DASK - en 1. generations computer

af

H. B. Hansen

4. Del

Operativsystemet på Dask

Da Dask blev født var operativsystemerne endnu i deres vorden. Første gang man hører ordet "operativsystem" er i 1956 i forbindelse med et hjælpeprogram til at holde styr på alle programmerne hos General Motors i USA. Ordet var ukendt i Danmark i 1957, hvor programmellet til Dask blev udviklet. Vi anskuede nærmest operativsystemet som en hjælp til programmering. Og der var faktisk tale om en hjælp. Vi kunne fx indlæse ordrer med decimale adresser, mens svenskerne måtte bruge sedecimale adresser til Besk. Vi havde tæt kontakt til svenskerne, idet Dask som tidligere nævnt skulle have været en kopi af Besk. Svenskerne kaldte deres hjælpeprogrammer til indlæsning af tal og kode, samt til trykning, for "Normalläget", som vel nærmest kan oversættes til "Standardtilstanden" på dansk. Det er måske derfor ikke så mærkeligt, men typisk dansk, at de tilsvarende programmer på Dask fik navnet "Normallejet". I tidens løb kom der en del versioner af dette normalleje, men det jeg vil omtale her er den første version, Normalleje 1.

Normallejets faste del

Med et så lille lager som Dasks var det meget vigtigt at normallejet fyldte så lidt som muligt. Derfor var det opdelt i en fast og en variabel del. Den faste del var lagret i hac 1984 - 2047, og fyldte altså kun 64 hac, hvilket netop er 1 kanal på tromlen. Dette er naturligvis ikke tilfældigt, men skyldtes at den faste del også lå på tromlen (på kanal 22) for det tilfælde at en klodrian kom til at ødelægge den faste del af normallejet med en kodefejl (der var ingen som helst lagerbeskyttelse). Så kunne normallejet retableres ved at udføre ordrerne:

```
 22 A 1C

1984 A 1D
```

Man prikkede dem binært ind på kontrolbordet og udførte dem én ad gangen på trin, eller indlæste en koldstartstrimmel.

Den variable del af normallejet skulle kun være inde i ferritlageret under indlæsning og udlæsning og var ellers lagret på tromlen. Der var derfor to indhop i den faste del, det hentede hhv. indlæseprogrammet og udlæseprogrammet fra tromlen. Når man ikke brugte den variable del kunne pladsen udnyttes til andre formål, typiske til arrays af data (fx matricer ved løsning af lineære ligninger).

Den faste del bestod ellers af følgende dele:

- **De flydende registre**. Som tidligere nævnt var flydende regning programmeret på Dask. Man arbejdede med to former for flydende tal: oppakkede og pakkede. De oppakkede flydende tal bestod af en taldel, der fyldte en hec (40 bits), og en eksponent, der fyldte en adressedel (12 bits). Dette gav mulighed for et regneinterval på $10^{-309} < |x| < 10^{308}$, med en præcision på ca. 12 decimale cifre. I den pakkede form fyldte et flydende tal kun én hec; talintervallet var det samme, men taldelen var 12 bits kortere, så præcisionen var ikke så stor, kun ca. 10 decimale cifre, men dog fuldt på højde med moderne maskiners præcision. De flydende registre hed FMD, FAR og

FMR og kunne lagre et oppakket flydende tal hver. De lå i ferritlageret fra celle 1996 til 2007 incl.

- **Pakning og oppakning af flydende tal.** Der var fire småprogrammer: FAR til AR pakket, AR pakket til FMD, AR pakket til FAR og AR pakket til FMR. Desuden rådede koderen over en biblioteksekvens, FR1, der kunne udføre de fire regningsarter med flydende oppakkede tal, men denne sekvens var ikke en del af normallejet.
- **De permanente konstanter.** Dette var en meget nyttig del af normallejet, hvor der var lagret nogle bitmønstrer i 5 hac. Ved at udnytte hel- og halvcelleordrer kunne man faktisk generere i hvert fald ni forskellige konstanter ud fra disse bitmønstrer, nemlig 2^{-11}, 2^{-31},-1 (det mindste Dask-tal), $1-2^{-39}$ (det største Dask-tal), 2^{-19}, 2^{-39}, 0, 2^{-21} og 2^{-1}.
- **Etikettecellerne.** Dette var måske nok den vigtigste del af den faste del af normallejet. Etikettecellerne var 8 hac (2008 til og med 2015), der fungerede nogenlunde som det vi i dag kalder basisregistre. De arbejdede sammen med indlæseprogrammet under indlæsning af ordrer. Den ydre repræsentation af en ordre kunne forsynes med et *etikettemærke* imellem indeksmærkningen og operationskoden. Dette etikettemærke kunne være 8, 9, A, B, C, D, E eller F. Når indlæseprogrammet stødte på et etikettemærke skete der det, at indholdet af den tilhørende etikettecelle blev lagt til ordrens adresse. Ved at lagre passende konstanter i etikettecellerne kunne man altså modificere adresserne i ordrerne under indlæsningen. Det var en fast konvention at hac 2008 altid skulle indeholde begyndelsesadressen for det program der var ved at blive indlæst; derfor kunne lokale adresser kodes relativt til begyndelsesadressen. Dette blev kaldt 0A8-form. Når først en ordre var indlæst var dens adresse absolut, så systemet var mere primitivt end de basisregistre, man finder i fx IBM 360.
- **Korrektion ved fortegnsskift.** Hvis man skifter fortegn på Dasktallet -1 får man spild, fordi Daskintervallet kun går til $1-2^{-39}$. Derfor var der et lille underprogram i Normallejet der sørgede for at udskifte spild med $1-2^{-39}$. Det benyttede man hvis man var usikker på om man kunne komme til at skifte fortegn på -1.

Normallejets variable del

Den variable del af Normalleje 1 var lagret på tromlen, men optog helcellerne 1536 til 1982 når den var "inde". Der var to dele: trykning og kontrolprogrammer (hec 1536-1790) og indlæsning (hec 1792-1982). Trykkeprogrammet var et efter tidens standard særdeles fleksibelt program, der kunne trykke heltal og flydende tal på mange forskellige måder. Man kunne via parametre angive trykkefeltets størrelse, antal decimaler, om tallet skulle trykkes på eksponentform, om tallet skulle trykkes til højre eller venstre i trykkefeltet, og endda om man ville have tal trykt med et konstant antal betydende cifre, sådan at antallet af decimaler blev formindsket når tallene blev store, og da om trykningen skulle ske med komma under komma. Man kaldte det at *trimme* trykprogrammet. Det var derfor let at fremstille output i form af tabeller. Derimod var der ingen særlige faciliteter til trykning af tekst, hvilket understreger den tids opfattelse af at computere var en avanceret form for regnemaskiner.

Indlæseprogrammet var på en måde mere primitivt. Det kunne ganske vist indlæse både ordrer og tal, men ikke tekst, og tallene skulle kodes efter et bestemt system fordi der kun måtte indgå de sedecimale tegn 0-9 og A-F. Dette skyldtes at indlæsning foregik fra en 5-kanals strimmellæser, hvor den ene kanal var reserveret. I princippet skulle hvert tal omgives af tre bogstaver:

- Et fortegn (C for plus og D for minus)
- Et komma (F for heltal i hac, A for heltal i hec, B for DASK-tal i hac, C for DASK-tal i hec, D for flydende pakket tal og E for flydende oppakket tal)
- Et sluttegn (altid A, blev dog udeladt ved komma A og F)

Heltallet +25 skulle fx kodes som C25A hvis det skulle lagres i en hec, og som C25F hvis det skulle lagres i en hac. DASK-tallet -0.67342 skulle kodes DB67342A eller DC67342A afhængigt af om det

skulle opfattes som en tal i hac eller hec, og det flydende pakkede tal 3.7913 kom til at se således ud: C3D7913A.

Indlæsning af ordrer, ydre etiketter

Indlæsning af ordrer skete på grundlag af en ekstern repræsentation der er beskrevet tidligere. Ved hjælp af etikkettecellerne var det muligt at "oversætte" denne repræsentation til binær form i een passage. Dette var meget vigtigt, eftersom indlæsemediet var en perforeret strimmel, og det ville være meget upraktisk hvis strimlen skulle indlæses mere end een gang (og på grund af det lille lager ville man ikke afse plads til at lagre selve kildeteksten i lageret). Det forudsætter imidlertid at man kan manipulere med etikettecellerne under selve indlæsningen, og det var da også muligt, nemlig ved hjælp af de såkaldte **ydre etiketter**. En ydre etiket er en kommando til normallejet; den har ordreform, men indeksmærket er et E.

Jeg skal ikke her gå i detaljer med de ydre etiketters funktion, nogle få eksempler må vise princippet. Indlæseprogrammet opererede med begrebet *den løbende adresse*, der betød adressen på den hac hvor næste ordre ville blive indlæst. Man kunne sætte den løbende adresse til fx 100 ved hjælp af den ydre etikette

```
100 E 3
```

Her er "operationskoden" 3 altså en kommando til normallejet, og næste ordre vil blive indlæst til hac 100. Hvis næste ordre hænder sig at være første ordre i et delprogram - også kaldet en sekvens - så ville det være praktisk at sætte etikkettecelle 8, hac 2008, til 100 (jfr. 0A8-formen). Det kunne ske ved hjælp af den ydre etikette

```
0 E 0
```

Denne kommando virker sådan at adressen (det der står til venstre for E'et) bliver lagt til løbende adresse, og resulatet lagres i etikettecelle 8. Nullet til højre for E'et er altså operationskoden, som angiver etikettecelle 8.

Hvis sekvensen i celle 100 skal kunne kaldes af andre sekvenser, er det praktisk at dens begyndelsesadresse findes i en eller anden etikettecelle, fx etikettecelle A, hac 2010. Det kunne fx ske ved hjælp af den ydre etikette

```
0 E 1A
```

hvor 1-tallet siger at der er tale om en etikettecelle forskellig fra etikettecelle 8, og A'et udpeger etikettecellen, og som adderer "adressen" 0 og løbende adresse og lagrer resultatet i etikettecelle A.

Endelig vil jeg nævne etiketteoperationen 50, der betød: stop indlæsning; ved tryk på start hop til etikettekommandoens adresse. Denne operation får vi brug for i det følgende eksempel.

Et eksempel på et program med flydende tal

Eksemplet er hentet i "Lærebog i Kodning for Dask", der vist må siges at være den allerførste lærebog i programmering på dansk. Mit eksemplar er udgivet af Regnecentralen i september 1958, men før denne tid fandtes der et duplikeret hæfte der indeholdt nogenlunde det samme stof, blot langt mere rodet og "teknisk", uden ret mange eksempler. Det er ud fra det duplikerede hæfte jeg har lært at kode til Dask, men det har jeg nu foræret til Teknisk Museum i Helsingør.

Givet de to 20-dimensionale vektorer A = {a0,a1,...,a19} og B = {b0,b1,...,b19}. Dan det skalære produkt A*B.

Vi springer indlæsningen eller beregningen af elementerne i A og B over, og koncentrerer os om selve

beregningen af skalarproduktet. Først må vi bestemme os til hvor de to vektorer skal lagres. Lad os vedtage at A har startadressen 100 og B har startadressen 200. Det i'te element i A er altså lagret i hec 100 + 2i, og det i'te element i B er lagret i hec 200 + 2i. Det skalære produkt bliver et enkelt flydende tal, som passende kan lagres i hec 98. Vi udfører de flydende beregninger ved hjælp af standardprogrammet til flydende regning, FR1, som vi lægger med startadresse i hac 714, altså et godt stykke efter tallene, så er vi sikre på at program og data ikke karambolerer. For at kunne bruge FR1 hensigtsmæssigt, må vi se på biblioteksspecifikationen for FR1.

REGNECENTRALEN
DANSK INSTITUT FOR MATEMATIKMASKINER

DASK - BIBLIOTEKSSPECIFIKATION

SEKVENS-BETEGNELSE
FR 1
side 1/7

Kodet af studie-kreds d. 1-2 1957
Indkørt af JHH WH d. 10-10 1957
Udgivet d. -6-1958

Flydende tal:

addition, multiplikation, division

Indhops-adresser	Udhops-adresser	Indgang	Udgang	Max. ordre-antal	Køretid min.	Køretid max.
0A8	45A8	C(FMD) = y C(FAR) = x	x+y → FAR & FMD	39	32 AT (19 AT)	ca.39 AT (46 AT)
2A8	20A8 37A8 39A8		x+y → FAR	31	24 AT (11 AT)	ca.31 AT (38 AT)
50A8	37A8 39A8		$\frac{x}{y}$ → FAR	23	31 AT (31 AT)	31 AT (38 AT)
57A8	37A8 39A8	C(FAR) = x C(FMR) = y	xy → FAR	19	$21\frac{1}{2}$ AT (21 AT)	$21\frac{1}{2}$ AT ($28\frac{1}{2}$ AT)

Kodelængde	0 - 62	Undersekvenser	Ingen
Begyndelsesadresse	Lige	Arbejdsceller	I sekvensen, samt 1998v, 2002v, 2006v
Grundparametre	Ingen	Perm. konstanter	C(2039)
Programparametre	Ingen		

Her står alt hvad der er værd at vide for en bruger af FR1. Bemærk at kodelængden er 0-62, dvs. ialt 63 hac; det må man vide for ikke at komme til at lægge andre programmer eller data oven i FR1. Endvidere fremgår det at begyndelsesadressen skal være lige, så det er altså OK at lægge FR1 med

begyndelsesadresse 714, som planlagt. Indhopsadresserne er angivet relativt til starten af sekvensen, altså på 0A8- form, og hele sekvensen er programmeret på 0A8-form. FR1 skal bruges som et underprogram, der bliver kaldt af vores vektorprogram, og det vil derfor være praktisk at fylde startadressen for FR1 i en eller anden etikettecelle, fx etikettecelle 9. Begyndelsen af papirstrimlen får da følgende udseende:

```
714 E 3 Sæt løbende adresse til 714
0 E 19 Sæt 0 + løbende adresse = 714 i hac 2009
0 E 0 Sæt 0 + løbende adresse = 714 i hac 2008
FR1 Her indkopieres strimlen med FR1
```

Bemærk: kommentarerne til højre for etikettekommandoerne skulle **ikke** hulles med, men det vidste hulledamen!

Nu kunne man jo passende lægge selve beregningsprogrammet bag efter FR1, dvs. i forlængelse af FR1. Det får så begyndelsesadresse 714 + 63 = 777, men ved snedig brug af etikettecelle 8 behøver man faktisk ikke at vide det; man kunne nemlig fortsætte strimlen således:

```
0 E 0      Sæt 0 + løbende adresse (= 777) i hac 2008
PROGRAM    Her indkopieres selve programmet
0 E8 50    Stop, hop til 0A8 i programmet (dvs. 777)
```

Så er vi endelig klar til at afsløre hvordan vektormultiplikationsprogrammet ser ud. Det er programmeret i 0A8-form, og det forudsættes at FR1 starter i 0A9. Igen skal man huske at kommentarerne til højre for selve ordrerne ikke blev hullet med; de er der kun til glæde for en menneskelig læser. Det var altså ikke sådan som i vore dage, hvor kommentarerne springes over af oversætteren, det havde man ikke fundet på i 1957. Ligeledes skulle de relative adresser til venstre for ordrerne springes over af hulledamen; de er der kun for at programmøren kan finde ud af at anføre de rigtige adresser i fx hopordrer, jfr. ordren i relativt 10. Kommentarerne så heller ikke ud som her, for Algol kom først i 1960, og først da blev tildelingsoperatoren := opfundet. Man brugte en pil, fx 0 --> AR i stedet for AR := 0. Efter programteksten har jeg kommenteret nogle af ordrerne nærmere.

```
0 1 A8 50 AR := 0, hop til næste ordre
1 2021 A 16 FMD := 0
2 40 A 55 IRC := 40, dvs indeks i = 20 (hec)
3 2046 C 55 IRC := IRC - 2
4 100 C 40 AR := a[i] flydende pakket
5 2031 A 16 FMR := a[i] flydende oppakket
6 200 C 40 AR := b[i] flydende pakket
7 2026 A 16 FAR := b[i] flydende oppakket
8 57 A9 16 FAR := a[i]*b[i] flydende oppakket
9 0 A9 16 FAR := FMD := FMD + FAR fl. oppakket
10 3 A8 53 Hop til relativt 3 hvis i <> 0
11 2016 A 16 AR := a*b flydende pakket
12 98 A 08 Hec 98 := resultatet
13 0 A8 30 Stop. Ved start: en gang til
```

Ordren i hac 0A8 er et typisk kodetrick; i stedet for at hente et nul til AR, udnytter man variantsystemet på Dask. Et ubetinget hop har grundformen 10, men når man lægger 40 til fås 0 til AR inden ordrens udførelse. På denne måde sparer man to ting: en celle med et nul og en lageracces til denne celle. Ganske vist findes der allerede et nul i den faste del af normallejet, nemlig hac 2039, så den første ordre i programmet kunne have været 2039 A 60, der betyder "sæt 0 i AR og addér derefter hac 2039"; men at spare lageraccessen var også af betydning på en maskine der var så relativt langsom som Dask, og alle kodere satte en ære i at gøre deres programmer så hurtige som overhovedet muligt.

Den næste ordre, 2021 A 16, er et sekvenshop til den faste del af normallejet, der hvor der ligger en kodestump der kan konvertere AR som flydende pakket tal til FMD som flydende oppakket tal. Sådanne konverteringer var nødvendige når man regnede med flydende tal, fordi FR1 regnede med oppakkede tal.

Der er talrige af den slags konverteringer i programmet (5A8, 7A8 og 11A8, hvor 11A8 konverterer fra oppakket til pakket form). Adresserne i disse ordrer er absolutte, og der er ingen som helst lagerkontrol, så hvis man anførte en gal adresse måtte man selv tage skraldet.

Fra 2A8 starter en løkke. Der er 20 elementer i vektorerne, men de er lagret i helceller, som har de lige adresser og derfor springer med 2. Når man skal tælle til 20 putter man følgelig 40 i et indeksregister, her indeksregister C. Planen er at tælle baglæns, fra 40 med spring på 2 ned til 0. Man talte så vidt muligt baglæns på Dask, hvilket skyldtes at man havde en betinget hopordre på om et indeksregister var 0.

Nu skal der trækkes 2 fra indeksregister C; men der var ingen mulighed for negative adresser i normalleje 1. Derfor kommer der i 3A8 en lidt mystik ordre; den har den absolutte adresse 2046 og er indeksmærket med C. Den resulterende adresse bliver derfor 2046 plus indholdet af indeksregister C, hvilket - hvis indeksregister C fx har værdien 40 - bliver til 2086. Den højeste lageradresse er 2047, så 2086 er slet ingen lageradresse! Nu var det imidlertid sådan at adressearitmetikken i Dask skete modulo lagerstørrelsen, altså modulo 2048, og 2086 modulo 2048 er 38, altså 2 mindre end 40. De 2046 i adressen på ordren i 3A8 repræsenterer altså -2. Reglen er let nok: når man skal trække noget fra en adresse tæller man baglæns fra 2048: 2047 står for -1, 2046 står for -2, 2045 for -3 o.s.v.

Løkken strækker sig ned til 10A8, hvor der hoppes til 3A8 så længe IRC <> 0. Inde i løkken sker der følgende:

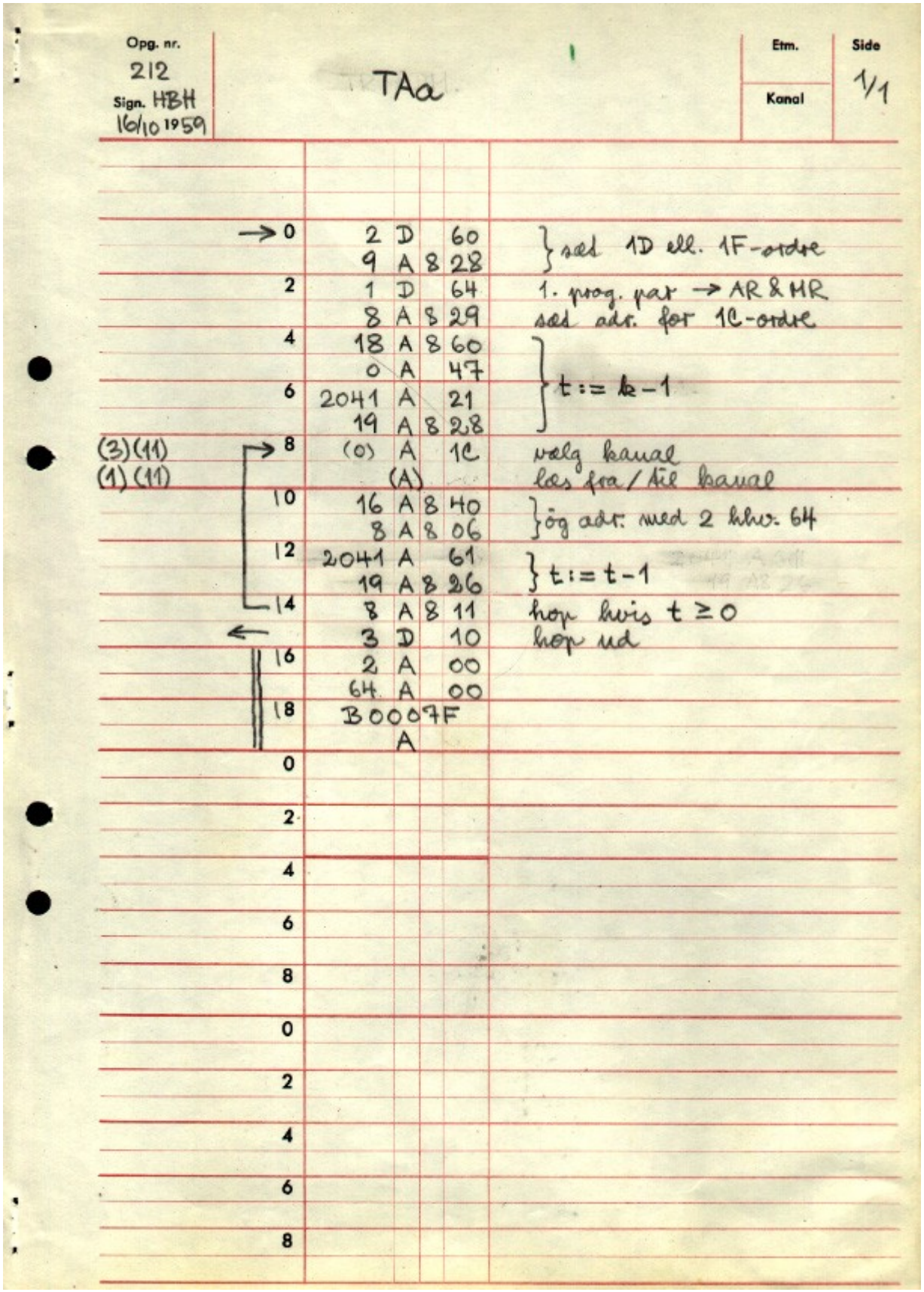
I 4A8 hentes et element fra den første vektor til AR; første gang fra hec 138, næste gang fra hec 136, o.s.v. og sidste gang fra hec 100. Dette element pakkes op til FMR i 5A8. Derefter hentes et element fra den anden vektor (6A8) og pakkes op til FAR (7A8). Nu er scenen klar til at benytte FR1's multiplikationssekvens (se specifikationen af FR1) og det sker i 8A8. Nu er det sådan at FMD, som til at begynde med er nul (jfr. 1A8), skal indeholde det skalære produkt af de to vektorer når løkken stopper. Derfor adderes FAR til FMD ved hjælp af FR1 i 9A8. Bemærk at indhoppet 0A9 i FR1 afleverer resultatet både i FAR og FMD, det får betydning om et øjeblik.

Den sidste del af programmet består blot i at pakke FAR sammen og gemme resultatet i hec 98, hvorefter programmet stopper. Man kunne kun pakke oppakkede tal sammen fra FAR til AR, ikke fra FMR og FMD, men som det ses er FR1 lavet så snedigt at man ikke kommer ud i problemer - i hvert fald ikke i dette eksempel.

Et lille program kodet af mig selv

Jeg vil slutte dette notat med et eksempel der viser tre ting: (1) hvordan en kodeblanket så ud, (2) hvordan man typisk overførte parametre til underprogrammer, og (3) hvordan man typisk programmerede i slutningen af 60'erne. Det drejer sig om et lille tromleadministrationsprogram, ved hvis hjælp man kunne overføre et vist antal konsekutive tromlekanaler til/fra ferritlageret. Sådanne småprogrammer var der mange der sad og lavede til sig selv, men for at andre ikke behøvede at opfinde hjulet hver gang udgav man sine hjælpeprogrammer i Dask-biblioteket, som efterhånden voksede sig ret stort. FR1 er et eksempel på et biblioteksprogram, men der var også sekvenser for kvadratrod, trigonometriske funktioner, Besselfunktioner og meget mere. Her kan man faktisk se hvordan standardfunktioner i højere sprog kan tænkes at være programmeret den dag i dag.

Mit lille program er ret simpelt, men lettere kryptisk. Det følger her:



Billedet viser et stykke **kodepapir**. Det var nogle blanketter i format A4, som var forberedt til at skrive ordrer på. Der var kodepapir i mange forskellige farver, og når man kodede et større program brugte man forskellige farver til de forskellige underprogrammer. Jeg er ked af at dette program er på hvidt papir, det havde været kønnere med fx blåt eller grønt, men jeg syntes jeg ville vise originalen.

Øverst på kodearket står nogle generelle oplysninger om programmet. Det fremgår fx at programmet er kodet som, eller i forbindelse med, opgave nr. 212 af HBH (mig, sa'e hunden) den 16. oktober 1959. Programmet hedder TAa. Alle programmer skulle have en kort betegnelse. Flydende regning hed som sagt FR1, kvadratrod hed AF1 o.s.v. Grunden til at det lille a ikke er et nummer er, at dette program ikke

er udgivet under de strenge regler der gjalt for egentlige biblioteksprogrammer, men derimod som en såkaldt "uofficiel sekvens". Det betød at brugen af den var på eget ansvar, og man kunne ikke fx rejse erstatningskrav imod Regnecentralen hvis der var kodefejl.

Selve programteksten finder man mellem de lodrette steger på blanketten; det var kun det der skulle hulles, og endda ikke alt: paranteserne i ordrerne 8A8 og 9A8 skulle fx ikke hulles med. Men det vidste hulledamen.

For at forstå programmet må man vide hvordan man kaldte det. Hvis det fx var lagret med begyndelsesadresse i 0AB, så skulle det kaldes med ordren:

```
0 AB 16
```

Hvis etikkettecelle B eksempelvis indeholdt konstanten 328, så svarer denne ordre til et sekvenshop til hac 328. Lad os endvidere antage at selve ordren 0 AB 16 står hac 105, som er et sted i det kaldende program; udførelsen af ordren bevirker da, at adressen 105 overføres til indeksregister D, så inde i TAa kan man bruge IRD til at referere tilbage til det kaldende program. Denne egenskab ved 16-hoppet udnyttes i TAa til parameteroverførsel. Der skal overføres følgende parametre:

- Startadressen for de tromlekanaler, der ønskes læst/skrevet.
- Antallet af tromlekanaler der ønskes læst/skrevet.
- Startadressen i ferritlageret, hvor tromlekanalernes indhold ønskes lagret eller befinder sig.
- Om der er tale om en læsning eller en skrivning.

Disse fire parametre overføres ved hjælp af to hac, hvis indhold er på ordreform, og som skulle placeres umiddelbart efter 16-hoppet. Formen skulle være som følger:

```
Startkanal Indeks antal
Lageradresse Indeks læs/skriv
```

Hvis man fx ønskede at læse 10 kanaler startende med kanal 100 til ferritlageradresse 1000, så ser hele kaldet således ud:

```
0 AB 16
100 A 0A
1000 A 1D
```

Bemærk at antal kanaler er angivet sedecimalt (fordi parametrene skulle være på ordreform), og at den anden parameter simpelthen er selve læse/skrive-ordren. For at forøge fleksibiliteten af programmet er det lavet sådan at anden parameter gerne må være indeksmærket med B eller C (men ikke med D, for IRD bruges jo til at gemme hopadressen i). Det kunne fx bruges på den måde at et af indeksregistrene kunne indeholde en basisadresse for bufferen i ferritlageret. Denne form for basisadressering kan også anvendes for kanaladressernes vedkommende, men her kan man kun bruge indeksregister C (forklaring følger).

Nu er vejen banet for at gå ind i selve koden og se hvad der sker. De to første ordrer henter den anden parameter og lagrer den i hac 9A8, altså i selve programmet! Vi har med andre ord at gøre med et program der ændrer sig selv. En sådan programmeringsteknik er nu helt forladt, men på den tid vi taler om her var det meget almindeligt at kode på denne måde.

Nu kommer vi til ordrerne i 2A8 og 3A8. Det der sker her er, at første programparameter hentes til både AR og MR og at adressepositionerne i AR gemmes i 8A8. Man ser at 8A8 på denne måde modificeres til at være en 1C-ordre (vælg kanal) med den rigtige kanaladresse. Her kommer muligheden for basisadressering af kanaladressen ind i billedet. Det var nemlig sådan, at 29-ordren gemte AR position 0 til 11, altså 12 positioner, mens den egentlige adresse i en ordre jo går fra position 1 til 11. Position 0 og position 12 i en ordre angav indeksmærkningen, og indeks C og D havde en etter i position 0; heraf

kommer det at den første programparameter godt måtte være C-mærket.

Alt dette går jo kun godt hvis man ikke anvender indeksregistrene i selve programmet. IRD anvendes til parameteradministration, så det kan ikke bruges som basisadresse, men IRB og IRC kan så bruges som beskrevet, forudsat at programmet ikke rører ved deres indhold. Derfor er den følgende løkke programmeret uden brug af indeksregistre.

I MR-registeret står i øjeblikket en kopi af første programparameter. Operationsdelen af denne parameter er det antal kanaler der skal overføres; det gælder derfor nu om at få isoleret denne operationsdel. Det sker i ordrerne 4A8 og 5A8. I 4A8 hentes en maske, med ettere i operationsdelen, fra 18A8 til AR, og 5A8 er en logisk konjuktion af AR og MR (resultat i AR). Den maske der hentes fra 18A8 er kodet som et tal på sedecimal form (5 sedecimale cifre); jeg kunne lige så godt have kodet: 0 A 7F, altså på ordreform, men jeg har åbenbart ment at man bedre kunne "se" bittene i masken, hvis den var sedecimal.

Efter 5A8 står det ønskede antal kanaler i AR som et heltal; det er antallet af gennemløb af den følgende løkke. Nu er det lettest at teste på 0 i Dask, så jeg trækker 1 fra AR inden løkken (ordren i 6A8) og gemmer resultatet som en tællekonstant i en *arbejdscelle*, nemlig hac 19A8 (ordren i 7A8). Hac 2041 er en af de permanente konstanter i normallejet; den indeholder heltallet 1.

Herefter ryger man ind i løkken, der går fra 8A8 til og med 14A8. De to første ordrer er dem der blev modificeret i begyndelsen af programmet, og som vælger en tromlekanal og læser eller skriver til/fra ferritlageret. Problemet er nu at tromlekanaladressen skal forøges med 2 (tromlekanalerne har lige adresser) og ferritlageradressen skal forøges med 64 (en kanal er 64 hec stor) inden næste gennemløb af løkken. Det klares ved at der i hec 16A8 er lagret en konstant med de rigtige adressedele. Her skal man passe på; det må fremgå tydeligt af programspecifikationen, at TAa skal lagres med lige begyndelsesadresse. Ellers kommer de to dele af denne konstant ikke til at stå i samme helcelle, og kan følgelig ikke hentes med en enkelt 40-ordre som vist i 10A8. Den slags kodetricks var meget almindelige når man kodede til Dask, og skete naturligvis for at spare ordrer. På grund af det lille ferritlager var det altafgørende at gøre sine programmer - og da navnlig biblioteksprogrammerne - så korte som muligt. 06-ordren i 11A8 er en "adder til helcelle" og bevirker at de to adresser i hec 8A8 bliver forøget med 2 hhv. 64; også her ser vi at disse to modificerede ordrer skal begynde med en lige adresse, hvilket de med lidt snilde også er kommet til forudsat at hele programmet begynder i en lige adresse.

Sidstre del af løkken består i at trække 1 fra tællekonstanten i 19A8 og hoppe forfra hvis den er større end eller lig med nul. Her er brugt en halvcellevariant (26) af grundoperationen 06, som adderer og lagrer i een operation; resultatet står altså både i AR og i 19A8.

Tilbagehoppet til det kaldende program sker med ordren 3 D 10, og det er naturligvis for at hoppe forbi de to programparametre.

Som sagt er det kun det der står mellem de lodrette linier på kodearket, der skal hulles, men der er skrevet en del andre ting på arket; de er der for en menneskelig læsers skyld. Der er nogle pile ind i og ud af visse af ordrerne; indadgående pile markerer ordrer hvortil der hoppes fra andre steder i programmet eller fra det kaldende program, og udadgående pile markerer selve hopordrerne. I dette program er der en forholdvis kort løkke, så her har jeg lavet en ubrudt pil fra bunden af løkken til toppen. Hvis løkken havde været større ville der nok have været både en indgående og en udgående pil, og den indgående pil ville have været mærket med den relative adresse på den eller de ordre(r), der udførte hoppet. Tallen i parentes til venstre på kodearket angiver fra hvilke andre ordrer ordren modificeres; fx modificeres ordren i 8A8 fra ordrerne i 3A8 og 11A8. Den lodrette dobbeltstreg til venstre for de sidste ordrer angiver at der slet ikke er tale om egentlige ordrer, men om konstanter o.lign. som bruges i programmet, men ikke udføres.

Sådanne oplysninger var helt essentielle i indkøringsfasen, navnlig hvis ens program var stort.

Det var alt hvad jeg vil sige om programmering til Dask. Jeg håber det er nogenlunde forståeligt for den

interesserede læser, og at det giver et indtryk af hvor primitivt arbejdet med computere var i Danmark i halvtredserne. I 1957 kom den første Fortranoversætter, og i 1960 fik vi Algol - og så blev programmørens arbejde revolutioneret. Men det er en anden historie.