den skal vare, og når og hvorfra neste ordre skal hentes inn i TS COUNT.

1. Operasjonens art

Dette spesifiseres av S- og D-verdiene. (Kap. V).

2. Operasjonens start

För en ordre kan utföres må den overföres til TS COUNT (kontrollregisteret). Da en ordre normalt overföres til TS COUNT fra en av de 8 förste delay-linjene (DL 1-8), må dette naturligvis skje i den minor-cycle i hvilken ordren er lagret i delay-linjen. Hvis ordren er lagret i DL 4_{15}, må overföringen finne sted i m.c. 15, er den lagret i DL 3_{20}, må overföringen skje i m.c. 20 osv. Generelt må en ordre som er lagret i m.c. (m) i en delay-linje, overföres til TS COUNT i m.c. (m). Neste minor-cycle, m.c. (m + 1), kalles "Set-up Minor-cycle", fordi den förste

halvparten av denne cycle blir benyttet til å velge ut de riktige forbindelsene på grunnlag av kodene i ordre-ordet. I denne cycle kan ikke noen andre operasjoner utföres, da alle overföringer vil komme fra to forskjellige Sources (i den gamle og den nye ordren) og vil derfor være fullstendig meningslöse. I minor-cycle deretter, m.c. (m+2), kan forbindelsene bli satt opp slik at operasjonen kan starte. Det som bestemmer når operasjonen starter er W; er denne = 0, starter operasjonen i m.c. (m+2), dvs. ingen ventetid. Hvis W = 1, blir det en ventetid på 1 minor-cycle slik at operasjonen starter i m.c. (m+3). Generelt kan en si at det blir en ventetid på "W" minor-cycles för operasjonen starter, dvs. den starter i m.c. (m+2+W).

En kan tenke seg at kontroll-organet bestemmer dette ved å telle nedover i W (subtrahere 1) for hver minor-cycle. Operasjonen vil da starte i den minor-cycle som följer like etter den minor-cycle hvor W er gått over fra 0 (00000) til 31 (111111).

Eksempel:

1. I ordren "13-14" som står i DL 4_5 er W = 0:

   m.c. 5: "13-14,0" ——> TS COUNT W = 00000
   " 6: "Set-up minor-cycle" W = 11111
   " 7: Operasjonen starter

2. I ordren "24-14" i DL 2_{19} er W = 2:

   m.c. 19: "24-14,2" ——> TS COUNT W = 01000
   " 20: "Set-up minor-cycle" W = 10000
   " 21: Venting W = 00000
   " 22: Venting W = 11111
   " 23: Operasjonen starter

I det förste eksemplet ser en at det ikke er venting, dvs. W = 0, og operasjonen starter i m.c. (m+2+W) = (5+2+0) = m.c. 7. I det andre eksemplet derimot må kontroll-organet vente i to minor-cycles för operasjonen kan starte, dette er også markert ved at W = 2. Operasjonen tar til m.c. (m+2+W) = (19+2+2) = m.c. 23.

3. Operasjonens varighet

Det som i förste rekke bestemmer en operasjons lengde er C-verdien i ordren. Som för nevnt kan C ha följende verdier og betydninger:

C = 0: kort operasjon; operasjonen varer bare én minor-cycle.

C = 1: lang operasjon; operasjonen varer inntil neste ordre blir overfört til TS COUNT, dvs. avhengig av T.

C = 2: dobbelt operasjon; operasjonen varer to minor-cycles.

C = 3: denne benyttes vanligvis ikke.

a. K o r t o p e r a s j o n . Operasjonen varer bare i den minor-cycle som den starter i, altså m.c. (m+2+W).

b. D o b b e l t o p e r a s j o n . Operasjonen varer den minor-cycle som den starter i, pluss neste, dvs. m.c. (m+2+W) og m.c. (m+3+W).

c. L a n g o p e r a s j o n . Operasjonen starter i m.c. (m+2+W) og varer til og med den minor-cycle hvor neste ordre blir overfört til TS COUNT, dvs. m.c. (m+T+2). En kan her tenke seg at kontroll-organet teller nedover på både T og W, operasjonen vil da starte i den minor-cycle som kommer etter den hvor W = 31 og vil vare til og med den minor-cycle som kommer like etter at T = 31. Hvis en har ordren "24-14 1,1,5" (dvs. W = 1, T = 5) i DL 1_2, får en:

m.c. 2: "24-14 1,1,5" ——> TS COUNT	W = 10000	T = 10100
" 3: Set-up	W = 00000	T = 00100
" 4: Venting	W = 11111	T = 11000
" 5: Operasjonen starter		T = 01000
" 6: Operasjon		T = 10000
" 7: "		T = 00000
" 8: "		T = 11111
" 9: Operasjonen avsluttes. Neste ordre ——> TS COUNT		

I dette tilfelle vil operasjonen vare i 5 minor-cycles. Generelt varer en lang operasjon i (T+1-W) minor-cycles. Hvis W = T, vil operasjonen vare bare i en minor-cycle, altså i virkeligheten en kort operasjon. På tilsvarende måte vil operasjonen være dobbelt hvis T = W + 1. Det kan også hende at W > T, i slike tilfeller blir T oppfattet som (T+32) og operasjonen vil da vare i (T+33-W) minor-cycles. Det som skjer i disse tilfellene er at når kontroll-organet teller nedover på W og T, vil T bli 31 för W. Nå kan ikke neste ordre bli hentet inn i kontroll-registeret för operasjonen starter, derfor blir T = 31 ignorert i dette tilfelle. Tellingen vil så fortsette inntil W = 31, da vil operasjonen starte i neste minor-cycle og vare til og med den minor-cycle som följer etter at T = 31 igjen.

Eksempel:

Ordren "8-25 1,5,1" står i DL 1_{30}:

	W	T
m.c. 30: "8-25 1,5,1" ——> TS COUNT	10100	10000
" 31: Set-up	00100	00000
" 0: Venting	11000	11111 (ignoreres)

m.c. 1:	Venting	01000	01111
" 2:	"	10000	10111
" 3:	"	00000	00111
" 4:	"	11111	11011
" 5:	Operasjonen starter		01011
" 6:	Operasjon		10011
" -:	osv.		
" -:	osv.		
" 31:	Operasjon		00000
" 0:	"		11111
" 1:	Siste operasjons-cycle. Neste ordre ⟶ TS COUNT		

En får altså her at:

Operasjonen starter i m.c. (m+2+W) = (30+2+5) = m.c. 5

" varer i (T+33-W) m.c. = (1+33-5) = 29 minor-cycles

Neste ordre hentes i m.c. (m+2+T) = (30+2+1) = m.c. 1

d. C = 3 . Dette er en kode som normalt ikke forekommer, men det kan hende at den er brukt enten ved et mistak eller tilsiktet, og i slike tilfelle er det av interesse å vite hva som vil skje.

Hvis W ≠ T, vil C = 3 ha samme virkning som en lang operasjon.

Hvis W = T, vil C = 3 öke den effektive verdien av T til (T+32) slik at operasjonen som betraktes som lang, vil vare i 33 minor-cycles, nemlig fra og med m.c. (m+2+W) til og med m.c. (m+34+W) eller (m+34+T).

4. Neste ordre

Neste ordre blir hentet inn i TS COUNT i m.c. (m+2+T). Denne overföringen av neste ordre til kontroll-registeret kan ikke foregå för operasjonen i ordren som står i TS COUNT er utfört, men den kan overföres samtidig med siste minor-cycle i operasjonen.

W = T: I ordren med kort eller lang operasjons-lengde vil, når W = T, neste ordre bli hentet inn i TS COUNT samtidig med at operasjonen starter, dvs. at også den lange operasjonen bare varer én minor-cycle. I en dobbelt operasjon kan ikke neste ordre overföres samtidig med starten av operasjonen, i m.c. (m+2+W), fordi da vil neste operasjons-cycle falle sammen med "Set-up Minor-cycle" for neste ordre. Derfor blir T i dette tilfelle oppfattet som (T+32) slik at neste ordre blir hentet til TS COUNT i m.c. (m+2+T) i neste major-cycle, altså en major-cycle (32 minor-cycles) etter at operasjonen starter.

W > T: Neste ordre kan, som nevnt, ikke overföres til TS COUNT för operasjonen i inneværende ordre i TS COUNT er utfört eller tidligst i siste operasjons-cycle. Derfor blir T i de tilfeller W > T alltid oppfattet som (T+32) slik at neste ordre hentes inn i m.c. (m+2+T) i neste major-cycle etter at operasjonen er satt i gang.

W < T: Neste ordre overföres til TS COUNT i m.c. (m+2+T) eller (T-W) minor-cycles etter at operasjonen er sluttet.

T-delen i ordren bestemmer altså i hvilken minor-cycle neste ordre skal overföres til TS COUNT, men forteller intet om hvorfra denne ordren skal hentes. Denne opplysningen gis av NIS, som angir i hvilken delay-linje neste ordre befinner seg. NIS er bare på 3 bits og kan derfor bare ha verdier fra 0-7. NIS 1-7 angir at det er DL 1-7, mens NIS 0 betyr at det er DL 8 det gjelder. Neste ordre kan derfor normalt aldri befinne seg i DL 9-12.

En har følgende regel: Neste ordre vil normalt befinne seg i m.c. (m+2+T) i den delay-linje som er angitt av NIS.

De eneste unntakene fra denne regelen er:

a. ordrer med Destinations 27 og 28 og Destination Trigger "2-24", dvs. test-ordrer.

b. ordrer med Destination 0.

T e s t - o r d r e r :

Ved de tre test-ordrene vil en ha et vegvalg i programmet slik at neste ordre kan hentes fra to steder avhengig av testene.

D 27: Hvis det ordet eller ordene som undersökes er positivt, overföres neste ordre til TS COUNT i m.c. (m+2+T). Er ordet som testes negativt, vil denne overföringen foregå i m.c. (m+3+T), dvs. at kontroll-organet automatisk forhöyer T med 1 i dette tilfelle. Det samme vil også forekomme hvis en tester flere ord samtidig og ett eller flere av disse ordene er negative.

D 28: Neste ordre overföres til TS COUNT i m.c. (m+2+T) hvis det som testes er lik null. Er noe av det som testes forskjellig fra null, blir T automatisk forhöyd med 1 slik at neste ordre kommer inn i TS COUNT i m.c. (m+3+T).

2 - 24: Denne ordren har som virkning at TIL-signalet blir sendt til D 28, slik at når TIL ikke er tilstede overföres neste ordre i m.c. (m+2+T), men når TIL er tilstede vil denne overföringen skje i m.c. (m+3+T).

Eksempel:
"3, 14-28, 0,2" står i DL 3_4:
(TS 14) = 0: neste ordre i (m+2+T), dvs. DL 3_8
(TS 14) ≠ 0: " " i (m+3+T), dvs. DL 3_9

" D e s t i n a t i o n " 0 :

I en ordre med D 0 blir NIS-delen ignorert og neste ordre blir hentet fra den lagerplass som er angitt ved source-verdien i ordren, hvis overföringen til D 0 finner sted i den minor-cycle som neste ordre blir hentet inn i TS COUNT.

Eksempel:

"0, 13-0, 1, 1, 4" står i DL 2_0:

	W	T
m.c. 0: "0, 13-0, 1, 1, 4" ⟶ TS COUNT	10000	00100
" 1: Set-up	00000	11000
" 2: Venting	11111	01000
" 3: Operasjonen starter		10000
" 4: Operasjon		00000
" 5: "		11111
" 6: " , neste ordre ⟶ TS COUNT		

En ser her at i m.c. 6, hvor TS COUNT er mottakelig for å ta inn neste ordre, finner også operasjonen fra foregående ordre sted. Denne operasjonen er "13-0", dvs. en overföring av innholdet i TS 13 til TS COUNT. Da denne er mottakelig, vil det altså være innholdet i TS 13 som hentes inn som ny ordre.

Hvis operasjonen i denne ordre ikke hadde vært lang, ville operasjonen bare vart i m.c. 3, slik at i m.c. 6 ville ikke operasjonen "13-0" utföres. Da ville ikke innholdet i TS 13 bli overfört til TS COUNT i m.c. 6, men neste ordre ville bli hentet som normalt, dvs. fra DL 8_6 (NIS 0 betyr DL 8).

En bör derfor gjöre til regel at alle ordrer som inneholder D 0, skal være lange, dvs. C = 1.

5. Tidsberegning

Ofte vil det være nödvendig å vite hvor lang tid DEUCE trenger for å utföre et bestemt program eller del av et program. Data til DEUCE mates inn via hullkort, denne innlesningen foregår med en maksimal hastighet av 200 kort pr. minutt. For å kunne holde denne hastigheten må ikke DEUCE bruke mer enn ca. 300 ms. på å utföre programmet for hvert enkelt kort,

derfor må programmereren vite den nöyaktige tiden programmet krever.

Hver enkelt ordre i DEUCE krever minst 2 minor-cycles, en minor-cycle for overföring av ordren til TS COUNT og en "Set-up minor-cycle". Dertil kommer den tiden kontroll-organet må vente för neste ordre kan hentes inn i TS COUNT, dette vil oftest være "T" minor-cycles, men i noen tilfeller vil det være "T + 32" minor-cycles. Ventetiden vil være avhengig av forholdet mellom W og T, og i noen tilfelle verdien av C.

T > W: I disse tilfellene vil ventetiden för neste ordre hentes inn være "T" minor-cycles slik at den tiden som ordren trenger er "T + 2" minor-cycles.

T = W: I de tilfellene C = 0 eller 1, dvs. kort eller lang operasjon, trenger ordren "T + 2" minor-cycles. Hvis C = 2, dobbelt operasjon, eller C = 3, vil T bli ökt til "T + 32" slik at ordren trenger "T + 34" minor-cycles.

T < W: Her vil alltid T bli ökt til "T + 32" og krever "T + 34" minor-cycles.

Eksempel:

1. Ordren "3,10-13,0,2" vil kreve "T + 2" m.c. eller 4 minor-cycles.
2. Ordren "2,16-12,4,0" vil kreve "T+32+2" m.c. eller 34 minor-cycles.
3. Ordren "4,18-17 d, 0,0" krever "T + 32 + 2" m.c. eller 34 minor-cycles.

K o d e - e k s e m p l e r

1. I eks. 3 på side 33 har en följende program:

7_{14} - 13
13 - 25
7_{15} - 26
13 - 7_{16}

Det förste som må gjöres med dette programmet för kodingen kan begynne, er at en bestemmer seg for hvor de enkelte ordrene skal lagres i hurtigminnet. Ordrene kan bare overföres til TS COUNT fra de 8 förste DL, derfor bör alle ordrene i et program helst lagres i disse DL med en gang. Hvis ordrene lagres et annet sted i maskinen, må de i tilfelle overföres til disse 8 DL för ordrene skal utföres. I dette tilfelle kan en for eksempel velge å plasere ordrene i DL 2. De kan naturligvis lagres hvor som helst innen DL 2, men en bör under fastsettelsene av lagerplassene for ordrene pröve å ordne dette slik at

det blir minst mulig ventetid, både til operasjonen skal finne sted og til hentingen av neste ordre. Med andre ord: En bör sörge for at W og T blir minst mulige.

Den förste ordren i programmet, "7_{14} - 13", må utföres i en bestemt minor-cycle, nemlig m.c. 14, fordi DL 7_{14} bare kan overföres i denne minor-cycle. For at ventetiden til denne overföringen skal være kortest mulig, bör denne ordren være lagret i m.c. 12, dvs. i DL 2_{12}, slik at m.c. 13 blir "Set-up Minor-cycle" og m.c. 14 blir operasjons-cycle. Neste ordre kan overföres til TS COUNT i samme minor-cycle, m.c. 14, og kan derfor lagres i DL 2_{14}. Den förste ordren vil altså ikke ha ventetid, dvs. W = T = 0.

Neste ordre som nå er plasert i DL 2_{14}, spesifiserer en operasjon på et kort register (TS 13). Disse korte registrene er som kjent tilgjengelige i alle minor-cycles, derfor kan operasjonene på disse utföres uten ventetid. I dette tilfelle betyr det at operasjonen kan utföres i m.c. 16 (mc. 15 er "Set-up Minor-cycle"). Neste ordre overföres til TS COUNT samtidig og må derfor være lagret i DL 2_{16}, dvs. at også den ordren har W = T = 0.

Ordren "17_{15} - 26" er nå plasert i DL 2_{16}, men operasjonen kan ikke utföres för i m.c. 15 fordi DL 7_{15} bare er tilgjengelig i denne minor-cycle. Her får en altså en ventetid på nesten en hel major-cycle, nærmere bestemt 29 minor-cycles. Neste ordre overföres samtidig, nemlig fra DL 2_{15}, slik at W = T = 29.

Den siste ordren som nå er lagret i DL 2_{15}, inneholder en overföring som må skje i m.c. 16, fordi DL 7_{16} bare kan motta opplysninger i denne minor-cycle. Den förste m.c. 16 er "Set-up minor-cycle", derfor kan ikke overföringen skje i denne m.c., men i m.c. 16 i neste major-cycle. Dette blir en ventetid fra m.c. 17 til m.c. 16, altså 31 minor-cycles. Neste ordre i programmet kan være lagret i DL 2_{16}, men denne plassen er allerede opptatt av ordren "7_{15} - 26" så en lagrer neste ordre i DL 2_{17}. Dette betyr at i den siste ordren blir W = 31 og T = 0 (eg. 32, men blir skrevet som 0. DEUCE vil i denne ordren automatisk forhöye T med 32, fordi $T < W$).

Når en har bestemt hvor de enkelte ordrene i programmet skal lagres, er kodingen av ordrene en enkel oppgave.

I dette tilfelle får en:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_{12}	7_{14} - 13	2	7	13	0	0	0	1	2
2_{14}	13 - 25	2	13	25	0	0	0	1	2
2_{16}	7_{15} - 26	2	7	26	0	29	29	1	31
2_{15}	13 - 7_{16}	2	13	7	0	31	0	1	34
2_{17}	Neste ordre	-	-	-	-	-	-	-	-
									69

Som en ser trenger dette programmet 69 minor-cycles, hvorav 61 minor-cycles er ventetid, dvs. minor-cycles som hverken er "Set-up"-cycles eller operasjons-cycles. Denne ventetiden kan reduseres betraktelig hvis en först overförer "b" og "d" fra DL 7_{14+15} til DS 19. Ved utregningen av "2 b-d" kan en så hente "b" og "d" fra DS 19_2 og DS 19_3 som er tilgjengelig i henholdsvis hver "like" og hver "ulike" minor-cycle. En kan da som eksempel ha följende program:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_{12}	7_{14} - 19 d	2	7	19	2	0	1	1	3
2_{15}	19_2 - 13	2	19	13	0	1	1	1	3
2_{18}	13 - 25	2	13	25	0	0	0	1	2
2_{20}	19_3 - 26	2	19	26	0	1	1	1	3
2_{23}	13 - 7_{16}	2	13	7	0	23	23	1	25
2_{16}	Neste ordre	-	-	-	-	-	-	-	-
									36

Dette programmet består av en ordre mer enn det förste, men til gjengjeld er det nesten dobbelt så raskt som dette. Dette siste programmet utförer operasjonen "X = 2 b-d" på raskest mulig måte i DEUCE, forutsatt at "b" og "d" er lagret i DL 7_{14+15} og resultatet "2 b-d" skal lagres i DL 7_{16}. Hvis en har mulighet for å lagre disse verdiene på andre plasser innen hurtig-minnet, kunne ventetiden elimineres helt. Hvis for eksempel "d" kan lagres i DL 7_{18} istedetfor DL 7_{15} og "2 b-d" lagres i DL 7_{20} istedetfor DL 7_{16}, vil en få följende program:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_{12}	7_{14} - 13	2	7	13	0	0	0	1	2
2_{14}	13 - 25	2	13	25	0	0	0	1	2
2_{16}	7_{18} - 26	2	7	26	0	0	0	1	2
2_{18}	13 - 7_{20}	2	13	7	0	0	0	1	2
2_{20}	Neste ordre	-	-	-	-	-	-	-	-
									8

Vi kaller dette e t o p t i m a l t - k o d e t p r o g r a m . Med o p t i m a l k o d i n g mener en at data og ordrer lagres slik at de er tilgjengelige med et minimum av ventetid.

2. Det neste eksemplet som skal kodes står også på side 33 (eks. 2). En begrenser seg også her til bare å benytte DL 2 til lagring av ordrer. Lösning "a" vil da bli slik:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_{30}	8_0 - 13	2	8	13	0	0	0	1	2
2_0	8_1 - 25	2	8	25	0	31	31	1	33
2_1	8_1 - 25	2	8	25	0	30	31	1	33
2_2	8_2 - 25	2	8	25	0	30	31	1	33
2_3	8_2 - 25	2	8	25	0	29	31	1	33
2_4	8_2 - 25	2	8	25	0	28	31	1	33
2_5	13 - 8_3	2	13	8	0	28	31	1	33
2_6	Neste ordre	-	-	-	-	-	-	-	-
									200

Merknader:

Den fjerde ordren i programmet kan overföres til TS COUNT i m.c. 1, men nå er DL 2_1 allerede benyttet som lagerplass derfor er denne ordren plasert i nærmeste ledige lagerplass som er DL 2_2. Tilsvarende betraktninger gjelder for de etterfölgende ordrer i programmet. Disse er derfor hele tiden lagret i nærmeste ledige lagerplass i DL 2. Det er naturligvis intet til hinder for at også andre DL kan benyttes til lagring av ordrene, således kan den fjerde ordren lagres i DL 3_1, og tilsvarende for de övrige ordrene. Den eneste forandring som har betydning for total-tiden, er hvis "neste ordre" lagres i DL 3_3 istedetfor DL 2_6. Da vil total-tiden bli redusert med 3 minor-cycles til 197 minor-cycles.

Dette programmet trenger i alt 200 minor-cycles, hvorav 186 minor-cycles til ren venting, dvs. minor-cycles hvori ingen operasjoner finner sted. Den neste lösningen vil som en ser i det följende, være en del raskere.

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_{30}	8_0 - 13	2	8	13	0	0	0	1	2
2_0	8_1 - 25 d	2	8	25	2	31	0	1	34
2_2	8_1 - 25 d	2	8	25	2	29	31	1	33
2_3	8_2 - 25	2	8	25	0	29	31	1	33
2_4	13 - 8_3	2	13	8	0	29	31	1	33
2_5	Neste ordre	-	-	-	-	-	-	-	-
									135

Merknader:

Hvis en lagrer "neste ordre" i m.c. 3 i en annen DL, f.eks. DL 3_3, vil total-tiden reduseres til 133 minor-cycles, dvs. ingen vesentlig innsparing av tiden. Noe större innsparing oppnår en hvis en bytter om ordre nr. 2 og ordre nr. 4. Ordren "8_2 - 25" vil da lagres i DL 2_0 og se slik ut: "2, 8 - 25, 0,0,0,0" og ordren "8_1-25 d" lagres i DL 2_3 slik: "2, 8 - 25, 2, 28, 31, 0". Disse to ordrene vil nå kreve henholdsvis 2 og 33 minor-cycles, og total-tiden for hele programmet blir redusert med 32 m.c. til 103 minor-cycles

Slik som programmet står i eksemplet trenger det 135 minor-cycles, men hvis en bare bytter om to ordrer vil en altså kunne redusere denne tiden til 103 minor-cycles.

I mange programmer vil en ofte kunne redusere total-tiden betydelig ved å forandre rekkefölgen av enkelte ordrer. I lösningen "c" vil total-tiden bli ytterligere redusert. Også her antar en at alle ordrene skal lagres i DL 2. En får da följende program:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_{30}	8_0 - 13	2	8	13	0	0	0	1	2
2_0	8_1 - 21 d	2	8	21	2	31	0	1	34
2_2	21_2 - 25 l(5m.c.)	2	21	25	1	0	4	1	6
2_8	13 - 8_3	2	13	8	0	25	25	1	27
2_3	Neste ordre	-	-	-	-	-	-	-	-
									69

Også denne lösningen trenger forholdsvis lang tid, nemlig 69 minor-cycles. Av disse er 56 minor-cycles ventetid. Ved en liten omarbeiding av dette programmet kan en ytterligere redusere total-tiden. En får da:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_{31}	8_1 - 21 d	2	8	21	2	0	1	1	3
2_2	30 - 13	2	30	13	0	0	0	1	2
2_4	21_2 - 25 l(5m.c.)	2	21	25	1	0	4	1	6
2_{10}	8_0 - 25	2	8	25	0	20	20	1	22
2_0	13 - 8_3	2	13	8	0	1	1	1	3
2_3	Neste ordre	-	-	-	-	-	-	-	-
									36

Denne siste lösningen er den absolutt raskeste lösning av dette problemet, her er ventetiden redusert til det minst mulige; en har et optimalt-kodet program.

3. Kodingen av eksempel 11, lösning b på side 36 blir slik, hvis en velger å lagre ordrene i DL 2. Den förste ordren plaseres i DL 2_0.

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_0	19 - 21 d	2	19	21	2	0	1	1	3
2_3	21 - 22 l(4m.c.)	2	21	22	1	1	4	1	6
2_9	19 - 22 d	2	19	22	2	1	2	1	4
2_{13}	21_3 - 13	2	21	13	0	0	0	1	2
2_{15}	21_2 - 25 l(3m.c.)	2	21	25	1	1	3	1	5
2_{20}	13 - 16	2	13	16	0	0	0	1	2
2_{22}	Neste ordre	-	-	-	-	-	-	-	-
									22

Merknader:

Ordren "21 - 22 l (4 m.c.)" kan starte i m.c. 5, men når den skal vare i 4 minor-cycles vil den da vare til og med m.c. 8, som er en "like" minor-cycle. Hvis denne ordren skal avsluttes i en slik minor-cycle, må en av följende to betingelser være tilstede for å få et korrekt resultat:

a) "TCB" må være "på", slik at de to ordrene i DS 21 blir betraktet som to atskilte ord à 32 bits.

b) Innholdet av DS 21_2 må ha P 32 = 0, dvs. være positiv. Hvis TCB er "av" og DS 21_2 har en "1" i P 32, vil automatisk et ord med bare enere bli addert inn i DS 21_3 til slutt, dvs. at DS 21_2 blir omgjort til et langt ord med 64 bits.

De samme betraktninger gjelder for den tredje ordren, "19-22 d", også her må en vente over en minor-cycle slik at operasjonen avsluttes i en "ulike" cycle.

4. Eksemplet överst på side 38:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_{30}	8_0 - 14	2	8	14	0	0	0	1	2
2_0	12_0 - 15	2	12	15	0	30	31	1	33
2_1	25 - 8_1	2	25	8	0	30	31	1	33
2_2	23 - 14 l(8m.c.)	2	23	14	1	0	7	1	9
2_{11}	25 - 8_2	2	25	8	0	21	22	1	24
2_3	23 - 14 l(8m.c.)	2	23	14	1	0	7	1	9
2_{12}	25 - 8_3	2	25	8	0	21	22	1	24
2_4	23 - 14 l(8m.c.)	2	23	14	1	0	7	1	9
2_{13}	25 - 8_4	2	25	8	0	21	22	1	24
2_5	Neste ordre	-	-	-	-	-	-	-	-
									167

Dette programmet trenger 167 minor-cycles, herav går 128 minor-cycles med til ventetid. Dette programmet kan lages betydelig raskere hvis en plaserer den konstanten som benyttes ved ekstraheringen i DL 12_2, istedetfor i DL 12_0. Dessuten lönner det seg å lagre resultatene midlertidig i QS 17 (eller QS 18) og når alle resultatene er klare, overföre hele QS 17 til DL 8_{1-4}. En får da:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_{30}	8_0 - 14	2	8	14	0	0	0	1	2
2_0	12_2 - 15	2	12	15	0	0	0	1	2
2_2	25 - 17_1	2	25	17	0	1	1	1	3
2_5	23 - 14 1(8m.c.)	2	23	14	1	0	7	1	9
2_{14}	25 - 17_2	2	25	17	0	2	2	1	4
2_{18}	23 - 14 1(8m.c.)	2	23	14	1	0	7	1	9
2_{27}	25 - 17_3	2	25	17	0	2	2	1	4
2_{31}	23 - 14 1(8m.c.)	2	23	14	1	0	7	1	9
2_8	25 - 17_0	2	25	17	0	2	2	1	4
2_{12}	17_1 - $8_1$1(4m.c.)	2	17	8	1	19	22	1	24
2_4	Neste ordre	-	-	-	-	-	-	-	-
									70

Selv om dette siste programmet inneholder en ordre mer enn den første versjonen, er dette siste programmet betydelig raskere enn det første. Dette kommer vesentlig av at en benytter et 4-ords register som midlertidig lagerplass.

5. Eksempel 2 på side 39:

Dette programmet vil se slik ut:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
4_{30}	10_0 - 13	4	10	13	0	0	31	1	33
4_{31}	10_1 - 26	4	10	26	0	0	1	1	3
4_2	+ 13 - 27	4	13	27	0	0	5	1	(7)-8
4_9	10_0 - 13	4	10	13	0	21	21	1	(23)
4_0	10_2 - 26	4	10	26	0	0	1	1	(3)
4_3	+ 13 - 27	4	13	27	0	0	1	1	(3-4)
4_{10}	10_2 - 13	4	10	13	0	22	25	1	27
4_5	10_1 - 26	4	10	26	0	26	29	1	31
4_4	+ 13 - 27	4	13	27	0	0	1	1	3(4)
4_7	10_2 - 14	4	10	14	0	25	2	1	36
4_8	10_1 - 14	4	10	14	0	23	1	1	(35)
4_6	10_0 - 14	4	10	14	0	24	3	1	(37)
4_{11}	14 - 8_0	4	14	8	0	19	20	1	22
4_1	Neste ordre	-	-	-	-	-	-	-	-
									163

Merknader:

En har i dette programmet konsentrert lagringen av programmet til DL 4_{30-11}, uten at det dermed er blitt noe senere enn om hver enkelt ordre hadde vært plasert optimalt.

I et program som dette hvor det er vegvalg, vil aldri alle ordrene bli utfört for hvert kort, derfor kan heller ikke de tidene de enkelte ordrene trenger, summeres sammen til totaltiden. I slike tilfelle beregner en hvor lang tid de enkelte vegene i programmet trenger og sier at total-tiden er det samme som den lengste vegen. I dette programmet er det 4 veger, to av disse trenger 163 minor-cycles, mens de to övrige trenger bare 131 m.c., altså en major-cycle mindre.

Dette programmet kan gjöres betydelig raskere og kortere; raskere ved å benytte et fire-ords register og kortere ved å bruke en litt annen framgangsmåte.

En kan for eksempel ha:

	Programmet	Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
4_{30}	10_0 - 18_0 l(3m.c.)	4	10	18	1	0	2	1	4
4_2	18_1 - 13	4	18	13	0	1	1	1	3
4_5	18_2 - 26	4	18	26	0	3	3	1	5
4_{10}	+ 13 - 27	4	13	27	0	0	0	1	2-(3)
4_{12}	18_1 - 14	4	18	14	0	3	4	1	6
4_{13}	18_2 - 14	4	18	14	0	3	3	1	(5)
4_{18}	14 - 13	4	14	13	0	0	0	1	2
4_{20}	18_0 - 26	4	18	26	0	3	2	1	4
4_{24}	+ 13 - 27	4	13	27	0	0	0	1	2-(3)
4_{26}	14 - 8_0	4	14	8	0	4	4	1	6
4_{27}	18_0 - 8_0	4	18	8	0	3	3	1	(5)
4_0	Neste ordre	-	-	-	-	-	-	-	-
									34

Dette programmet er kodet helt optimalt, dvs. det er laget så raskt som mulig. Dessuten inneholder dette programmet to ordrer mindre enn den förste versjonen.

6. Multiplikasjon

Det som en i første rekke må passe på ved koding av multiplikasjon, er at en ikke rører produktet eller erstatter innholdet av TS 16 för 65 minor-cycles etter at multiplikasjonen er satt i gang ved ordren "0 - 24". Denne ordren må dessuten alltid startes i en "ulike" minor-cycle.

Hvis denne ordren starter i n.c. (m), vil multiplikasjonen vare til og med n.c. (m) to major-cycles senere, dvs. den varer i to major-cycles pluss en minor-cycle, dette er det samme som 65 minor-cycles.

Eksemplet på side 43 vil da se slik ut:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_{28}	$30 - 21_2$	2	30	21	0	0	0	1	2
2_{30}	$10_0 - 16$	2	10	16	0	0	0	1	2
2_0	$10_1 - 21_3$	2	10	21	0	31	31	1	33
2_1	$0 - 24$	2	0	24	0	0	31	1	33
2_2	$30 - 29$	2	30	29	0	1	0	1	34
2_4	$21 - 10_2$d	2	21	10	2	28	29	1	31
2_3	Neste ordre	-	-	-	-	-	-	-	-
									135

Dette programmet trenger 135 minor-cycles, men derav bare 65 til multiplikasjonen. Over halvparten av tiden går derfor med til å plasere multiplikator og multiplikand för multiplikasjonen og resultatet etter. Ved å benytte et dobbelt-register kan denne tiden reduseres endel, slik:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_{30}	$10_0 - 19$ d	2	10	19	2	0	1	1	3
2_1	$19_3 - 21_3$	2	19	21	0	0	1	1	3
2_4	$19_2 - 16$	2	19	16	0	0	0	1	2
2_6	$30 - 21_2$	2	30	21	0	0	0	1	2
2_8	$0 - 24$	2	0	24	0	1	0	1	34
2_{10}	$30 - 29$	2	30	29	0	0	29	1	31
2_9	$21 - 10_2$d	2	21	10	2	23	24	1	26
2_3	Neste ordre	-	-	-	-	-	-	-	-
									101

Merknader:

I ordren "0-24" som er lagret i DL 2_8, er W=1, fordi denne ordren må starte i en "ulike" minor-cycle. Multiplikasjonen starter altså i m.c. 11 og varer til og med m.c. 11 to major-cycles senere. Overföringen av DL 21 kunne derfor foregå i m.c. 12 og 13 i denne major-cycle, men da er ikke DL 10_{2+3} tilgjengelig. (En kunne overföre DS 21 i m.c. 11 og 12, men da ville DS 21_3 som er höyeste halvpart, bli overfört til DL 10_{11} og DS 21_2, dvs. laveste halvpart, til DL 10_{12}). Hvis en derfor velger å plasere resultatet i DL 10_{12+13}, vil siste ordre bli slik: "2, 21-10, 2, 1, 2" og neste ordre hentes fra DL 2_{13}. Total-tiden for programmet blir da på 79 minor-cycles, altså bare 14 minor-cycles mer enn det som går med til selve multiplikasjonen.

7. Multiplikasjon med fortegn

I eksemplet foran er ikke tatt hensyn til fortegn, slik at resultatet bare blir riktig i det tilfelle at begge faktorene er positive. I neste eksempel, side 44, har en innarbeidet i programmet en korreksjonsrutine slik at resultatene fra dette program vil være korrekte uansett fortegn.

I dette programmet trenger en ikke noen "dummy"-ordre, som bare har som oppgave å bruke tiden. Her må en sörge for at de ordrene som kommer för operasjonen "$13-23_3$" bruker så lang tid at multiplikasjonen er ferdig, dvs. 65 minor-cycles.

En får her:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_0	$30 - 21_2$	2	30	21	0	0	0	1	2
2_2	$14 - 21_3$	2	14	21	0	1	1	1	3
2_5	$0 - 24$	2	0	24	0	0	31	1	33
2_6	$14 - 27$ (−)	2	14	27	0	0	2	1	4-(5)
2_{10}	(+) $30 - 13$	2	30	13	0	0	1	1	3
2_{11}	$16 - 13$	2	16	13	0	0	0	1	(2)
2_{13}	$16 - 27$ (−)	2	16	27	0	0	0	1	2(3)
2_{15}	(+) $30 - 29$	2	30	29	0	0	1	1	3
2_{16}	$14 - 25$	2	14	25	0	0	0	1	(2)
2_{18}	$13 - 23_3$	2	13	23	0	21	21	1	23
2_9	$21 - 19$ d	2	21	19	2	0	1	1	3
2_{12}	Neste ordre	-	-	-	-	-	-	-	-

8. Divisjon

En kan kode eks. 1 på side 50 på følgende måte:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_5	$14 - 21_3$	2	14	21	0	0	0	1	2
2_7	$1 - 24$	2	1	24	0	0	31	1	33
2_8	$30 - 29$	2	30	29	0	1	0	1	34
2_{10}	$21_2 - 16$	2	21	16	0	0	0	1	2
2_{12}	Neste ordre	-	-	-	-	-	-	-	-
									71

Merknad:

En må her passe på at ordren "1-24" starter i en ulik m.c., dvs. i m.c. 9. Dessuten må aldri dividenden settes i DS 21_3 i den "ulike" minor-cycle som kommer umiddelbart för den "ulike" minor-cycle som ordren "0-24" starter i.

Det neste eksemplet, hvor en også er interessert i den eksakte resten, kan kodes på denne måten:

Programmet		Kodene							
Lager-plass	Ordre	N	S	D	C	W	T	G	Tid i m.c.
2_3	$16 - 21_3$	2	16	21	0	1	1	1	3
2_6	$14 - 16$	2	14	16	0	0	0	1	2
2_8	$1 - 24$	2	1	24	0	1	0	1	34
2_{10}	$30 - 29$	2	30	29	0	0	31	1	33
2_{11}	$22 - 21_3$	2	22	21	0	0	0	1	2
2_{13}	$16 - 22_3$	2	16	22	0	0	0	1	2
2_{15}	$22_3 - 15$	2	22	15	0	0	0	1	2
2_{17}	$21_2 - 14$	2	21	14	0	1	1	1	3
2_{20}	Neste ordre	-	-	-	-	-	-	-	-
									81

Merknad:

Her kan en ikke foreta korreksjoner med hensyn til resten för divisjonen er ferdig, dvs. 66 minor-cycles etter denne er startet.

VII. LÖKKER OG ORDREMODIFISERING

En av de store og viktige kjennetegn for DEUCE og liknende elektroniske maskiner som arbeider etter et internt lagret program, er deres evne til å gjenta deler av programmet med små forandringer av enkelte ordrer. En slik del av programmet som gjentas, kalles gjerne en l ö k k e . Som regel blir en eller flere ordre i en slik lökke forandret för hver gjentagelse, dette kalles for o r d r e m o d i f i s e r i n g . Lökker med modifisering er typiske trekk i de fleste programmer for DEUCE.

Som kjent blir alle opplysninger lagret i DEUCE som serier av 32 bits. DEUCE vil derfor ikke se noen forskjell på tall og ordrer slik at regneorganet ikke vil ha noen vanskelighet med å utföre regneoperasjoner på ord som inneholder ordrer. Denne evnen til å forandre ordrer i programmet ved å utföre aritmetiske operasjoner på dem, benytter en seg av ved ordremodifisering.

Oppbyggingen og organiseringen av en lökke kan best illustreres ved et enkelt eksempel:

> En önsker å lage et program som skal lagres i DL 1, og som skal sörge for at de andre DL, DL 2 - DL 12, blir nullstillet eller tömt.

En kan löse dette problemet ved å lage et program bestående av 11 ordrer av typen "30 - D 1 (32 m.c.)", hvor D varierer fra 2 til 12.

Dette programmet vil se slik ut:

```
30 -  2 1 (32 m.c.)
30 -  3 1 (32 m.c.)
30 -  4 1 (32 m.c.)
30 -  5 1 (32 m.c.)
30 -  6 1 (32 m.c.)
30 -  7 1 (32 m.c.)
30 -  8 1 (32 m.c.)
30 -  9 1 (32 m.c.)
30 - 10 1 (32 m.c.)
30 - 11 1 (32 m.c.)
30 - 12 1 (32 m.c.)
```

Som en ser er disse ordrene helt like bortsett fra D-verdiene i ordrene som stadig vokser med 1. Det har tidligere vært nevnt at DEUCE kan regne på ordrer, det er derfor rimelig å anta at maskinen lettvint vil kunne utföre denne forandringen eller modifiseringen av ordrene selv. En trenger da bare å lagre i DEUCE den originale utgaven av den ordren som

skal modifiseres, denne originalordren overföres til TS 13 og overföres derfra til TS COUNT. Etter at original-ordren er utfört på denne måten, adderes en modifiseringskonstant til innholdet av TS 13, slik at den ordren som står der blir modifisert. I dette tilfellet er modifiseringskonstanten lik (P 10) eller laveste bit i "D". Skrevet som en ordre ser modifiseringskonstanten slik ut "0,0-1,0,0,0 X". Den modifiserte ordren i TS 13 behöver ikke overföres til en lang delaylinje för den skal utföres, men kan overföres direkte til TS COUNT ved hjelp av "Destination C".

En får da följende programlökke:

```
(a)      I - 13                  (I = 30-2 1 (32 m.c.))
(b)  ┌-> 13 -  0
(c)  |   (30 -  2 1 (32 m.c.))
(d)  |    k - 25                (k = (0,0-1,0,0,0 X) eller P10)
     └----┘
```

Merknader:

(a) "I" som kan være lagret hvor som helst innen hurtigminnet (ikke DL 2 - DL 12), er originalordren, dvs. ordren slik den blir utfört förste gang för den blir modifisert. "I" blir ved denne ordren overfört til TS 13, hvor den siden skal modifiseres ved addisjon.

(b) Innholdet i TS 13 overföres til TS COUNT som neste ordre. Förste gangen vil innholdet være det samme som originalordren, neste gang vil innholdet være "30-3 1 (32 m.c.)" osv.

(c) Ordren, som er overfört fra TS 13, utföres. Förste gang bevirker dette tömming av DL 2, neste gang tömming av DL 3 osv.

(d) Konstanten "k" som kan være lagret hvor som helst i hurtigminnet (ikke DL 2 - DL 12), adderes til innholdet av TS 13 slik at ordren som befinner seg der får foröket sin D-verdi med 1. Neste ordre er (b).

Ordre (c) i dette programmet danner hoveddelen av lökken fordi det er denne ordren som sörger for at den önskede operasjon blir utfört. En kaller derfor gjerne denne delen av lökken for o p e r a s j o n s - d e l e n .

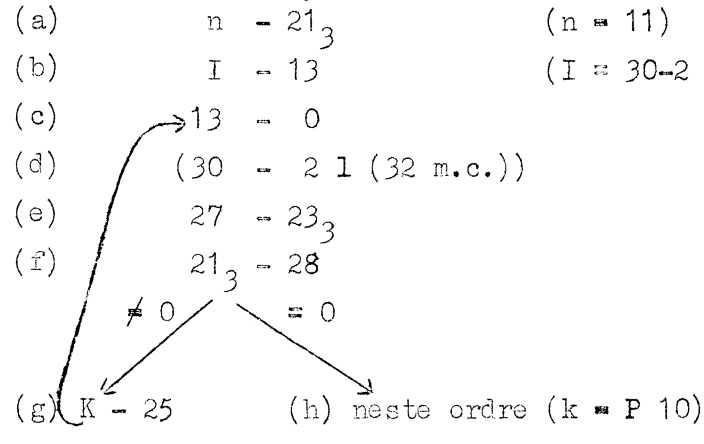
För DEUCE går inn i en lökke må som regel en del ordrer utföres; disse ordrene har til oppgave å sörge for at de riktige startbetingelsene er tilstede. I dette programmet består denne i n n l e d n i n g e n av bare en ordre, nemlig ordre (a) som sörger for at TS 13 har det riktige startinnhold.

Den siste ordren i dette programmet bevirker at ordren i TS 13 blir modifisert ved at en m o d i f i s e r i n g s k o n s t a n t blir addert til innholdet i TS 13.

Dette programmet mangler imidlertid forelöpig en meget viktig del. Programmet vil nå gå i en lökke, men det er ikke sörget for en utgang fra lökken etter at den er gjennomlöpt det önskede antall ganger. Den delen av programmet som skal fortelle når lökken er gjennomlöpt nok, kalles ofte t e l l i n g og t e s t i n g .

Lökken i dette programmet må gjennomlöpes 11 ganger fordi ordren "30-D 1 (32 m.c.)" må utföres 11 ganger. En kan benytte et ord i hurtigminnet som telleverk og addere 1 til dette ordet hver gang en går gjennom lökken. Når innholdet av ordet er lik 11, er lökken gjennomlöpt 11 ganger og en skal forlate lökken.

Hvis en benytter DS 21_3 som telleverk, får en följende program:



Merknader:

(a) Tallet 11 settes inn i telleverket (DS 21_3)

(b) Originalordren overföres til TS 13

(c) Innholdet i TS 13 overföres til TS COUNT som neste ordre

(d) Nullstilling av delay-linjene 2 - 12

(e) Subtraher 1 i telleverket

(f) Test innholdet i DS 21_2. Hvis dette er forskjellig fra null, er ikke lökken gjennomlöpt det önskede antall ganger og maskinen henter inn (g) som neste ordre.
Er DS 21_3 = 0, er lökken gjennomlöpt n (11) ganger og en går ut av lökken til (h)

(g) Modifisering av ordren i TS 13

(h) Utgang fra lökken

Telling og testing i lökken kan utföres på mange forskjellige måter. Den tellingen som en har benyttet her, vil i dette tilfelle ikke være den mest hensiktsmessige. I dette og mange andre programmer kan en med fordel bruke "Joe"-delen i ordren.

Som nevnt för har ikke "Joe" noe virkning på den operasjonen som en ordre spesifiserer, derfor benyttes normalt ikke denne delen i ordren. Ved modifisering derimot, kan "Joe" ofte være svært nyttig fordi den

ligger så gunstig plasert i ordre-ordet, nemlig mellom "Wait" og "Time" (P 22 - P 25). Da "Joe" bare opptar 4 bits, vil den aldri kunne ha verdier större enn 15 (1111). Har en derfor en ordre med Joe = 15 og adderer en ener til Joe i denne ordren, vil dette bevirke at Joe blir 0 og T-verdien blir forökt med 1, fordi en ener blir mentet over i P 26 som er laveste bit i T. Dette kan en benytte seg av ved at en sörger for at original-ordren har en Joe-verdi som er avpasset slik at når en adderer 1 til Joe hver gang lökken gjennomlöpes, vil dette resultere i mente over til T etter det nödvendige antall addisjoner. På denne måten kan en ordne en utgang av lökken.

I dette eksemplet kan addisjonen av 1 til Joe utföres samtidig med modifiseringen av D-verdien i ordren. Ordren "30 - D 1 (32 m.c.)" skal utföres 11 ganger, en må derfor sette startverdien til Joe slik at den 11. gangen ordren utföres er Joe = 0 og T forökt til T + 1. En får da följende program (lagret i DL 1):

(a) 1_1 I - 13 1_3 = I: = 30-2 1 (32 m.c.)(Joe = 6)
(b) 1_4 ⟶13 - 0
(c) Q_6 (30 - 2 1 (32 m.c.))
(d) 1_7 k - 25 1_9 = k = (P_{10} + P_{22})
1_8 neste ordre

Merknader:

(a) Ordren som er lagret i DL 1_1, sörger for å overföre innholdet av DL 1_3 til TS 13. Innholdet av DL 1_3 er originalordren, som har Joe = 6.

(b) Innholdet av TS 13 overföres til TS COUNT som neste ordre

(c) Denne ordren som overföres til TS COUNT fra TS 13 i m.c. 6, må være kodet med W og T som om den var lagret i m.c. 6 i en DL, derfor er ordren betegnet "Q_6" eller "Quasi" 6. Normalt vil denne ordren hente inn DL 1_7 som neste ordre. Den 11. gangen vil imidlertid T-verdien i ordren være ökt med 1 på grunn av mente fra Joe, slik at neste ordre kommer fra DL 1_8.

(d) Innholdet av DL 1_9 adderes til ordren i TS 13. Ordet i DL 1_9 inneholder (P_{10} + P_{22}), slik at ved addisjon av dette ordet til ordren, blir D og Joe begge forökt med 1.

Kodene for dette programmet vil se slik ut i DL 1:

DL 1:
m.c. 1: 1, 1-13, 0, 0, 1
" 3: 1, 30-2, 3, 0 (6), 31 (originalordren)
" 4: 0, 13-0, 0, 0, 0

m.c. 7: 1, 1-25, 0, 0, 27

" 8: neste ordre

" 9: 0, 0-1, 0, 0 (1), 0 X (P 10 + P 22)

I originalordren har en satt C = 3 og ikke C = 1, for hvis C = 1 vil ordren 11. gangen den blir utfört se slik ut: "1, 30-12, 1, 0 (0) 0" dvs. ordren varer bare en minor-cycle slik at bare DL 12_8 blir nullstillet. C = 3 vil imidlertid foruten å bevirke at ordren blir lang også ha den virkning at når W = T, blir T automatisk oppfattet som T + 32 slik at operasjonen varer i 33 minor-cycles. Dette betyr at DL 12_8 blir nullstillet to ganger, men dette spiller selvfölgelig ingen rolle. En kan sette C = 1, men da må Joe settes lik 5, slik at ordren blir utfört 12 ganger. De förste 11 gangene vil da bevirke at DL 2 - DL 12 blir nullstillet, mens den 12. gang ordren blir utfört, nullstilles TS 13.

Programmet slik det er kodet her, trenger 679 minor-cycles eller 21,7 ms. Uten lökke trenger programmet bare 363 minor-cycles eller 11,6 ms. For å löse det samme problemet vil altså et program laget som lökke, trenge dobbelt så lang tid som et program som er kodet rett fram. Til gjengjeld inneholder det förste programmet bare halvparten så mange ordre. Dette er noe som gjelder generelt: Et program kodet som en lökke, vil alltid være langsommere enn det tilsvarende program uten lökke, men til gjengjeld vil det trenge mindre plass i minnet. Ved kodingen må en da foreta en vurdering av hva som teller mest: tid eller plass. Endel programmer må imidlertid på grunn av sin natur, alltid kodes som lökker.

I eksemplet under "Addisjon" i kap. V (side 33) er det laget et program for addisjon av hvert fjerde ord i DL 10. En rekke av ordrene i dette programmet er helt like bortsett fra at de skal utföres i ulike minor-cycles. Nå er det W i ordren som bestemmer når operasjonen skal utföres, derfor kan en ved å addere 4 til W i ordren, få modifisert denne slik at den adderer hvert fjerde ord i DL 10. Som en lökke kan en f.eks. lage dette programmet slik:

2_{22} I - 13 2_{25} = I = (10_0 - 22_2) Joe = 10

2_{26} 10_{28} - 21_2

2_{28} → 13 - 0

Q_{30} (10_0 - 22_2)

2_{23} k - 25 2_{27} = k = (P_{19} + P_{22})

2_{24} 21_2 - 12_0

2_0 neste ordre

Kodet vil disse ordrene og konstantene se slik ut i DL 2:

m.c. 22: 2, 2-13, 0, 1, 2
" 23: 2, 2-25, 0, 2, 3
" 24: 2, 21-12, 0, 6, 6
" 25: 2, 10-22, 0, 0 (10) 23 (originalordren)
" 26: 2, 10-21, 0, 0, 0
" 27: 0, 0-0, 0, 4 (1) 0 X (P 19 + P 22)
" 28: 0, 13-0, 0, 0, 0

Programmet slik det her er kodet, trenger 234 minor-cycles eller 7,5 ms. Hvis en derimot koder programmet slik det står på side 33, trenger det bare litt over en major-cycle eller ca. 1 ms.

I de to eksemplene på lökker som er gjennomgått, er TS 13 brukt ved modifisering av ordrene. En kunne naturligvis også ha benyttet DS 21_2 eller DS 21_3 ved modifiseringen. For å unngå at TS 13 eller DS 21 blir opptatt på denne måten er det innebygd i DEUCE en anordning for a u t o m a t i s k o r d r e m o d i f i s e r i n g . Ved å benytte denne kan ordrer modifiseres uten å benytte TS 13 eller DS 21, dessuten trengs ikke spesielle modifiseringsordrer og -konstanter.

Den vanligste form for ordremodifisering består av addisjon av 1 til W i en ordre slik at denne blir utfört i suksessive minor-cycles. I ordrer som spesifiserer dobbelte operasjoner, önsker en ofte å addere 2 til Wait. Disse modifikasjonsformene og en del andre kan utföres ved automatisk ordremodifisering.

Automatisk ordremodifisering i DEUCE utföres bare i forbindelse med QS 17 og QS 18. Ordrer som er lagret i QS 17 eller QS 18, blir automatisk modifisert ved ordrene "17-0" eller "18-0". Modifiseringen skjer etter regler som skal spesifiseres senere.

Selve operasjonen i ordrene "17-0" og "18-0" er som vanlig for ordrer som inneholder Destination 0. Overföring av ordre fra QS 17 eller QS 18 foregår bare i de tilfellene hvor operasjonen utföres samtidig med at neste ordre hentes inn i TS COUNT, dvs. når C = 0, må W være lik T, når W ≠ T, må C være lik 1.

Selve modifiseringen utföres bare i förste operasjonscycle slik at bare et ord i en QS kan modifiseres ved en ordre. Varer "17-0" eller "18-0" f.eks. fem minor-cycles, vil modifiseringen bare foregå i den förste av disse fem operasjonscycles.

Eksempel:

Ordren "17-0" er lagret i DL 2_{30}. Ved å variere C, W og T kan en få följende operasjoner:

(a) 17 - 0,0,0,0 : 17_0 (modifisert) ⟶ TS COUNT
(b) 17 - 0,0,0,1 : 17_0 modifiseres; $(NIS)_1$ ⟶ TS COUNT
(c) 17 - 0,1,0,0 : 17_0 (modifisert) ⟶ TS COUNT
(d) 17 - 0,1,0,1 : 17_0 modifiseres; 17_1 ⟶ TS COUNT
(e) 17 - 0,1,0,4 : 17_0 (modifisert) ⟶ TS COUNT
(f) 17 - 0,2,0,0 : 17_0 modifiseres; $(NIS)_0$ ⟶ TS COUNT
(g) 17 - 0,2,0,1 : 17_0 modifiseres; 17_1 ⟶ TS COUNT
(h) 17 - 0,2,0,4 : 17_0 modifiseres; $(NIS)_4$ ⟶ TS COUNT

En merker seg her at hvis en önsker å modifisere den ordren som skal overföres til TS COUNT, må en, når operasjonen er kort (C = 0), sette W = T (eks. a) eller når operasjonen er lang, må differensen mellom W og T være lik 0 (eks. c) eller et eksakt multiplum av 4 (eks. e). I det siste tilfelle må en være oppmerksom på at modifiseringen og overföringen til TS COUNT ikke utföres i samme minor-cycle; i eks. (e) ovenfor blir ordren i QS 17_0 modifisert i m.c. 0, men overfört til TS COUNT först i m.c. 4. I dobbelte operasjoner kan aldri det ordet som modifiseres, bli overfört til TS COUNT. I eks. (f) ovenfor blir ordet i QS 17 modifisert i m.c. 0, i neste m.c. 0 (en minor-cycle senere) blir det ordet som er lagret i m.c. 0 i den delay-linjen NIS angir, overfört til TS COUNT. Hvis C = 3, kan det ordet som modifiseres, bare overföres til TS COUNT i de tilfellene differensen mellom W og T er lik 0 eller et eksakt multiplum av 4, og dessuten blir ikke ordet modifisert i den samme minor-cycle som det blir overfört til TS COUNT.

Arten av modifiseringen avhenger av kodingen av Destination C - ordrene. Det er verdien av P 1, P 2, P 3, P 4 og P 15 i ordrene som bestemmer modifiseringen. Modifiseringen er ikke den samme i QS 17 og QS 18.

Verdien av de to laveste sifrene i "17 - 0"-ordren bestemmer hvilke siffer som skal adderes til ordet i QS 17 etter dette skjema:

P_1 og P_2 i "17-0"	Siffer addert til QS 17
00 (= 0 x P_1)	+ P_5
10 (= 1 x P_1)	+ P_{10}
01 (= 2 x P_1)	+ P_{17}
11 (= 3 x P_1)	+ P_{18}

Hvis P 15 er tilstede i "17-0", dvs. "17-0,1" eller "17-0,3", vil 1 i P 3 ha den virkning at også P 22 blir addert og 1 i P 4 resulterer i bytte av fortegn, slik at sifrene blir subtrahert istedenfor addert ved modifiseringen.

Verdien av P 3 og P 4 har ingen virkning når P 15 ikke forekommer, dvs. "17-0,0" eller "17-0,2".

Hvis en betegner de 4 laveste bits i "17-0"-ordren for "y", har en dette fullstendige skjema for modifisering i QS 17:

	Siffer addert under modifiseringen	
$Y = (P_1P_2P_3P_4)$	17 - 0,0	17 - 0,1
0000 ($= 0 \times P_1$)	$+ P_5$	$+ P_5$
1000 ($= 1 \times P_1$)	$+ P_{10}$	$+ P_{10}$
0100 ($= 2 \times P_1$)	$+ P_{17}$	$+ P_{17}$
1100 ($= 3 \times P_1$)	$+ P_{18}$	$+ P_{18}$
0010 ($= 4 \times P_1$)	$+ P_5$	$+ P_5 \quad + P_{22}$
1010 ($= 5 \times P_1$)	$+ P_{10}$	$+ P_{10} \quad + P_{22}$
0110 ($= 6 \times P_1$)	$+ P_{17}$	$+ P_{17} \quad + P_{22}$
1110 ($= 7 \times P_1$)	$+ P_{18}$	$+ P_{18} \quad + P_{22}$
0001 ($= 8 \times P_1$)	$+ P_5$	$- P_5$
1001 ($= 9 \times P_1$)	$+ P_{10}$	$- P_{10}$
0101 ($= 10 \times P_1$)	$+ P_{17}$	$- P_{17}$
1101 ($= 11 \times P_1$)	$+ P_{18}$	$- P_{18}$
0011 ($= 12 \times P_1$)	$+ P_5$	$- P_5 \quad - P_{22}$
1011 ($= 13 \times P_1$)	$+ P_{10}$	$- P_{10} \quad - P_{22}$
0111 ($= 14 \times P_1$)	$+ P_{17}$	$- P_{17} \quad - P_{22}$
1111 ($= 15 \times P_1$)	$+ P_{18}$	$- P_{18} \quad - P_{22}$

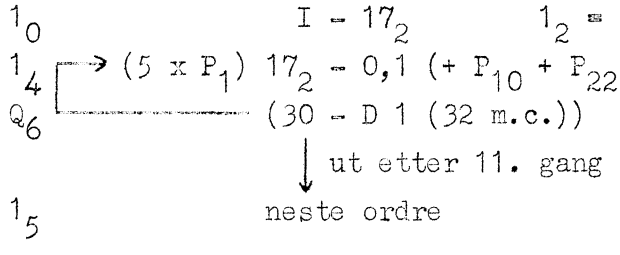
For modifisering i QS 18 med ordren "18-0" gjelder det samme skjema med bytte av fortegn.

En må være oppmerksom på at når det adderes både i Wait og Joe, vil det ikke opptre noen mente fra Wait til Joe fordi denne blir undertrykket. Dette gjelder da de "17-0"- eller "18-0"-ordrene hvor $(7 \times P_1) \geq y \geq (4 \times P_1)$ eller $(15 \times P_1) \geq y \geq (12 \times P_1)$. Dessuten er det viktig å vite at når et ord i en QS blir modifisert slik at "over-flow" inntreffer, vil dette resultere i en mente over til etterfölgende ord i QS. Dette skjer imidlertid så uregelmessig at en aldri kan benytte seg av det, f.eks. til modifisering av NIS.

Ofte önsker en å utföre en telling i lökken, dette kan utföres i QS 17 eller QS 18 ved at "17-0" eller "18-0" er kodet slik at det ordet som modifiseres ikke overföres til TS COUNT. En benytter da gjerne QS 17 med telling "oppover" (addisjon av 1) og QS 18 ved telling "nedover" (subtraksjon av 1).

En kan telle med P_5, P_{10}, P_{17} eller P_{18}, dette avgjöres som nevnt av de to laveste bits i "Destination 0"-ordren. Valget av hvilken posisjon en vil telle i gjör imidlertid at valget av NIS blir noe begrenset fordi P 2, foruten å ha betydning for hvilket siffer som skal adderes, som kjent er laveste bit i NIS. Skal en derfor telle med P_5 eller P_{10}, må NIS ha verdiene 0, 2, 4 eller 6, og på tilsvarende måte må NIS ha verdiene 1, 3, 5 eller 7 hvis en önsker å addere med P_{17} eller P_{18}.

I det förste eksemplet i dette kapitlet, nullstilling av delay-linjene DL 2 - DL 12, skal en modifisere D-verdien i ordren "30-D 1 (32 m.c.)" samtidig med telling i Joe, dvs. en skal addere $(P_{10} + P_{22})$ til ordren. Av modifiseringsskjemaet finner en ut at slik addisjon kan en få i QS 17 ved ordren "17-0, 1" med y $(P_1\text{-}P_4) = (5 \times P_1)$ eller i QS 18 ved ordren "17-0, 1" med y $= (13 \times P_1)$. I dette tilfelle velger en QS 17 og får da följende program:



Kodene for programmet blir slik i DL 1:

m.c. 0: 1,1 - 17,0,0,2
" 2: 1,30 - 1,3,29(5)28
" 4:(1)2,17 - 0,1,0,0
" 5: neste ordre

I dette programmet bör en merke seg följende:

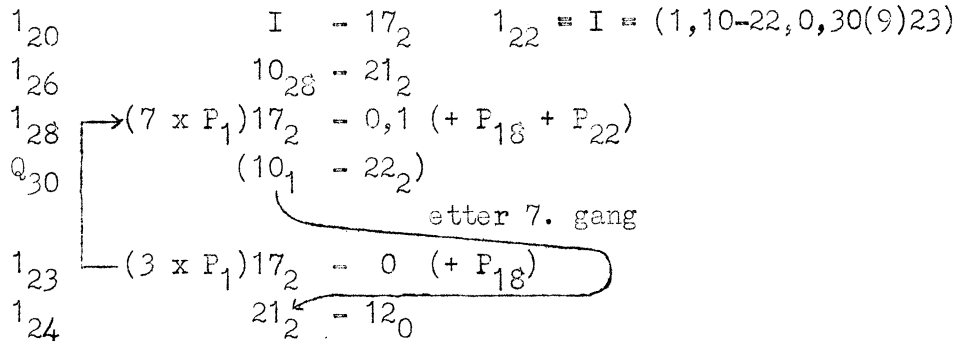
a. Originalordren må kodes slik: "1,30 - 1,3,29(5)28" fordi ordren blir modifisert også för den blir utfört förste gangen. Den förste gangen ordren blir utfört ser den altså slik ut: "1,30 - 2,3,29(6)28" og den siste gangen (den 11.) vil ordren se slik ut: "1,30 - 12,3,29(0)29".

b. En markerer hvordan modifiseringen skal utföres ved å sette verdien av y (P 1 - P 4) foran "Destination 0"-ordren i programmet og sette hvilke sifre som skal adderes eller subtraheres bak ordren, altså slik: "(5 x P_1) 17_2 - 0,1 (+ P_{10} + P_{22})".

c. Ved kodingen av "17-0" eller "18-0" ordrer kodes y-verdien som to tall: Verdien av P 2 - P 4 som NIS og verdien av P 1 i parentes foran NIS. I dette tilfelle har en: "(1)2,17 - 0,1,0,0".

Ved å benytte automatisk ordremodifisering har en her spart inn modifiseringsordren og -konstanten, men tiden blir omtrent den samme.

I det neste eksemplet, addisjon av hvert fjerde ord i DL 10, består modifiseringen i å addere 4 til Wait. En kan ikke ved hjelp av en ordre automatisk få addert P 19 til Wait (P 19 er det samme som 4 i Wait), derfor må en hvis en önsker å benytte seg av den automatiske ordremodifiseringen, foreta denne i to etapper. Programmet kan da f.eks. se slik ut:



DL 1:

m.c. 20: 1,1-17,0,0,4
" 22: 1,10-22,0,30(9)23
" 23: (1)1,17-0,0,1,3
" 24: 1,21-12,0,6,6
" 26: 1,10-21,0,0,0
" 28: (1)3,17-0,1,0,0

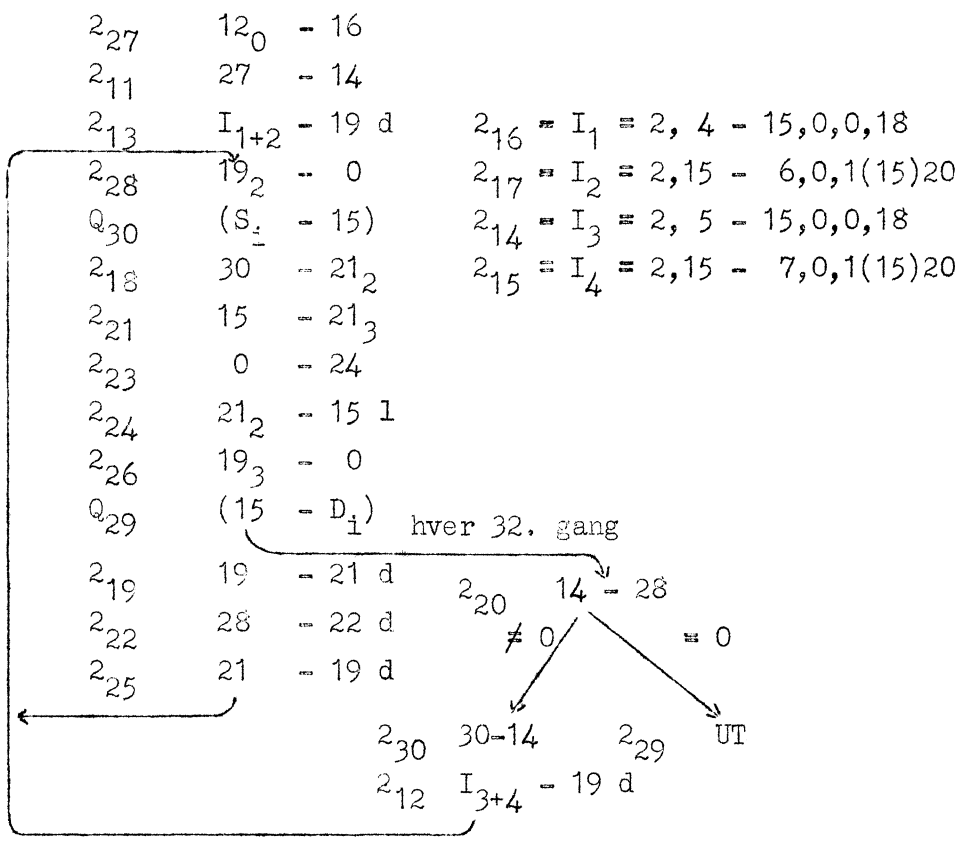
For å vise nærmere hvordan modifisering kan utföres skal en gå gjennom to eksempler. En anbefaler leseren å arbeide seg grundig gjennom disse programmene.

Eksempel

1. Multipliser alle tallene i DL 4 og DL 5 med den samme multiplikator (i DL 12_0) og plaser produktene i DL 6 og DL 7, slik at (4_0) x (12_0) ⟶ 6_0, (4_1) x (12_0) ⟶ 6_1 osv. En forutsetter at alle tall er positive hele tall som ikke er större enn at hvert enkelt produkt får plass i ett ord.

   En skal löse dette problemet både uten og med automatisk ordremodifisering.

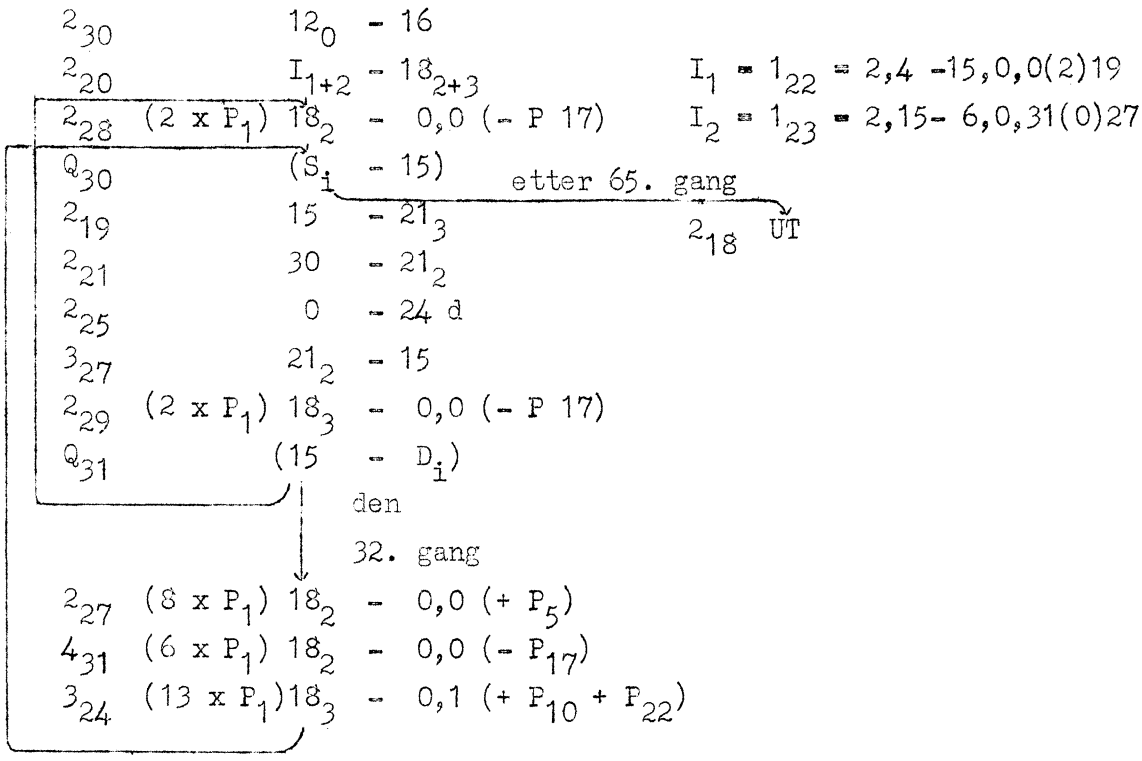
U t e n a u t o m a t i s k o r d r e m o d i f i s e r i n g :



DL 2:

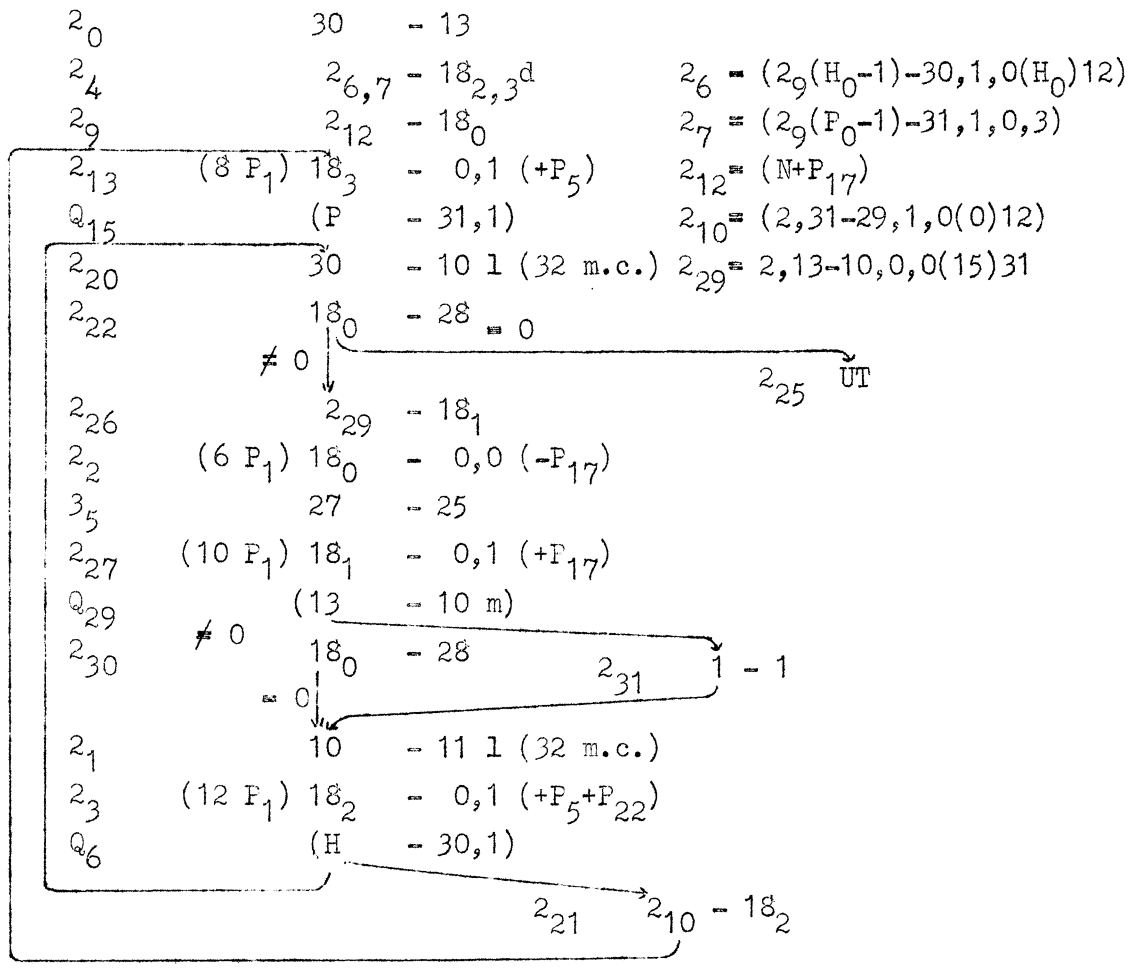
m.c. 11: 2,27 - 14,0,0,0
" 12: 2, 2 - 19,2,0,14
" 13: 2, 2 - 19,2,1,13
" 14: 2, 5 - 15,0,0,18
" 15: 2,15 - 7,0,1(15)20
" 16: 2, 4 - 15,0,0,18
" 17: 2,15 - 6,0,1(15)20
" 18: 2,30 - 21,0,0,1
" 19: 2,19 - 21,2,0,1
" 20: 2,14 - 28,0,0,7
" 21: 2,15 - 21,0,0,0
" 22: 2,28 - 22,2,0,1
" 23: 2, 0 - 24,0,0,31
" 24: 2,21 - 15,1,1,0
" 25: 2,21 - 19,2,0,1
" 26: 0,19 - 0,0,1,1
" 27: 2,12 - 16,0,3,14
" 28: 0,19 - 0,0,0,0
" 29: neste ordre
" 30: 2,30 - 14,0,0,12

M e d a u t o m a t i s k o r d r e m o d i f i s e r i n g :



	DL 2	DL 3	DL 4
m.c. 18:	neste ordre		
" 19:	2,15 - 21,0,0,0		
" 20:	2, 2 - 18,d,0,6		
" 21:	2,30 - 21,0,1,2		
" 22:	2, 4 - 15,0,0(2)19		
" 23:	2,15 - 6,0,31(0)27		
" 24:		(1)6,18 - 0,1,1,4	
" 25:	3, 0 - 24,2,0,0		
" 26:			
" 27:	4,18 - 0,0,1,2	2,21 - 15,0,31,0	
" 28:	1,18 - 0,0,0,0		
" 29:	1,18 - 0,0,0,0		
" 30:	2,12 - 16,0,0,20		
" 31:			3,18 - 0,0,1,23

2. Lag et program for lagring av de N første naturlige tall (1,2,3,4 ..... N) på magnettrommelen, idet en starter lagringen i m.c. 0 i spor P_0/H_0.



DL 2 DL 3

m.c. 0: 2,30 - 13,0,0,2
" 1: 2,10 - 11,1,1,0
" 2: 3,18 - 0,0,0,1
" 3: 6,18 - 0,1,1,1
" 4: 2,2 - 18,2,0,3
" 5: 2,27 - 25,0,0,20
" 6: $2,(H_0-1)$ - 30,1,0(H_0)12
" 7: $2,(P_0-1)$ - 31,1,0,3
" 9: 2,2 - 18,0,1,2
" 10: 2,31 - 29,1,0(0)12
" 12: (N x P_{17})
" 13: 4,18 - 0,1,0,0
" 20: 2,30 - 10,1,1,0

m.c. 21: 2,2 - 18,0,19,22
" 22: 2,18 - 28,0,0,1
" 25: Neste ordre
" 26: 2,2 - 18,0,1,6
" 27: 5,18 - 0,1,0,0
" 29: 2,13 - 10,0,0(15)31
" 30: 2,18 - 28,0,0,1
" 31: 2,1 - 1,0,0,0