NR. 7

MAI 1973

ND NYTT

Professor Heier viser frem månestøv fra Apollo-ferden.

A/S NORSK DATA-ELEKTRONIKK
Økernveien 145 - Oslo 5 - Tlf.: 21 73 71

AV INNHOLDET

ND NYTT

MAI 1973 - Nr. 7

Redaksjon:
Økernveien 145, Oslo 5. Tlf. 21 73 71

Redaktør:
Jan Aske Børresen

Redaksjonssekretær:
Inger Meidell-Haugland

Tidsskriftet distribueres gratis til alle interesserte. Vennligst send et brevkort til redaksjonen og kom med på distribusjonslisten.

Regneanlegget ved Mineralogisk-Geologisk Museum

UNIVERSITETET I OSLO

A. O. BRUNFELT - K. S. HEIER - E. RYEN - B. SUNDVOLL

All moderne naturvidenskapelig forskning er i dag mer eller mindre avhengig av elektronisk databehandling og de muligheter den åpner. I tillegg er den elektroniske regnemaskin også et utmerket verktøy for styring og overvåking av videnskapelige eksperimenter. Mineralogisk-geologisk Museum i Oslo har en lang tradisjon når det gjelder instrumentelle kjemiske analyser, et område hvor maskinell styring har vist seg å være meget lovende. For tiden er røntgenfluorescense-spektrografi og neutronaktiveringsanalyse basert på γ-spektrometri de to mest aktuelle analysemetoder ved museet. Det er videre planer om anskaffelse av et massespektrometer for geologiske aldersbestemmelser.

Da det i 1969 skulle kjøpes inn utstyr for γ-spektrometriske målinger, falt valget på en NORD-1 regnemaskin (4K hukommelse) og innkjøp av en analog-digitalomvandler (ADC), idet konvensjonell apparatur som digital-pulshøydeanalysatoren ikke ville falle vesentlig billigere i innkjøp. Et annet og viktig moment som spilte inn ved valget, var også mulighetene for direkte kommunikasjon mellom eksperimentator og hans data ved hjelp av en programmerbar dataskjerm og en on-line teletype samt raskere og mer fullstendig etterbehandling og reduksjon av data. Samtidig åpnet terminalprosjektet ved EDB-senteret på Blindern mulighet for en direkte tilknytning til et større regneanlegg som kan ta seg av større numeriske regneoppgaver og filebehandling av data. Anlegget ble installert og tatt i bruk høsten 1969. Gjennom samarbeid med Institutt for Atomenergi på Kjeller (IFA), som forøvrig gikk til innkjøp av lignende utstyr, ble det utviklet nødvendig interface og software for at anlegget skulle kunne tjene som et on-line γ-spektrometer for de aktiveringsanalytiske målinger.

Prinsippet for neutronaktiveringsanalysen går i korthet ut på å bestråle en kjent mengde av et stoff med ukjente bestanddeler med en kjent mengde av et stoff hvor bestanddelene er kjent (referanseprøve) i en atom-reaktor og deretter med eller uten kjemisk forbehandling, måle induserte γ-aktiviteter av dannede radionuklider fra kjemiske elementer på et spektrometer. Da de forskjellige radionuklider viser karakteristiske γ-spektra, kan disse lett identifiseres. Videre er den utsendte strålingsintensitet direkte proporsjonal med mengden av radionukliden slik at kvantitative data for absolutte elementmengder og – konsentrasjoner kan beregnes ut fra målingene. I Fig.

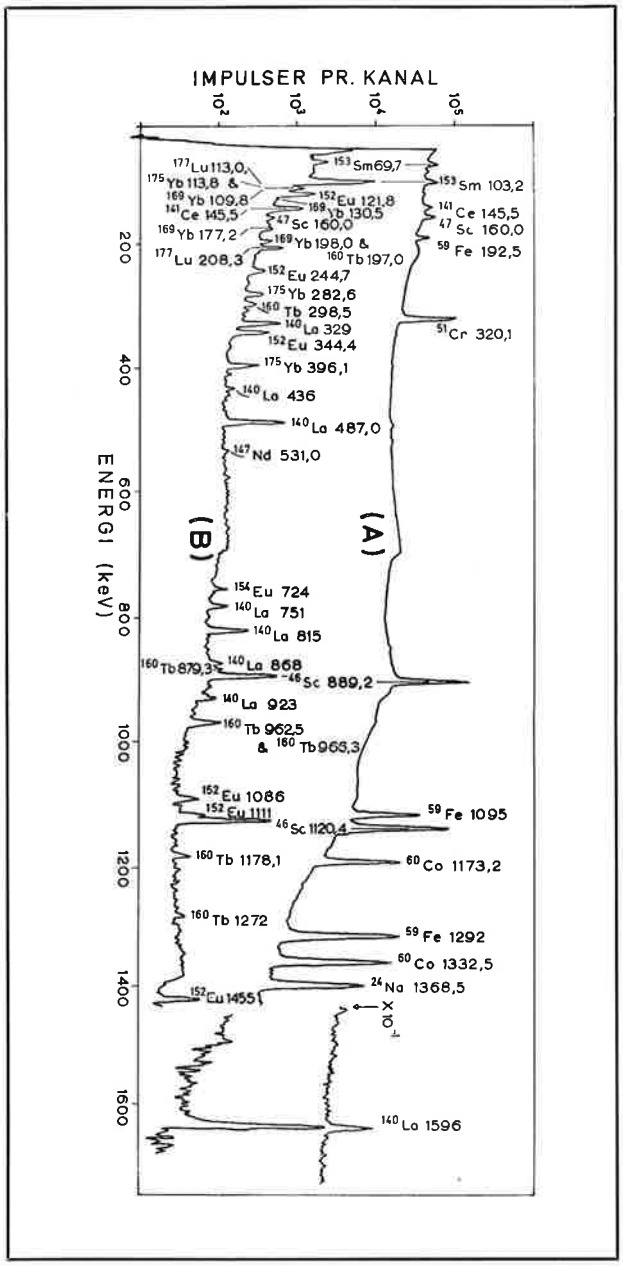
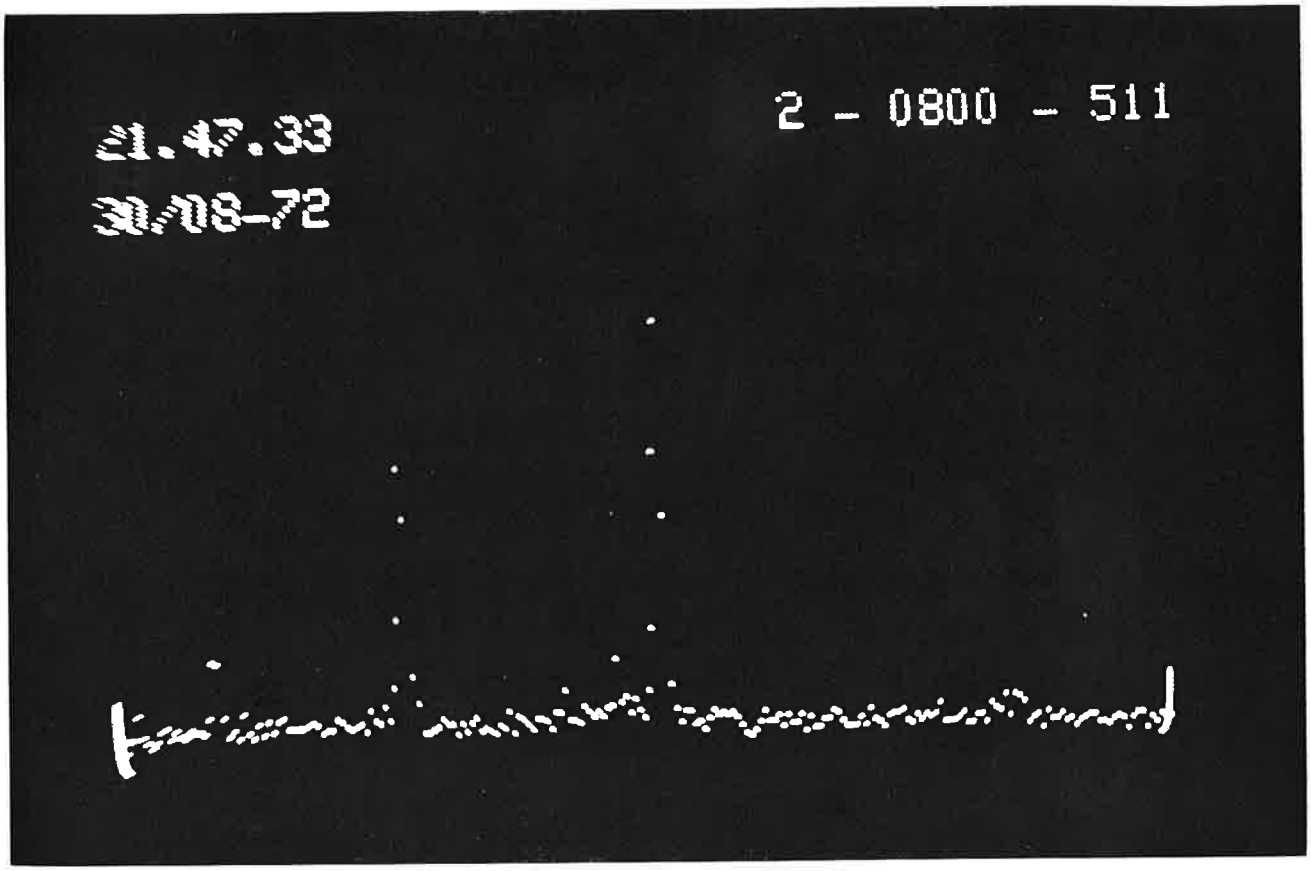


Fig. 1. Gammaspektra av en bestrålt bergart med basaltisk sammensetning, registrert 8 dager etter bestråling slutt (antall KeV pr. kanal ~ 2). A. Direkte av bestrålt prøve. B. Separat fraksjon av sjeldne jordarter.

Fig. 2. Et utsnitt av et gammaspekter fotografert fra den xy-programmerbare dataskjermen. Foruten angivelse av klokkeslett og dato, vises også hvilket av de to detektorsystemene som er i bruk, nummeret til den kanal som sees lengst til venstre og hvor mange tellinger som tilsvarer full skalering i y-retningen.



1 er det vist eksempler på den type γ-spektra som en får ved slike undersøkelser.

De måletekniske prinsipper i γ-spektrometrien er beskrevet tidligere i dette blad, og vil derfor bare kort bli gjentatt her: γ-kvantet eller fotonet registreres i en detektor av halvledermateriale, en såkalt Ge(Li) detektor hvor energien omformes til en strømpuls. Denne strømpulsen forsterkes opp, og overføres i en ADC til en adresse i regnemaskinens hukommelse. Via en hardwareinterface kalt DMI (direct memory increment), blir den spesifiserte hukommelsescelle økt med en hver gang. Normalt nyttes et område på 1–4K hukommelsesceller for lagring av slike data som dermed representerer et histogram med pulsenergier (γ-energier) som x-akse (celleadresse) og intensitet som y-akse (celleinnhold). DMI-interfacen er koplet til en datakanal, slik at akkumuleringen av data i hukommelsen fra ADC-en går uavhengig av regnemaskinens øvrige funksjon. Spekteret kan vises på en xy-programmerbar dataskjerm (Fig. 2), og display og akkumuleringen kan dirigeres fra en online teletype ved hjelp av et interrupt styrt program.

Erfaringene som ble høstet med dette anlegget var såpass oppmuntrende at høsten 1970 ble hukommelsen utvidet til 8K samtidig som det ble innkjøpt en del nytt periferutstyr som hullbåndleser og kortleser samt et modem og en modem-interface for tilknytning til Universitetets regneanlegg på Blindern. DMI-interfacen ble utvidet med en digital-routing enhet for simultan bruk av to detektor systemer, og software ble tilsvarende forandret. Dette nye system som er avbildet i Fig. 3 har i vesentlig grad bidratt til å

Fig. 3. Apparaturoppsetning for gammaspektrometri basert på en digital regnemaskin ved Mineralogisk-geologisk Museum.

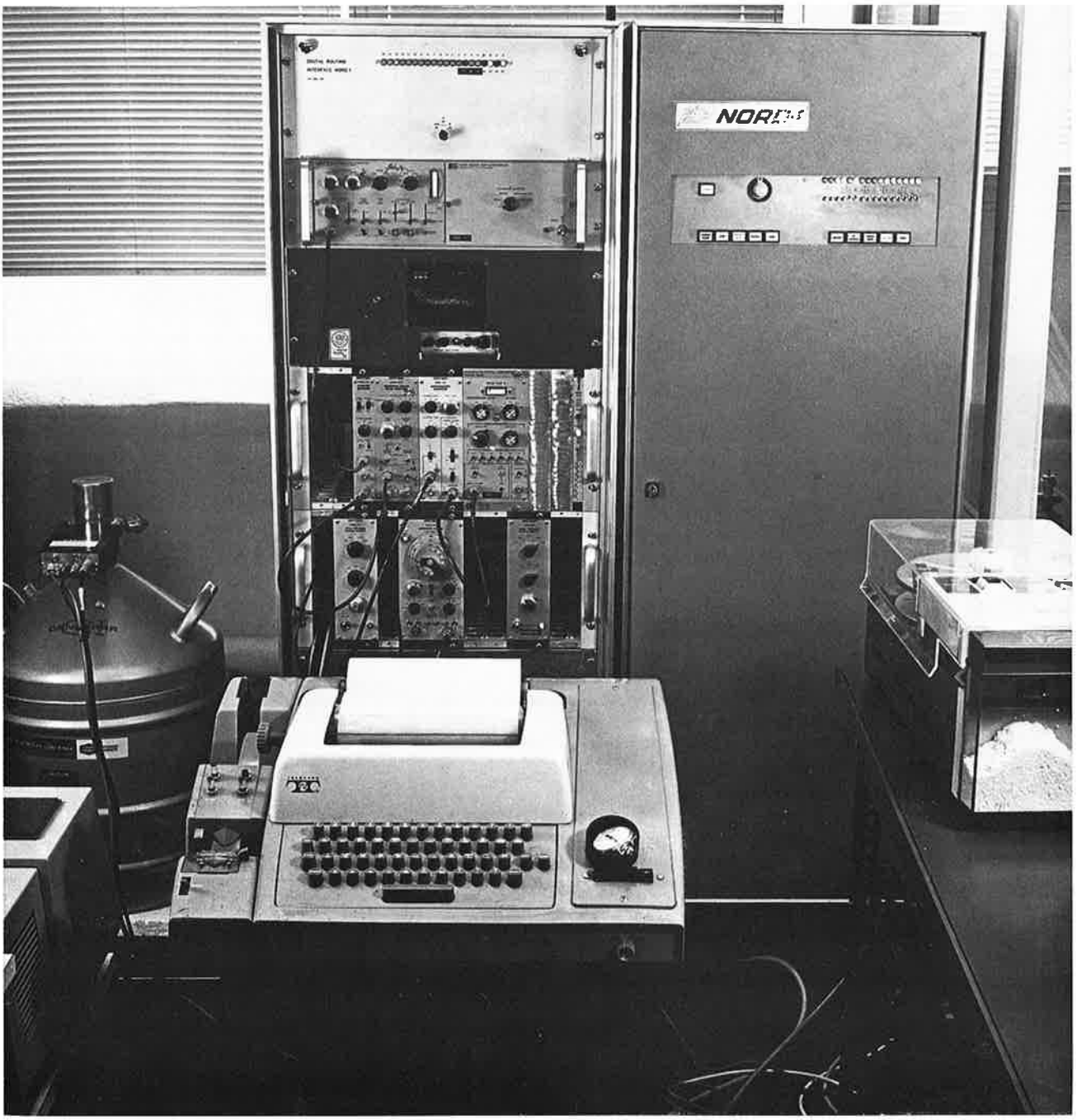
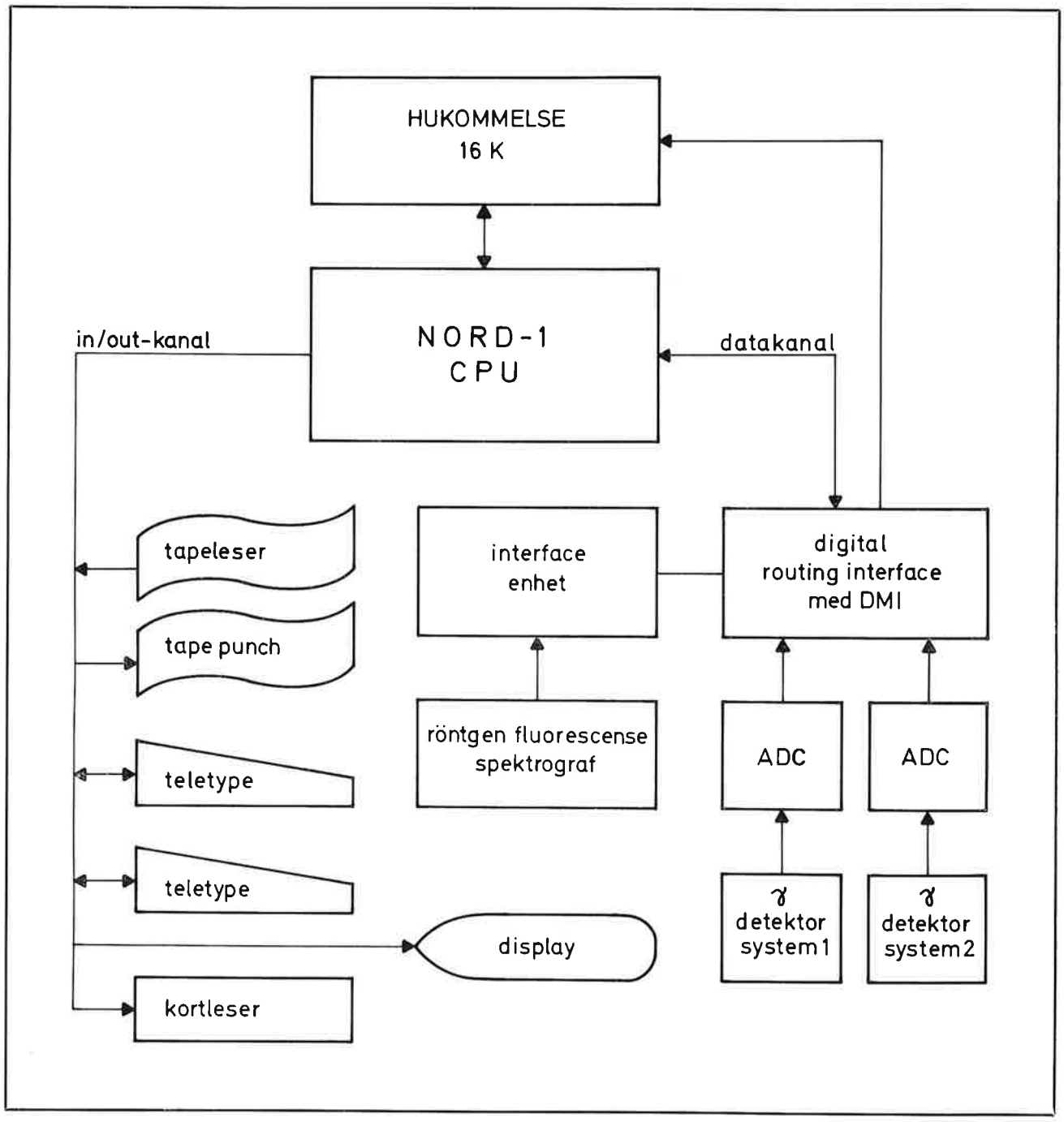


Fig. 4. Hardwarekonfigurasjon for regneanlegget ved Mineralogisk-geologisk Museum.



øke museets kapasitet når det gjelder sporelementanalyser. Av de prosjekter som i særlig grad har hatt glede av anlegget, kan nevnes Apollo-prosjektet (støttet av NTNF) hvor materiale fra Apollo 12, Apollo 14, Apollo 15 og Apollo 16 måneferdene er blitt studert; NAVF prosjektet EN NORSK GEOTRAVERS og NTNF prosjektet SJELDNE JORDARTERS GEOKJEMI.

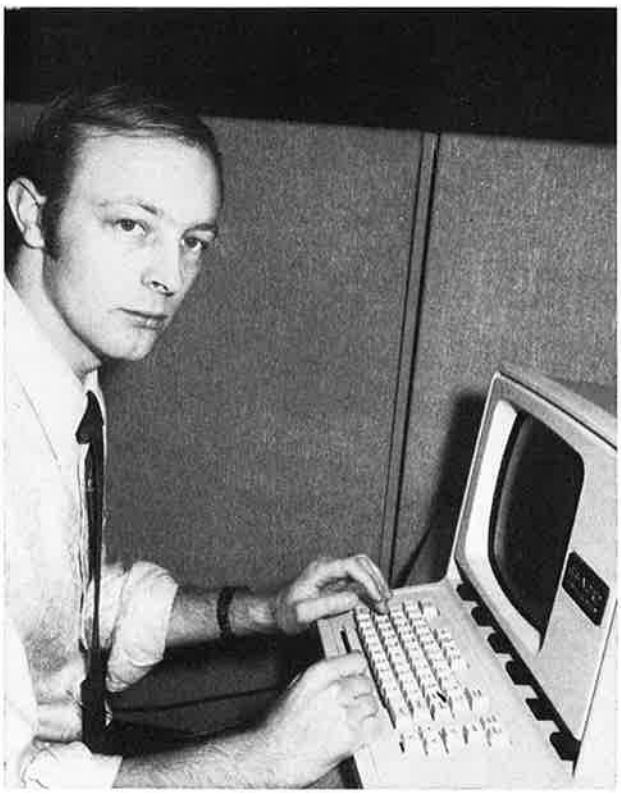
Våren 1972 fant en med museet tiden inne til å ta det endelige skritt mot et fullverdig regneanlegg. Hukommelsen ble utvidet til 16K og foruten innkjøp av ytterligere periferutstyr, ble røntgen fluorescense-spektrografen koplet til routing-interfacen. Den nåværende hardware-konfigurasjon er vist i Fig. 4. Samtidig ble en av oss (E. R.) engasjert for å lage den nødvendige software for det nye systemet. Som basis for dette ble brukt IFA's versjon av All-core-monitor. Programmeringsarbeidet er i alt vesentlig avsluttet og systemet som har fått navnet GEONORD er for tiden til uttesting.

Da Universitetets interne terminalanlegg ennå ikke er ferdig utviklet, er forbindelsen med regneanlegget på Blindern ennå ikke prøvd, selv om all nødvendig hardware er installert. Operasjonsystemet mangler også muligheter for kjøring av separate programmer (f.eks. små FORTRAN-jobber), samtidig med on-line-eksperimenter, noe som betinger enten ytterligere hukommelse eller et masselager. En tar imidlertid sikte på en fremtidig anskaffelse av et masselager og en linjeskriver, for å kunne utnytte effektivt den ennå forholdsvis rikelige «død-tid» som systemet nå opererer med. Dessuten er det meningen å kople on-line det planlagte massespektrometer.

Professor Heier viser frem månestøv fra Apollo-ferden.

Hashing – metoder og bruk

AV SIV.ING. OVE LANGE, A/S NORSK DATA-ELEKTRONIKK



Hashing er en adresseringsmetode hvor adresser gjenfinnes på grunnlag av dataenes innhold. Det er en softwaremetode for assosiativ lagring, og noe upresist kan man si at i stedet for å finne innholdet av en fysisk adresse, bestemmer man innholdets fysiske adresse. I denne artikkel vil bli gjennomgått noen av de programmeringsmetoder som benyttes for å transformere den logiske adresse (gitt av dataenes innhold) til den fysiske adresse i hurtiglageret eller et annet lager med direkte aksess. Hashing brukes både for innsetting og henting av data og i de fleste tilfeller kan man nå dataposten i et skritt, dette i motsetning til data f.eks. ordnet alfabetisk, der man ved binær søkning gjennomsnittlig vil trenge $\log_2 p$ tabelloppslag for å finne en adresse blant p totalt. I hashing benyttes ofte «scatter storage» som vil si at dataene lagres innenfor et definert fysisk adresseområde og plassen bestemmes av hashingmetoden.

Hashing benyttes ofte i systemprogrammering, f.eks. i kompilatorer, assemblere, og filesystemer hvor man trenger raske oppslag av data eller hopp til subrutinene, men prinsippet bør få anvendelser også ved applikasjonsprogrammering.

METODER

Ideen med hashing er å foreta en transformasjon på postens identifikator. Identifikatoren er den delinformasjon som karakteriserer posten, dette er gjerne et alfanumerisk «navn». Transformasjonen gir et «tilfeldig» heltall som representerer den fysiske adresse. Så lenge ingen identifikatorer genererer samme hash-kode, vil både innsetting og søkning foregå i ett adresseberegningsskritt, uavhengig av det fysiske adresseområdets størrelse. To identifikatorer kan imidlertid gi samme hash-kode, det er derfor viktig å anvende en hash-transformasjon som genererer hash-koder i en mest mulig sann tilfeldig distribusjon over det fysiske adresseområde. Hvis tilfeldig distribusjon kan oppnås, vil søking og innsetting gjennomsnittlig gå på mindre enn 2 adresseberegningsskritt for et fysisk adresseområde som er 75 % fylt opp med poster.

De vanligste hash-kodingsmetodene baserer seg på at man på grunnlag av et k-bits felt av identifikatoren beregner et heltall mellom 0 og 2^k-1. Dette medfører at størrelsen av det fysiske adresseområdet må begrenses til verdiene 2^k. Disse metodene kan grovt deles opp i l o g i s k e og m u l t i p l i k a t i v e. I tillegg skal vi her se på en foreslått metode som er blitt kalt d i v i s j o n s metoden (Maurer).

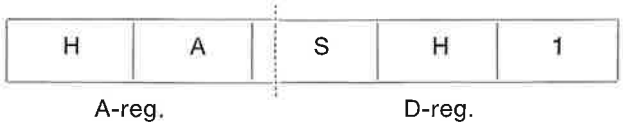
L o g i s k e m e t o d e r

Logiske metoder utfører gjerne en EXCLUSIVE OR-funksjon på deler av identifikatoren, f.eks. de fire første karakterene.

Logiske metoder er stort sett raskere enn de multiplikative. De kan imidlertid føre til dårlig distribusjon av hash-koden. Alle identifikatorer som starter med de samme symbolene (eks. HASH1, HASH2, HASH3) vil generere samme hash-kode for eks. metoden med de fire første karakterer.

Det er med hell brukt EXCLUSIVE OR på de enkelte (hele) ord for MAC-symboler. MAC-symboler består av opptil 5 karakterer a 6 bits, og følgende eksempel

viser hvordan man genererer et 16-bits heltall av et MAC-symbol med EXCLUSIVE OR:




```
LDD SYMBOL          % Hent symbolet
COPY SA DT
REXO SD DT
SKP IF DT GRE Ø     % Tving positiv
COPY ST DT CM2      % T inneh. pos. heltall
```

Symbolet HASH vil resultere i tallet 5352 (12350_8).

Dersom det fysiske adresseområdet er mindre enn de tall som genereres, føyes til en beregning på modulo adresseområdet. En nogenlunde tilsvarende metode er å addere bit-feltene og benytte de k nederste bits som hash-kode.

Multiplikative metoder

Her finnes flere metoder. Man kan multiplisere k-bits felt av identifikatoren med hverandre, multiplisere identifikatoren med seg selv eller med en konstant osv. Man bruker så et k-bits felt som hash-kode.
En ofte anvendt metode som gir god distribusjon av hash-kodene, er å kvadrere identifikatoren og ta ut et bitfelt midt i denne som hash-kode. Siden verdien av kvadratets midlere bits er avhengig av alle bits i identifikatoren, kan man anta en relativt stor sannsynlighet for at ulike identifikatorer gir ulike hash-koder, selv om identifikatorene er delvis like. Metoden er mest anvendelig for enkelt-ord identifikatorer, men kan også tilpasses multiord identifikatorer. Hvis vi bruker eksemplet med MAC-symboler, vil følgende program utføre en kvadrering av symbolets midtre åtte bits og anvende 9 bits av kvadratets øverste 10 bits for å adressere en symboltabell på 512 symboler.

```
LDD SYMBOL          % Logisk adresse
SAD 5
LDX (400-1
RAND SX DA          % A inneh. 8 midtre bits
STA TEMP
MPY TEMP            % Kvadratet
SHA SHR 6
LDX (1000-1
RAND SA DX          % Modulo Tab-størrelse
STX TEMP
RADD SX DX
SWAP SX DA
ADD TEMP            % 3 ord pr. record
SWAP SA DX
XXX I ,X (SYMTABELL
```

Divisjonsmetoden

Den foreslåtte metode er tidsmessig sammenlignbar med multiplikasjonsmetodene, men den har fordelen at den anvendte divisor som regel er et lite tall i forhold til multiplikator anvendt ved multiplikasjonsmetodene. Metoden er meget bedre enn både de logiske og de multiplikative når det gjelder å gi en tilfeldig distribusjon av hash-kodene.
Metoden anvender identifikatorer som er heltallsord (hvis identifikator består av flere ord, foretas først en EXCLUSIVE OR på alle ordene) som divideres med adresseområdets størrelse, hvoretter resten av divisjonen anvendes som hash-kode. Denne metoden legger bare den restriksjonen på tabellstørrelsen at den må være odde, ellers vil det laveste bit i den genererte hash-kode alltid bli lik laveste bit i identifikatoren.
Når heltallsordet er generert av identifikatoren, vil følgende algoritme utføre beregningen av hash-adressen:

$$\text{HASHADR} = \text{BASE} + (\text{HELTALL} - \left\lfloor \frac{\text{HELTALL}}{\text{MODULUS}} \right\rfloor \times \text{MODULUS})$$

hvor følgende notasjoner er brukt:

HASHADR	– generert fysisk adresse
BASE	– adresseområdets startadresse
HELTALL	– identifikator-heltallet
MODULUS	– adresseområdets størrelse
$\lfloor \quad \rfloor$	– heltallsdivisjon

Metoden vil ha en uheldig konsekvens idet at identifikatorer som adskiller seg med en i laveste bit (HASH1, HASH2, osv.) vil få hash-koder som også adskiller seg med en. Denne form for «klatting» innenfor adresseområdet kan være uheldig, men behøver ikke være kritisk hvis man velger en velegnet algoritme for å behandle kollisjoner.

KOLLISJONER

Kollisjoner oppstår når hash-algoritmen genererer identisk adresse for to nøkler. Den enkleste måten å løse problemet på er å søke suksessivt på celler som følger hash-adresen k, til man finner en ledig plass, hvor den nye posten plasseres. Metoden fører til sen søking når adresseområdet fylles opp. Adresseområdet (tabellen) blir gjerne betraktet som sirkulært, slik at telling gjennom tabellen foregår i modulo tabellstørrelsen. I stedet for å telle suksessive celler (inkrement 1), kan vi telle med større inkrementer enn 1. Dette forbedrer erfaringsmessig ikke metodens hurtighet svært meget, men klumping av poster reduseres.

Kvadratisk søking kan brukes ved at inkrementet i = 1, 2, 3 behandles som en kvadratisk ligning, k + ai + bi² (modulo tabellstørrelsen), hvor k, a og b er konstanter. En fordel med denne søkingen er at hvis den for eksempel går over x_1, x_2, ..., x_{10} når søkingen starter i x_1, så vil denne ikke dekke over elementene x_i, x_{i+1}, ..., X_{10} når denne starter fra x_i (i > 1).

Maurer foreslår en kvadratisk metode etter algoritmen

$$D_i = -(i^2 - N)/2$$

hvor D_i er i'te oppslag (i = 1, 2 ....) og N er adresseområdets størrelse. Dette gir et meget enkelt program:

```
        LDX H          % Initiell hash-adr.
        LDT (-NØ5      % = -(N + 1)/2
        STT INCR
        LDT (N         % Tabell-størrelse, primtall!
        JMP TEST
LOOP,   LDA INCR
        RSUB SA DX
        SKP IF DX LST ST
        RSUB ST DX     % Modulo tabellstr.

TEST,   [   ?   ]      % T og X gjemmes

        MIN INCR
        JMP LOOP
        JMP TABELLFULL % hvis mer enn 1/2 tab.
                       % scannes, antas
                       % tabell full
```

Muligheten for kollisjoner øker når tabellen blir fylt opp. En metode som er relativt enkel i programkompleksitet er å innføre pekere fra tabellen når en kollisjon opptrer, og samle alle kolliderende elementer i lenker i en separat tabell (Overflowtable). Dette krever en større tabell for kolliderende elementer som ikke kan settes i hash-tabellen, men hash-tabellen selv kan til gjengjeld gjøres mindre.

En effektiv og elegant metode for å løse kollisjoner er å bruke en pseudo-tilfeldig tallgenerator til å produsere det ønskede inkrement til den initielle hash-adresse. Tallgeneratoren D gir et tall D_i for oppslag i (i = 0, 1, 2 ....) mellom null og N-1 (evt. 0 til 2k-1), og neste søking gjøres i H + D_i (H initiell hash-adresse).

I og med at muligheten for kollisjon eksisterer i enhver hash-adresse, må algoritmen sjekke om den post man har funnet er den ønskede. Derfor må postens identifikator lagres sammen med posten. Det er dog ikke nødvendig å sammenligne identifikatorene for poster hvor kollisjon ikke er inntruffet. Man kan benytte en testbit i posten, som settes når flere enn en identifikator har generert samme initielle hash-kode. Hvis dette bit er satt, må identifikatorene sjekkes mot hverandre, ellers ikke. Siden det er et mål med hash-algoritmene å unngå kollisjoner, vil dette kunne redusere test-tider for identifikasjon av postene.

KVALITETSESTIMERING

For å vurdere hvor godt en hashing-algoritme fungerer, vil vi for enkelthets skyld si at målet er å gjøre post-søkingen så rask som mulig. Vi skal her se bort fra tidsoptimalisering av søking på roterende masselager, og kan da grovt regne at tidsforbruket er proporsjonalt med antall oppslag for lokalisering av posten.

Definisjoner:

- N — hash-områdets størrelse (antall poster)
- P — antall poster innsatt
- α — P/N, fyllfaktor
- E — gjennomsnittlig antall oppslag for lokalisering av alle P poster.

En minimalisert E vil bli brukt som godhetskriterium. Hvis man idealiserer ved å anta tilfeldig og uniform distribusjon av hash-adresser over det tilgjengelige

adresserom, så blir forventet antall oppslag (S) for innsetting av den (P + 1)'te posten:

$$(1)\ S(\alpha) = 1 + \frac{P}{N} + \frac{P(P-1)}{N(N-1)} + \cdots + \frac{P(P-1)\cdots 1}{N(N-1)\cdots(N-P+1)} = \frac{1}{1-\frac{P}{N+1}} \approx \frac{1}{1-\alpha}$$

Vi er interessert i å bestemme gjennomsnittlig antall oppslag med hensyn til fyllfaktoren α, og finner dette ved å integrere S med hensyn til α:

$$(2)\ E = \frac{1}{\alpha}\int_0^{\alpha} S(\xi)\,d\xi = \frac{1}{\alpha}\int_0^{\alpha} \frac{d\xi}{1-\xi} = -\frac{1}{\alpha}\ln(1-\alpha)$$

Følgende tabell gir noen verdier av gjennomsnittlig antall oppslag E, med hensyn til fyllfaktoren α:

α	E
0.1	1.054
0.25	1.151
0.5	1.386
0.75	1.848
0.9	2.558

En vesentlig egenskap ved resultatet er at P eller N bare er med i form av forholdet mellom dem. Teoretisk sett vil lokaliseringen av en post i et tabellområde fylt med 8 av 10 mulige poster, og en annen med 800 av 1000 mulige poster ta like lang tid. Dette er en typisk egenskap for hashing-metodene.

SLUTTBETRAKTNINGER

Fremgangsmåten for å sette inn en ny eller søke opp en gammel post er helt lik. Det er derfor nok å bruke et flagg-bit for å markere hvilken funksjon som skal utføres, og samme hash-algoritme kan brukes.

Et spesielt problem som oppstår når man skal slette en post, er ikke behandlet i det foregående. Når posten er lokalisert ved hjelp av hash-algoritmen, kan den ikke uten videre slettes, da det alltid er mulighet for at andre poster har kollidert på samme adresse og ligger etter den post som skal slettes. Fordi kollisjoner behandles ved en eller annen inkrementerende metode, vil man i så fall miste tilgang til de poster som ligger etter, fordi den slettede post vil bli oppfattet som «første ledige» post.

En metode til å løse dette problem, er å utstyre alle poster med to flagg-bits, hvor det ene indikerer «flere poster følger/siste post», det andre «post ledig/post opptatt». Med en slik metode vil den lokaliserte post kunne slettes direkte. Ved søking etter en ledig (tom) post, vil første «post ledig» kunne brukes, dersom det er «tomrom» i en kollisjonslenke.

Dersom det er ugunstig å avsette plass til flagg-bits, kan man alternativt foreta ny hashing på alle poster som «ligger etter» den som skal slettes.

Av og til vil hash-tabellen være helt konstant (f.eks. mnemonics-tabellen i MAC). Hvis det er mulig vil det være en stor fordel å installere de mest brukte (repeterte) symbolene først når tabellen genereres. Dette vil kunne medføre en drastisk reduksjon i den praktiske E. (I MAC ville det sannsynligvis være riktig å installere LDA/STA først, og ION/IOF til slutt, f.eks.)

I denne korte oversikten har en i første rekke utelatt en viktig side, og det er eksempler på egnede tabellstrukturer. Man er imidlertid av den formening at dette har så sterk tilknytning til den aktuelle anvendelse at det i denne sammenheng ligger litt på siden.

Referanser:

1. Maurer: «An improved hash code for scatter storage», CACM Jan. 68.
2. Morris: «Scatter storage techniques», CACM Jan. 68.
3. Radke: «The use of quadratic residue research», CACM Feb. 70.
4. Bell: «The quadratic quotient method: A hash code eliminating secondary clustering», CACM Feb. 70.
5. Lamport: «Comment on Bell's quadratic quotient mehod for hash code searching», CACM Sep. 70.
6. Day: «Full table quadratic searching for scatter storage», CACM Aug. 70.
7. Bell and Kaman: «The linear quotient hash code», CACM Nov. 70.
8. Lum, Yuen and Dodd: «Key-to-address transform techniques: A fundamental study on large existing formatted files», CACM Apr. 70.
9. Price: «Table lookup techniques», Computing Surveys Jun. 70.
10. Harrison: «Implementation of the substring test by hashing», CACM Dec. 70.
11. Bloom: «Space/time trade-offs in hash coding with allowable errors», CACM July 70.

Software for NORD maskiner

AV ROLF JØRGEN OLSEN, SYSTEMPROGRAMMERER, A/S NORSK DATA-ELEKTRONIKK

I de siste årene har A/S Norsk Data-Elektronikk satset sterkt på softwareutvikling for NORD datamaskiner. Softwareavdelingen er således den avdeling som har ekspandert sterkest.

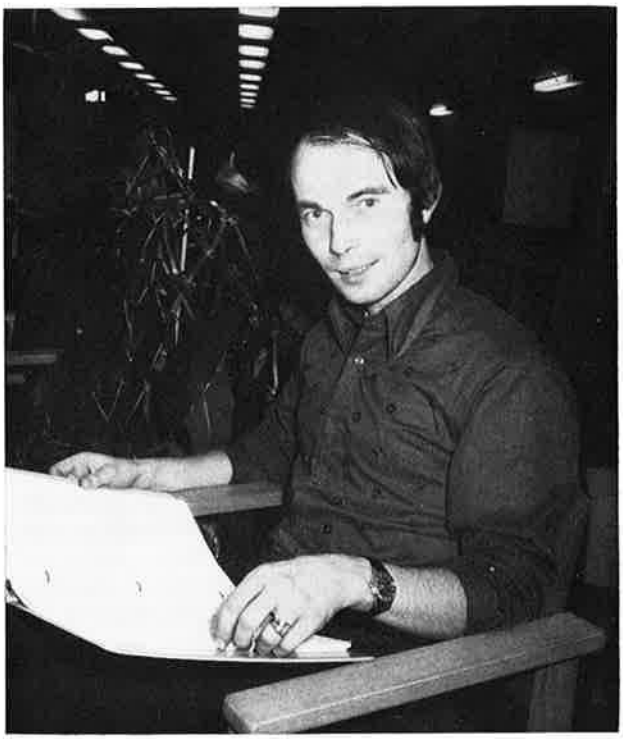
Software deles gjerne opp i to grupper: systemprogrammer og applikasjonsrettede programmer. For enkelt å kunne anvende en datamaskin som «oppgaveløser» må den utstyres med systemprogrammer. Det vil si, standard programmer som kan oversette mer eller mindre komplisert symbolsk kode til maskinens interne instruksjoner, eller kort sagt programmer som gjør det lettere for en bruker å utnytte alle funksjoner i maskinen uten å ha detaljkunnskap om dens indre virkemåte. Eksempler på systemprogrammer kan være: Assembler, FORTRAN kompilator, BASIC, sanntidsmonitorer, batch operasjonssystem, Time-Sharing system osv.

De tre førstnevnte kalles ofte for programmeringssprog. Assemblere er maskinorienterte (lavnivåsprog) i motsetning til for eksempel FORTRAN som er et problemorientert høynivåsprog, dvs. FORmula TRANslation.

Hos A/S Norsk Data-Elektronikk er MAC Assembly det sprog som er verktøyet for utviklingen av alle systemprogrammer.

En vanskelighet ved utvikling av systemprogrammer er kravene til kompatibilitet. Utviklingen på datamaskinområdet krever en stadig modernisering og forbedring av programmene, men man kan ikke uten videre erstatte gamle funksjoner med nye og bedre. Av hensyn til allerede eksisterende software må gamle funksjoner beholdes.

Utviklingen viser at det blir billigere og enklere å lage datamaskiner, og det er derfor viktig for en datamaskinprodusent å stå sterkt når det gjelder programutrustning. A/S Norsk Data-Elektronikk har tidlig tatt konsekvensen av dette, og en vesentlig del av innsatsen i softwareavdelingen har vært konsentrert rundt utviklingen av generell standard software. Imidlertid har avdelingen også opparbeidet seg adskillig know-how innenfor en rekke applikasjonsområder. ND påtar seg totale systemløsninger, og leverer således både hardware og applikasjonssoftware.



ND'S STANDARD SYSTEMER

MAC

MAC er et interaktivt assembly og debugging program for NORD datamaskiner. Programsystemet er modulært og har blant annet topass opsjon og makro opsjon. For øvrig kan man velge mellom å assemblere et program direkte inn i hukommelsen – slik at det kan kjøres umiddelbart, eller programmet kan assembleres ut på for eksempel file eller papirbånd i BRF format. BRF er forkortelse for Binær Relokerbart Format. Det samme formatet produseres av FORTRAN slik at det er mulig å forene programmer skrevet i disse ulike sprogene. Tidskritiske programenheter kan således skrives i assemblykode for å oppnå maksimal effektivitet mens resten av programmet kan skrives i høynivåsprog.
To spesielle versjoner av MAC har blitt laget i forbindelse med utviklingen av SINTRAN real-time monitor: MACM, MAC Mass Storage Assembler kan as-

semblere programmer til et core-bilde på masselager (trommer eller disk). MACD, MAC Debugging Assembler brukes sammen med SINTRAN monitor i masselagersystemer for on-line debuggings-formål.
Konseptet som MAC bygger på ble i sin tid utviklet ved MIT (Massachusett's Institute of Technology, U.S.A.). Ved Forsvarets Forskningsinstitutt, Kjeller ble det på grunnlag av dette utviklet to assemblere, ASEM og SMIL for henholdsvis SAM 1 og SAM 2 maskinene. Med erfaring fra disse assemblere utviklet ND MAC for NORD-1 i 1968. Hovedansvarlig var siv.ing. Rolf Skår. Senere har MAC gått gjennom flere utviklingsfaser, blant annet for å kunne produsere samme format som FORTRAN (BRF).

FORTRAN

FORTRAN IV

FORTRAN IV består av en enpass FORTRAN IV kompilator, relokerende loader, FORTRAN IV Library og FORTRAN IV inn/ut system. FORTRAN library er re-entrant og inneholder en rekke programmer, så som SIN, COS, SQRT, ATAN, SINH, EXP, etc. Det er også en sanntidsversjon av biblioteket.

FORTRAN IV er definert i ANSI Standard FORTRAN X3.0-1966, og ND's versjon er tilpasset denne definisjonen. En versjon av FORTRAN IV, kalt RT-FORTRAN, produserer objektkode som kan kjøres under SINTRAN monitor. Med dette har ND muliggjort skriving av sanntidsprogrammer (for eksempel for prosesskontroll) i høynivåsprog. Som nevnt i kapitlet om MAC har brukerne full mulighet til å skrive deler av programsystemet i assembly sprog.

ND leverer også FORTRAN i to andre versjoner: FORTRAN II og MINI FORTRAN, hvor det førstnevnte også kan produsere objektkode som kan kjøres under SINTRAN monitor.
Utviklingen av FORTRAN hos A/S Norsk Data-Elektronikk kan deles i 3 trinn: MINI FORTRAN i 1967–1970, FORTRAN II i 1969–1970 og FORTRAN IV i 1970–1971–1972. Som man kan se har utviklingen så å si pågått kontinuerlig siden firmaets start. Hovedansvarlige har vært henholdsvis cand. real. T. Amble, cand. real. C. P. Handberg og cand. mag. A. Lindheim.

BASIC

BASIC (Beginner's All-purpose Symbol Instruction Code) er et relativt enkelt programmeringssprog for å løse matematiske problemer. Sproget er meget brukt innen undervisning og forskning.
BASIC er utviklet ved Dartmouth College, U.S.A. BASIC følger de spesifikasjoner som utgis derfra.
BASIC er et høynivåsprog, men skiller seg vesentlig fra FORTRAN ved at det virker interpretativt og interaktivt, hvilket vil si at man umiddelbart får beskjed om syntaksfeil og således kan rette disse. BASIC må ligge inne i maskinen når brukerens program kjøres.
A/S Norsk Data-Elektronikk har utviklet en del spesielle funksjoner til sin BASIC som kort kommenteres her:

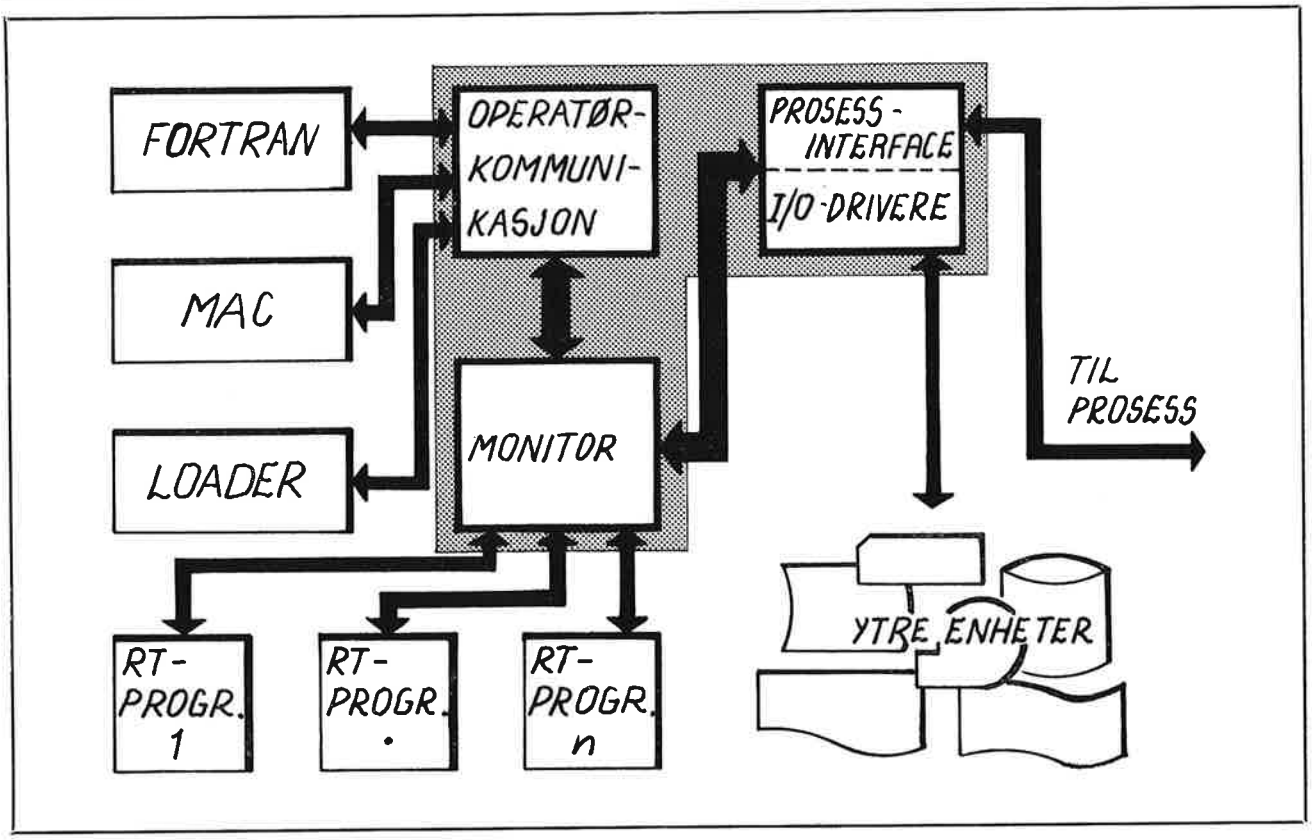
1. En monitor som gjør det mulig for flere brukere å kjøre BASIC tidsdelt uten masselager. Teoretisk er det ingen begrensning for antall brukere.
2. Muligheter for å lese inn og kalle opp subrutiner på BRF (Binær Relokerbart Format), det vil si programmer skrevet i MAC eller FORTRAN IV. Parametre kan transporteres til og fra BASIC programmet. Dette gjør det mulig å anvende BASIC til for eksempel prosesskontroll.
3. Hvis BASIC kjøres som subsystem under NORD TIME-SHARING SYSTEM kan man direkte benytte det file-system som her eksisterer.

BASIC har vært under utvikling siden 1969. Den første versjon var ferdig i 1971, og inkluderte da 1967-spesifikasjonene fra Dartmouth College. Etter hvert som man har mottatt nye funksjoner har disse blitt implementert. Hovedansvarlig har vært cand. mag. J. Håberg.

SINTRAN II

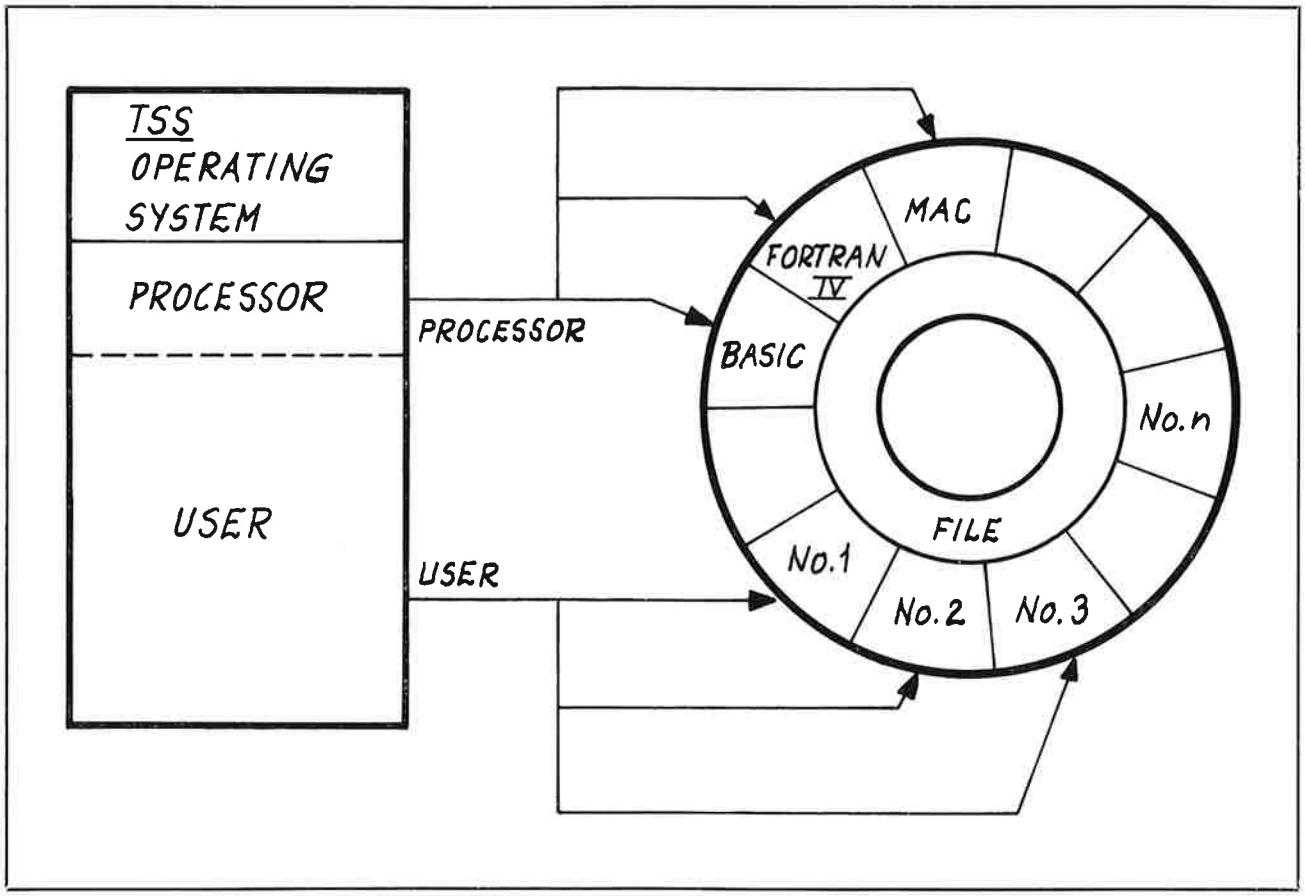
SINTRAN er et sanntids multiprogrammerbart operasjonssystem for NORD datamaskiner. Hovedapplikasjonsområdet er prosesskontroll, men programmet kan bli brukt der det er behov for en sanntids operasjon. Systemet består av monitor, FORTRAN kompilator, BRF Loader med library, og MAC Assembler.
SINTRAN Monitor kontrollerer en rekke brukerprogrammer som kalles RT-programmer (Real-Time programmer). Et RT-program kan startes på et bestemt

Fig. 1. Skjematisk oversikt over SINTRAN operasjonssystem. Blokkdiagrammet viser hvorledes SINTRAN kommuniserer med prosessen, ytre enheter og de enkelte programsystemer.



tidspunkt, startes med en gitt frekvens eller startes som resultat av et eksternt signal.
Utførelsen av et RT-program kan avbrytes av andre RT-programmer som brukeren har gitt høyere prioritet.
RT-programmer kan kodes i assemblykode eller i FORTRAN. Applikasjonsprogrammer for real-time systemer kan derfor skrives i høynivåsproget FORTRAN. Dette kan spare brukeren for meget programmeringsarbeide. ND's FORTRAN kompilatorer, FORTRAN II eller IV, genererer re-entrant kode.
SINTRAN kan kjøres på en minimumskonfigurasjon som består av en NORD-1 med 8K hukommelse, men dersom en virkelig skal dra full nytte av alle muligheter i SINTRAN bør man ha minst 16K hukommelse pluss masselager (tromme eller disk).
Ferdige programpakker (tilpasset SINTRAN) for innlesning og behandling av måleverdier (alarmscanning) og for PID regulering er tilgjengelig fra A/S Norsk Data-Elektronikk.
I et SINTRAN system har en også muligheter for å kjøre bakgrunnsprogrammer. Dette vil si at mens for eksempel et prosesskontrollsystem står on-line, kan en samtidig drive programutvikling på maskinen og kjøre for eksempel FORTRAN jobber, bruke MAC, BASIC, drive med programeditering etc. Da bakgrunnsprogrammene kjøres som RT-programmer med laveste prioritet, vil de ikke forstyrre prosesskontrollprogrammene. NORD-1 memory protect system er utnyttet slik at et bakgrunnsprogram som «løper løpsk» ikke kan ødelegge resten av systemet.
SITRAN I ble utviklet av et team fra SINTEF, NTH, i tiden 1969–1970. Systemet ble senere kjøpt av A/S Norsk Data-Elektronikk og videreutviklet til SINTRAN II. Hovedansvarlig: siv.ing. T. Matre.

Fig. 2. NORD Time-Sharing System. Figuren viser organisering av hukommelse og masselager. Man ser her hvordan den enkelte bruker har adgang til de forskjellige programsystemer. Man kan swappe brukerprogram og systemprogram. BASIC, MAC og QED som er skrevet i re-entrant kode blir liggende fast selv om det er flere brukere som kjører samtidig.



NORD TIME-SHARING SYSTEM (NORD-TSS)
Dette er et generelt tidsdelingssystem for NORD konfigurasjoner med masselager.
Systemet er svært fordelaktig for installasjoner hvor mange brukere trenger umiddelbar adgang til et datasystem. Time-sharing systemet innebærer at hver bruker har tilgjengelig alle maskinens egenskaper uavhengig av andre brukere. Det kan her nevnes at effektiviteten i ND's softwareavdeling økte betraktelig etter at ND's eget datasenter tok systemet i bruk. Terminalbrukeren har tilgjengelig ressurser som vanligvis bare finnes på meget store time-sharing systemer. Systemet er i dag komplett, men det vil bli foretatt en del utvidelser som har vist seg nødvendig ved praktisk bruk.
Systemet inneholder den nødvendige dokumentasjon, slik at en terminalbruker stort sett unngår å konsultere skriftlig dokumentasjon i form av manualer.
Systemet inneholder brukerovervåkning, slik at bare registrerte brukere har anledning til å benytte systemet, passord forhindrer ikke-autorisert bruk.
Følgende oversikt viser noen systemer som er implementert under TSS:

FORTRAN IV
Dette inkluderer også et bibliotek av matematiske funksjoner, og de nødvendige rutiner for kjøring av FORTRAN programmer.

BASIC
System for generering og kjøring av programmer skrevet i sproget BASIC.

QED
Redigeringsprogram for tekst (brukes for dokumentasjon, program og data), programmet er av interaktiv type.
MAC

Oversetter av symbolsk maskinsprog, inneholder også et kommuniserende feil-finningsystem.

MAIL
Kommunikasjonssystem for meldinger mellom brukerne.

ACCOUNTING
System for registrering av hvilke ressurser som er benyttet av individuelle prosjekter og brukere.

FILE SYSTEM
Generelt filesystem som administrerer permanente filer, hjelpefiler og ytre enheter. Tre typer aksess er tilgjengelig: Sekvensiell byte adressering, sekvensiell blokk adressering og tilfeldig blokk adressering. Filebeskyttelse er organisert på tre nivåer: fileeier, godkjent bruker og alment tilgjengelig. For hvert av disse nivåene kan spesifiseres en hvilken som helst kombinasjon av følgende modi: Modifikasjon av filekatalog, skriv aksess og les aksess. Største filestørrelse er 8 millioner tegn.

UTILITY FUNCTIONS
Mange hjelpeprogrammer er tilgjengelig fra brukerterminalen eller fra brukerens program for å forenkle kjøring og uttesting av programmer.

MAC, QED og BASIC er alle skrevet i re-entrant kode, noe som gjør det mulig at systemene ligger fast i core selv om det er flere brukere som samtidig benytter systemet.
I tillegg til MAIL systemet foreligger en kommunikasjonsmulighet mellom brukerne ved at en bruker kan linke seg til en annen.
Sett fra brukerens synspunkt er TSS representert ved to distinkte deler:

1) «Utility Command Processor», som tillater brukeren å manipulere og kontrollere «Brukermaskinen»; behandling av bruker-filer skjer også her.
2) «Monitor Call System» gir brukeren tilgang til alle nyttige brukerfunksjoner som finnes i tillegg til den vanlige NORD maskin som brukeren disponerer.

Kommunikasjon med TSS skjer fortrinnsvis ved hjelp av Teletypes. Tilknytning til systemet kan gjerne skje via telefon og modem. Papirbåndleser og hullkortleser kan også brukes, men er mindre velegnet for primær input. Linjeskriver og hullbåndpunch er tilgjengelig for input.
Den første versjon som ble implementert på ND's datasenter var ferdig i januar 1972. Hovedansvarlig har vært Bo Lewendal.
Ideene til systemet er hentet fra University of California, Berkeley, som utviklet et av de første kommersielle time-sharing systemer i verden, og fra Berkeley Computer Corporation som blant annet har utviklet et avansert time-sharing system for 500 brukere.

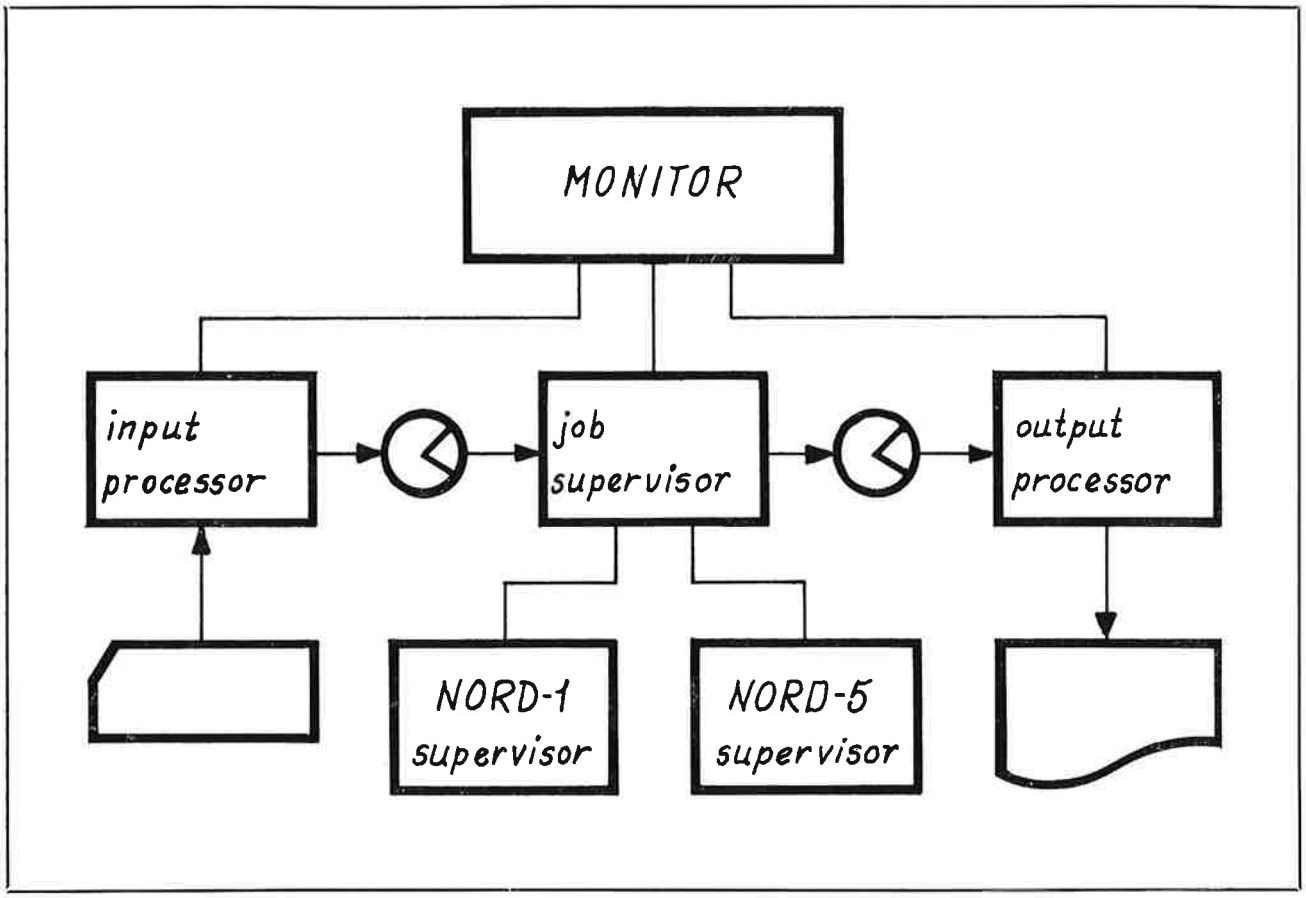
NORD-OPS
NORD-OPS er et batch orientert operasjonssystem. Systemet kjøres under SINTRAN multi-program monitor. Dette gjør det mulig for brukeren å ha seriell kjøring samtidig som SINTRAN blir brukt for sanntidsoppgaver.

NORD-OPS kan brukes ved forskjellige kombinasjoner av NORD-1/NORD-10 og NORD-5 maskiner, disker, trommer, magnetbånd og annet eksternt utstyr. For å optimalisere bruk av eksternt utstyr som kortlesere og linjeskrivere, benytter NORD-OPS buffring av input/output på masselager. Dette forhindrer ikke at systemet kan brukes i «konversasjonsmodus» fra for eksempel display eller annen interaktiv terminal.

En mengde systemrutiner, for eksempel for input/output prosessering, kan kalles opp fra brukerprogrammer.
Jobbene styres ved hjelp av kontrollkommandoer (kontrollkort). Ved hjelp av disse kan brukeren få til sin rådighet for eksempel:
- FORTRAN IV
- MAC (Assembler)
- QED (Editor)
- FILE SYSTEM (diverse rutiner for file behandling)

NORD-OPS er utviklet av et team fra softwareavdelingen hos ND. Utviklingen startet i 1970, og første versjon ble tatt i bruk året etter. Hovedansvarlige har vært cand. mag. J. Vik, og senere siv.ing. K. Nordbye.

Fig. 3. Figuren viser NORD-OPS operasjonssystem for et multianlegg bestående av en NORD-1 og en NORD-5. NORD-OPS er bygget av en rekke supervisor rutiner og kan enkelt utvides til å omfatte nye oppgaver.



NORD FILE SYSTEM

Dette er et generelt file system som har permanente filer, scratch filer og perifer device filer. Det er 3 mulige modi for tilgang til filene: Sekvensiell byte aksess, sekvensiell blokk aksess og random blokk aksess. File sikring har tre nivåer: Owner, Friend og Public. For hvert av disse nivåene kan hvilke som helst av de følgende modi bli brukt: File Directory Modification, Write Access og Read Access. Maksimum file størrelse er 8 millioner tegn.

NORD FILE SYSTEM er en essensiell del i både NORD TIME-SHARING SYSTEM og NORD-OPS operasjonssystemer. Systemet er bygget opp som en katalog som inneholder informasjoner om eier, filenavn og hvor filen befinner seg på masselageret.

Ved å kalle opp forskjellige rutiner i systemet kan man for eksempel skape, fjerne, utvide, åpne og lukke filer.

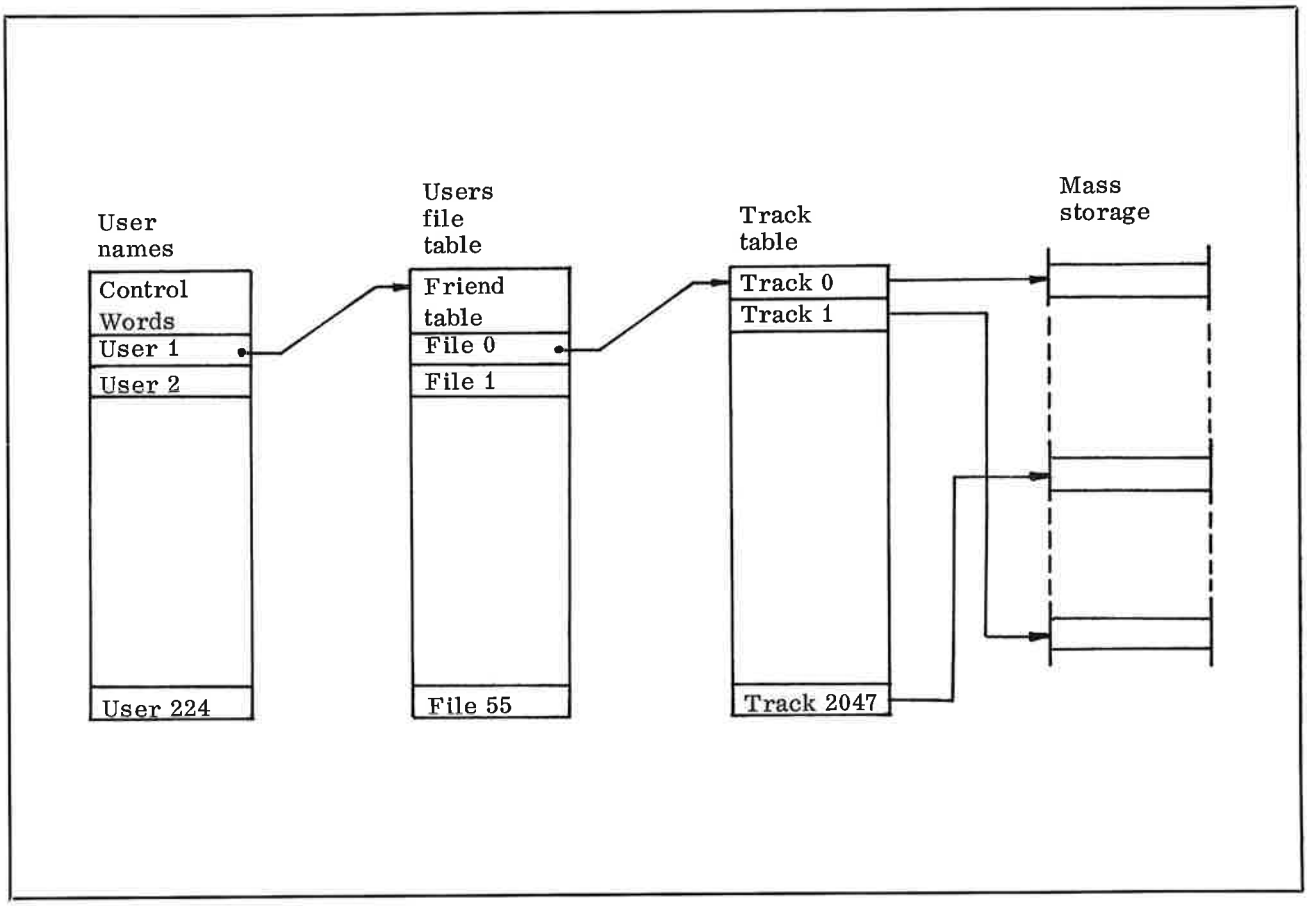
Systemet er utviklet av et team fra softwareavdelingen hos ND, og hovedansvarlig har vært systemprogrammerer Bo Lewendal.

EDITORER

QED

QED er et program for editering av symbolsk tekst En kan sette inn og stryke linjer. Det er mulighet for symbolsk søkning, automatiske tabs som blir bestemt av brukeren, samt en rekke andre hjelpemidler for programmererne.

Fig. 4. Figuren viser organiseringen av file tabeller og brukerområdene på masselageret. Tilkomst til file systemet foregår på brukernavn og file navn.



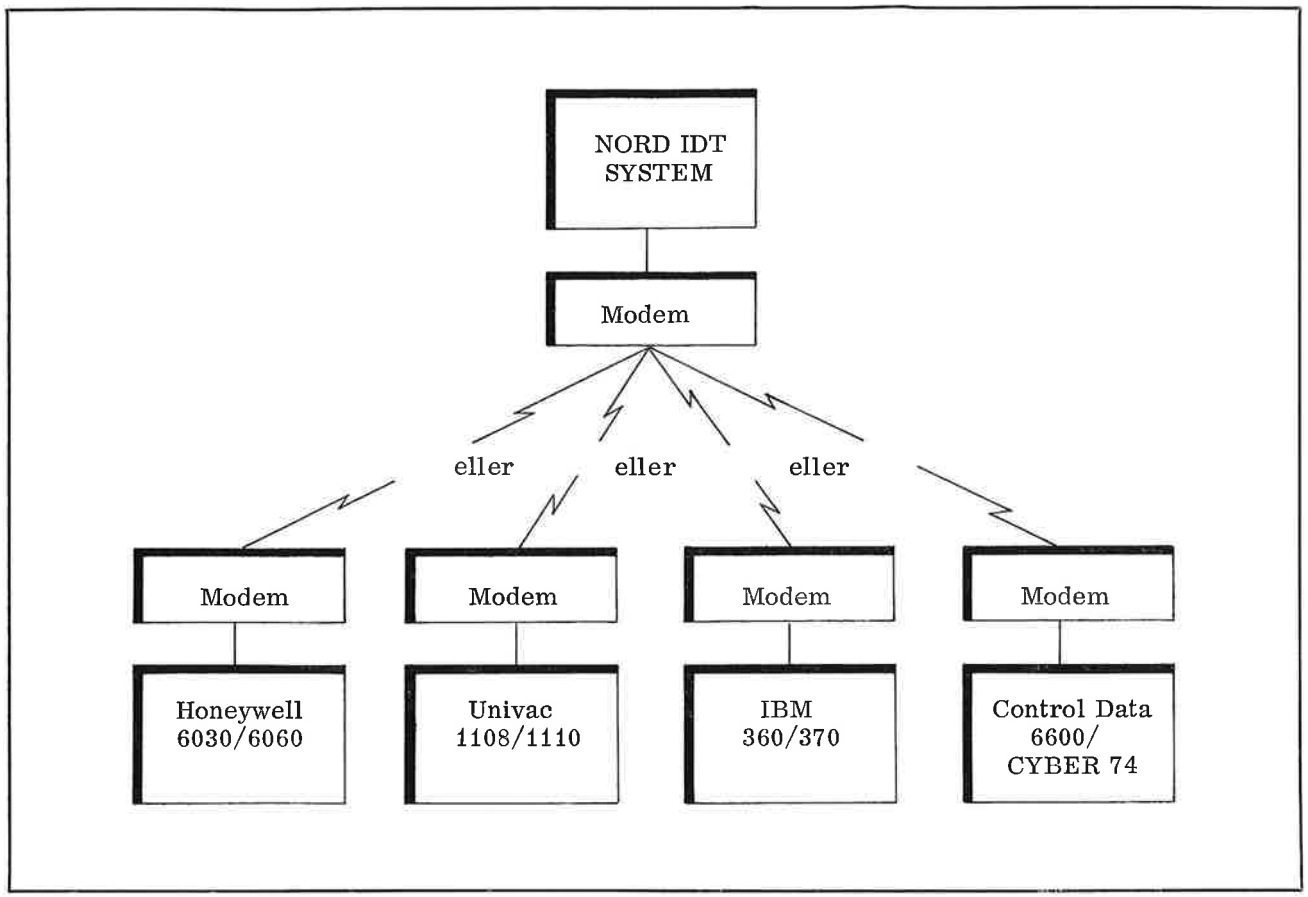
QED står for «Quick and Easy Editor» og er programmert av B. Lewendal (ND).

CONVERSATIONAL EDITOR
CONVERSATIONAL EDITOR gjør det mulig for brukeren å preparere og editere symbolsk tape on-line i ASCII kode ved hjelp av for eksempel en Teletype. Editoren er programmert for ND av siv.ing. P. R. Osmundsen.

ML-EDIT
Multilevel Editor kan som navnet indikerer brukes ved at man anvender rettelser på en allerede innlest rettesekvens osv. Forøvrig skiller editoren seg ut ved at den ikke er av konversasjonell type. Alle rettelser til en file må foreligge som kommandoer til editoren. Ut fra disse kommandoene og original filen, vil editoren produsere en ny file.
ML-EDIT kan også «samarbeide» med for eksempel MAC eller FORTRAN ved at editorens output blir input til assembleren/kompilatoren. Således behøver man ikke generere en ny file før man vet om rettelsene har falt riktig ut.
Programmet er skrevet av cand. real. C. P. Handberg.

NORD IDT — Intelligente DataTerminaler
NORD IDT DCT 2000
NORD IDT DCT 2000 er et program som simulerer Univac DCT 2000 og således gir NORD datamaskin-

Fig. 5. NORD IDT — Intelligente DataTerminaler. Fra NORD IDT kan man ringe opp og kjøre mot hovedanlegg av forskjellige fabrikat.



brukere muligheten til å benytte sin datamaskin som en intelligent batch terminal eller remote-job-entry terminal tilknyttet Univac datamaskiner.
Programmet ble utviklet i 1970 og hovedansvarlig har vært cand. mag. J. Håberg.

NORD IDT CDC 200 USER
Dette er et program som simulerer CDC 200 USERS's Terminal slik at NORD datamaskiner kan benyttes som intelligent terminal til CDC datamaskiner.
Programmet ble utviklet for ND i 1972 av et team fra EDB-sentret, Universitetet i Oslo.

NORD IDT IBM 2780
Dette programmet simulerer IBM 2780 slik at NORD datamaskiner kan brukes som intelligente terminaler til IBM datamaskiner.
Programmet ble utviklet i 1972 med systemprogrammerer T. Paulsen som hovedansvarlig.

NORD IDT GERTS 115
Programmet simulerer GERTS 115 slik at NORD datamaskiner kan benyttes som intelligent terminal til Honeywell Bull's datamaskiner.
Programmet ble utviklet i 1972, og hovedansvarlig har vært siv.ing. H. Madsen.
Disse programpakkene kan kjøres under NORD TIMESHARING SYSTEM.

NORDCOM
NORDCOM er et farve-displaysystem basert på vanlige, kommersielle farve-TV-skjermer. Dette tillater blant annet behandling av billed-output fra datamaskinen med farvebilder fra en annen kilde (for eksempel antenne, kamera, videobåndspiller). Som programmerbar kontroller for systemet brukes en NORD-20 med 4K hukommelse. NORDCOM tilkobles et datamaskinanlegg («hovedmaskinen») med en datakanal som kan skrive inn i, og lese fra NORD-20's hukommelse.
Programsystemet i NORD-20 har som sine to hovedoppgaver å

1) generere grafiske og alfanumeriske farvebilder på grunnlag av data sendt fra hovedmaskinen. Bildene legges i spesielle refresh-hukommelser. Hovedmaskinen kan også beordre et bilde kopiert fra en refresh-hukommelse inn i en annen.
2) betjene periferutrustningen til NORD-20. Denne kan bestå av keyboards, trackerballs, operatørkonsoll, linjeskriver etc.

Hovedmaskinen gir et grafisk bilde ved endepunktskoordinatene for de linjestykkene som skal tegnes. Billedteksten kan gis av en karakterstring som har omtrent samme oppbygning som en FORTRAN format-array. Samme tekstnotasjon kan brukes for rent alfanumeriske bilder.
NORDCOM programsystemet har en modulær oppbygning, og har en maksimal størrelse på ca. 3K. Det ble laget i tidsrommet september 1971 til januar 1972 etter at første del av 1971 var brukt til funksjonsbeskrivelse og produksjon av en foreløpig versjon. Hovedansvarlig har vært cand. real. S. Nyberg.

NORD PLOT PACKAGE
Programmet inneholder assembly og FORTRAN programmerte rutiner for å plotte linjer, kurver, bokstaver, tall, spesialsymboler, forstørre og forminske koordinatsystemer etc.
Samtlige rutiner i plotterpakken kan kalles fra FORTRAN, BASIC og MAC.
Plotterpakken ble utviklet i 1970 og har senere blitt forbedret. Hovedansvarlig har vært systemprogrammerer T. Glavin.

SCIENTIFIC SUBROUTINES
Dette er en programpakke skrevet i FORTRAN. Den inneholder ca. 180 matematiske og vitenskapelige subrutiner. Eksempelvis: Interpolasjon, fourieranalyse, besselfunksjoner, løsning av differensialligninger, etc.

COMMERCIAL SUBROUTINES
Dette er en subrutinepakke som er skrevet i maskinkode, og kan benyttes av FORTRAN programmer. Denne programpakken, pluss FORTRAN IV kan benyttes til oppgaver der en ellers ville ha benyttet COBOL.

GPM
GPM (General Purpose Macrogenerator) er beskrevet av C. Strachey i Computer Journal 8, 3 (Oct. 1965). Denne beskrivelsen er oversatt til MAC Assembly. Programmet finnes både som selvstendig program og under NORD TSS.
GMP er en makroprosessor. Input er en string av karakterer som kopieres direkte til output unntatt når makro-kall påtreffes. Disse makro-kallene evalueres. Rekursive makroer og betingede makroer håndteres lett.
GMP burde især være av interesse for studenter og skoleelever, da den ofte omtales i litteraturen.
Ansvarlig har vært systemprogrammerer T. Paulsen, ND.

STANDARD I/O RUTINER (STIO)
STIO er et program som inneholder standard subrutiner for byte-orientert input/output utstyr. Det er to versjoner av STIO. Den ene versjonen inneholder IOT venteløkker og den andre benyttes ved interruptstyrte og buffret input/output enheter. Hovedansvarlig har vært henholdsvis cand. real. C. P. Handberg og siv.ing. H. Eide.

TRAM
TRAM (Time-shared Reactive Arithmetic Marco System) er en aritmetisk oversetter som gir et generelt konsept av en programmerbar bordkalkulator. Programmet ligner ALGOL og FORTRAN når det gjelder de aritmetiske statementene. TRAM ble utviklet i 1968 med cand. real. T. Amble som hovedansvarlig.

2BCOM
2BCOM er et program for kommunikasjon med NORD-20 eller NORD-2B via Teletype. Programmet er et «Software Control Panel» på NORD-20 og NORD-2B, og simulerer de funksjonene som er på kontrollpanelet på NORD-1. Hovedansvarlig har vært siv.ing. H. Eide.

SIMUL
SIMUL er en programpakke som gjør det mulig å benytte programmer laget for NORD-1 på NORD-20 og NORD-2B. De NORD-1 instruksjoner som mangler i NORD-20 og NORD-2B blir simulert i SIMUL. Hovedansvarlig har vært siv.ing. H. Eide, ND.

Den oversikt som er gitt foran er ment å inneholde de standard programsystemer som oftest blir brukt i NORD datamaskinsystemer. En oppramsing av andre viktige systemer følger:

1. Assembler for NORD-5.
2. FORTRAN IV system for NORD-5.
3. Programpakke for landmåling (omtalt i ND-NYTT nr. 3).
4. Rutiner for dobbeltpresisjons floating point aritmetikk.
5. MINIMON – en liten sanntids monitor for prosesskontroll applikasjoner.
6. Rutiner for innlesing, filtrering, grenseverdisjekking etc. av måleverdier. Tilpasset SINTRAN II og MINIMON.
7. Diverse hardware testprogrammer.
8. PL sprog for NORD-5.
9. PL-sprog for NORD-1.
10. NODAL (Nord All Purpose Language). Sanntids interpretativt høynivåsprog.

SOFTWARE ORGANISASJON HOS ND
Alle programmer blir fortløpende registrert i en såkalt PD katalog ved et PD nummer (PD = Program Deck). Katalogen inneholder en kort beskrivelse av programmet; størrelse, symbolsk, binær, etc. Selve programmet oppbevares i PD arkivet.
I ND's Software Catalogue registreres siste versjon av de forskjellige systemer. Katalogen består av en en-sides beskrivelse av de forskjellige programmene. Disse er igjen gruppert etter software kategori. Beskrivelsen består blant annet av PD nummer, henvisning til dokumentasjon, ansvarshavende. Katalogen følger alle NORD maskiner ved leveranse, og oppdateringer blir sendt ut sammen med nye eller rettede programmer. Det er viktig at katalogen oppdateres.

ND deler software i fire klasser:

A **Software som følger maskinen**
Denne software vedlikeholdes av ND. Oppdaterte versjoner og dokumentasjon sendes ut uten vederlag. Ekstra kopier av program eller dokumentasjon kan bestilles til reproduksjonspris.

B **Software til kopieringspris**
Denne software vedlikeholdes av ND. Kan bestilles til priser oppgitt i Software Catalogue.

C **Software uten plikt til vedlikehold**
Kan bestilles til priser oppgitt i Software Catalogue.

D **Software til spesialpris**
Denne software vedlikeholdes av ND. Prisen er avhengig av eventuelle avgifter og utviklingskostnader.

Software og dokumentasjon kan bestilles ved henvendelse til ND's Software avdeling ved Aud Sæstad.

For feilrapportering har ND i samarbeid med NOCUS utarbeidet et feilrapporteringssystem. Rapporteringen skal være skriftlig og utført på følgende skjemaer: «Software System Field Report» som brukes for programmer, og fås ved henvendelse til ND's software avdeling, og «Comment and Evaluation Sheet» som brukes for dokumentasjon. Dette skjema følger alle manualer som utgis av ND.
En feilliste for hver programart skal distribueres av ND. Denne gir veiledning om hvordan feil kan unngås eller eventuelt midlertidig kan rettes.

NOCUS
NORD COMPUTER USERS SOCIETY

AV MORTEN HELDAL HAUGERUD, FORMANN

For nye lesere av ND-NYTT kan en presentasjon av NOCUS være påkrevet. NOCUS er en organisasjon hvor alle brukere av NORD datamaskiner har anledning til å bli medlemmer. Det har vært avholdt brukermøte 2 ganger årlig siden det konstituerende møtet på Røros 15.–16. oktober 1970 hvor følgende formålsparagraf ble gitt:

«Formålet med NOCUS er å hjelpe brukerne av NORD datamaskiner til å utnytte maskinene optimalt. Organisasjonen skal skape kontakt mellom brukerne og ivareta felles interesser overfor A/S Norsk Data-Elektronikk.»

NOCUS omfatter i dag 44 institusjoner, private såvel som statlige.
De fire første møtene bestod i vesentlig grad av plenumsforedrag med orienteringer om brukernes og ND's daværende og fremtidige prosjekter. På brukermøtet høsten 1972 innførte man imidlertid et annet opplegg, hvilket blir beskrevet i følgende referatsammendrag.

Brukermøtet høsten 1972
Møtet ble holdt på Hotell Alexandra, Loen, i tidsrommet 1.–3. oktober 1972, der 84 deltagere fra 37 institusjoner var samlet. Det var denne gang, i motsetning til tidligere møter, avsatt vesentlig tid til gruppearbeid, hvor følgende emner ble tatt opp:

Gruppe 1: NORD som terminalmaskin
Gruppe 2: NORD i undervisning
Gruppe 3: NORD i prosesskontroll
Gruppe 4: Drift og vedlikehold
Gruppe 5: NORD i skipsmiljø
Gruppe 6: Prosjektarbeid
Gruppe 7: NORD i forskningsmiljø
Gruppe 8: NORD software

Dette opplegget ble meget godt mottatt av brukerne som betegnet det som utbytterik. Gruppene la frem ialt 5 resolusjoner som kort summert hadde følgende henstillinger til ND:

1. Sørge for at NORD-10 software også kan benyttes på NORD-1.
2. Opprette programbibliotek for applikasjonsprogrammer.
3. Frembringe mer utførlige spesifikasjoner til kraftforsyning, jording, og miljø for øvrig.
4. Tilpasse utstyret for batteridrift og bufferbatterier.
5. Anmelde lærebøker i datateknikk i ND-NYTT.

De viktigste plenumsinnleggene var presentasjonene av ND's vedlikeholdstilbud og av NTNF-prosjektet DUPP (**D**elvis **U**bemannede **P**roduksjons**P**rosesser). Begge innleggene skapte engasjert debatt som viste at det var grunnlag for videre diskusjon om disse emnene.
Møtet skilte seg også fra de tidligere ved at et sportslig innslag var tatt med i programmet. I det flotte høstværet ble det en opplevelse og en fin avkobling med en fottur til Briksdalsbreen. Deltagerne fikk for øvrig denne gang bedre anledning til å komme i kontakt med hverandre, noe mange fant verdifullt.
Det fullstendige referat er gjennomgått av styret, og er sendt medlemsinstitusjonene.

Det nye styret
På møtet ble det valgt nytt styre for kommende 1-årsperide. Styret har nå konstituert seg, og har følgende sammensetning:

Formann
Morten Heldal Haugerud, Computas A.S, Oslo
Nestformann
Sverre Maudal, NEBB, Oslo
Sekretær
Audun Sæthre, Elkem-Spigerverket A/S, Oslo
Styremedlemmer
(kasserer) Bjørn Halseth, Oslo Oppmålingsvesen, Oslo
Jon M. Sørland, Sørlandets tekniske skole, Grimstad

Ingen av disse var tidligere styremedlemmer, men formannen var varamann i det avgåtte styret.
De viktigste sakene det nye styret har tatt opp er:

- Oppfølgning av feilrapporteringsskjemaene.
- Oppfølgning av arbeidet med service/vedlikeholdskontrakt.
- Arrangere møte om DUPP i samarbeid med NSEI (Norsk Selskap for Elektronisk Informasjonsbehandling).
- Vurdere ND's svar på resolusjonene.

— Vurdere eventuell opprettelse av programbibliotek for applikasjonsprogrammer.
— Neste brukermøte.

Feilrapporteringsskjemaene
På bakgrunn av at brukerne har hatt vanskelig for å få informasjon om feil som andre brukere har funnet i sitt ND-utstyr (spesielt software), vil styret anmode på det innstendigste om at medlemmene benytter «Software System Field Report»-skjemaene i vesentlig større grad enn tidligere. Skjemaene må benyttes både ved feil og misforståelser (feil i manualer).
For at informasjon om feil skal tilflyte brukerne på en måte de har glede av har styret fremmet et forslag overfor ND som i korhet går ut på følgende:

«Det foreslås at en oversikt over registrerte feil utgis med 14 dagers mellomrom. Oversikten bør inneholde:

Angående softwarefeil: Det gis en kort beskrivelse av alle feil i siste periode enten de er løst eller ikke. Når løsningen på gamle feil finnes tas denne med.

Angående hardwarefeil: Det gis en kort beskrivelse av de design-feil, marginaliteter o.l. som er funnet og hvilke versjoner av maskinene de gjelder.

Feilstatistikk: Denne foreligger halvårlig.
ND's innstilling til dette forslaget var positiv, og det er nå viktig at brukerne følger opp med å sende inn feilmeldingsskjemaer. Det vil da også være mulig å få en ende på det stadig tilbakevendende fenomen at «Dette er jo en feil som man har kjent lenge».

Service/Vedlikeholdskontrakter
Styret har funnet det gunstig å trekke inn ressurser utenfor styret, og Jan Caspersen, NEBB, har sagt seg villig til å bearbeide denne saken.

NOCUS-møtet våren 1973
Neste møte skal finne sted på Caledonien Hotell, Kristiansand i tidsrommet 4.—6. mai 1973. Programmet er under utarbeidelse, og styret vil gå inn for å følge opplegget fra forrige møte ved å holde plenumsinnlegg, parallellsesjoner og gruppemøter. Som hovedemner vil man søke å legge vekt på:

— Debuggingsteknikk
— Sosiotekniske forhold for den vanlige arbeidstaker ved innføring av datamaskinteknologi
— Norsk datamaskinindustris fremtid

I forbindelse med emnet Debuggingsteknikk er styret meget interessert i å få vite hvordan brukerne ser på de debuggingsmulighetene som ND i dag leverer for NORD datamaskiner, og hvor fornøyde de er med disse mulighetene. Styret er også interessert i å få vite i hvor sterk grad brukerne har laget debuggingsmuligheter selv.
Vi håper NOCUS' medlemmer har kommentarer til dette eller andre punkter i NOCUS' virksomhet, og ser gjerne at det tas telefonisk eller skriftlig kontakt med styret. NOCUS' adresse er:

NOCUS
c/o Morten Heldal Haugerud
A.S Computas
Økernveien 145
OSLO 5
Telefon (02) 22 01 55

Data Prosessering med Kalman-filteret

DR. PHILOS. BENT AASNÆS, A/S INFORMASJONSKONTROLL

Bent Aasnæs, født i Oslo 1939. B.S. i elektro ved M.I.T., 1964. M.S. i elektro ved Stanford University i 1965. Arbeidet med prosess-kontroll utstyr ved IBM, Sand Jose 1965—67. Ansatt som stipendiat ved Reguleringsteknikk, NTH 1967—68. Ph.D. i elektro ved Stanford University 1971. Fra 1971 ansatt ved A/S Informasjonskontroll og arbeider med flykontroll og prosesskontroll systemer. Publisert 4 vitenskapelige artikler i kontroll-teori, kontroll-utstyr og statistisk databehandling.

Veien fra konsept til praktisk anvendelse er ofte trang og kronglete. Kalman-filteret så dagens lys tidlig i 60-årene [1], [2], [3], men det er først i de siste fem år eller så at metoden er blitt tatt i bruk i industrien. Forsinkelsen skyldes primært at det først i de senere år er blitt vanlig å benytte en datamaskin i sanntidssystemer. En annen grunn er kanskje en manglende forståelse blant ingeniørene av hva Kalman-filteret er.

Hva er Kalman-filtrering?

Kalman-filtrering er «optimal» rekursiv prosessering av støybelagte måledata.

La oss anta at vi ønsker å estimere verdien av en ukjent størrelse $\chi(t)$ på grunnlag av observasjoner. Disse observasjonene er belagt med målestøy, slik at en nøyaktig beregning av $\chi(t)$ er umulig. Det beste man kan prøve er å prosessere observasjonene slik at disse gir en verdi for $\chi(t)$ som er, gjennomsnittlig, så nær som mulig den sanne verdi.

Det kan for eksempel tenkes at vi ønsker å finne posisjon og hastighet av et skip, slik at

$$\chi(t) = \left\{ \begin{array}{l} \text{avstand til skipet ved tiden t} \\ \text{peiling til skipet ved tiden t} \\ \text{kurs ved tiden t} \\ \text{fart ved tiden t} \end{array} \right\} \quad (1)$$

Observasjonen ved tiden t, som vi kaller $\gamma(t)$, kan bestå av vår avstandsmåling og peiling

$$\gamma(t) = \left\{ \begin{array}{l} \text{avstand ved tiden t} \\ \text{peiling ved tiden t} \end{array} \right\} + \text{målestøy}$$

Kalman-filteret gir det estimatet av $\chi(t)$ som er en lineær funksjon av de foregående observasjonene γ_t, $\gamma_{t-1}, \ldots$, og som har den minste feilkovariansen. Det vil si at den forventede verdi (populært: gjennomsnittsverdi) av kvadratet av feilen er minst mulig. Dette er kalt lineær minste kvadrats estimering. Intet er nytt her. Gauss brukte denne metoden i 1795 i forbindelse med baneberegninger for planeter. Lineær regresjonsanalyse er basert på det samme prinsipp. Videre er Nobert Wiener's berømte filter av denne typen.

Hva er så annerledes med Kalman-filteret? Forskjellen består i at Kalman postulerer en modell — den såkalte Markov modellen — for prosessen. Denne modellen er en differensialligning, hvor hvit inngangsstøy u(t) er pålagt prosessen

$$\chi(t+1) = \Phi_t\chi(t) + u(t) \quad (2)$$

og hvor hvit målestøy v(t) er pålagt målingene

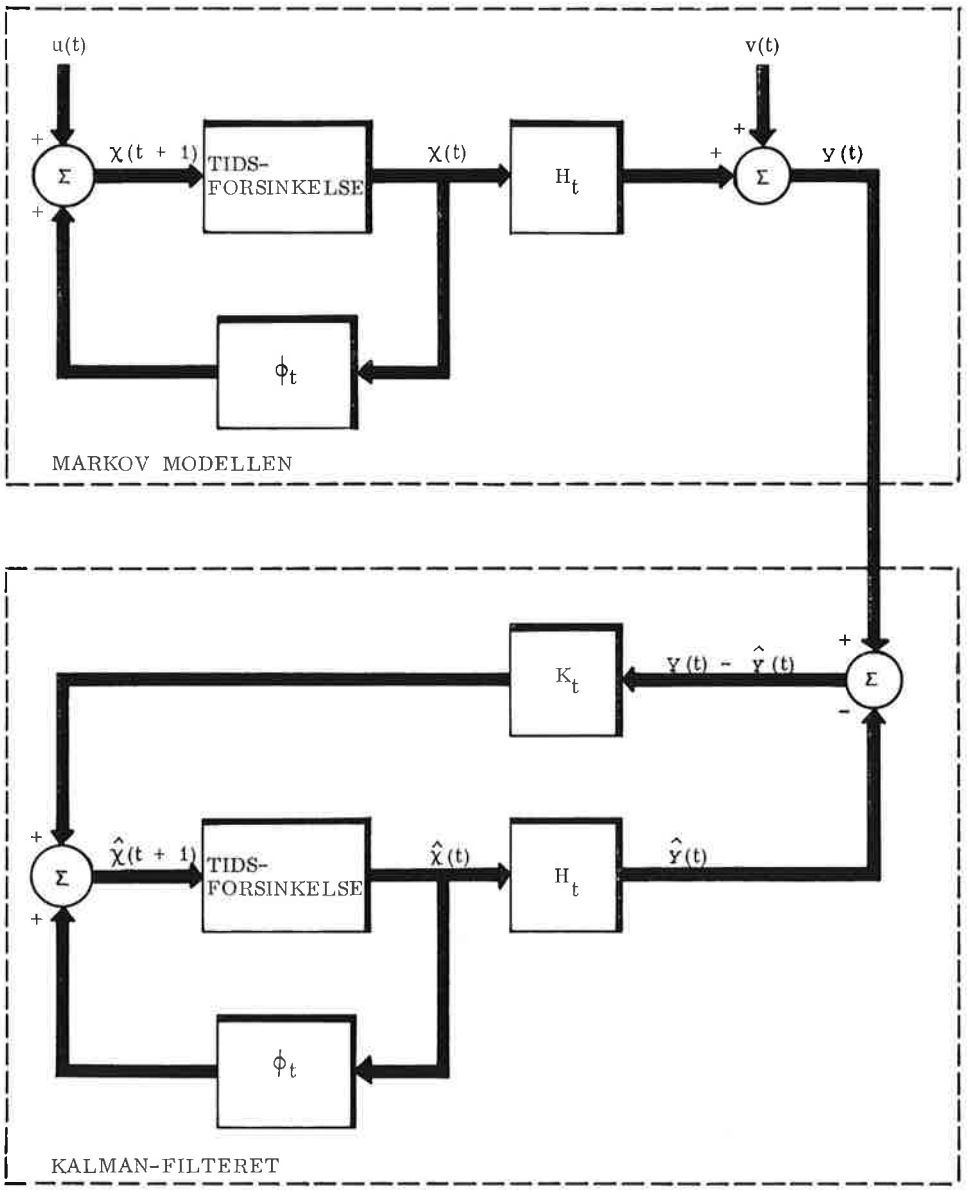
$$\gamma(t) = H_t\chi(t) + v(t) \quad (3)$$

Her er Φ_t og H_t kjente koeffisient-matriser som karakteriserer prosessen. Tilstandsvektoren $\chi(t)$ består av de variablene som er nødvendig for fullstendig å beskrive prosessens fremtidige adferd når støyen u(t) er lik null. Posisjon og kurs av et skip i rettlinjet bevegelse med konstant hastighet bestemmer fullstendig skipets fremtidige posisjoner og (1) er derfor en tilstandsvektor i dette tilfellet. Andre eksempler på Markov-prosesser er kjemiske prosesser hvor tilstandsvektoren beskriver temperatur, trykk og konsentrasjoner; økonomiske systemer hvor $\chi(t)$ gir produksjonskapasiteter og lagerbeholdninger; biologiske systemer hvor $\chi(t)$ representerer konsentrasjoner av en medisin i forskjellige organer.

Ikke alle prosesser er Markov-prosesser. Dette er således en begrensning i Kalman's teori. De aller fleste prosesser av interesse kan imidlertid modelleres som (2) og (3), enten nøyaktig eller tilnærmet ved hjelp av linearisering. I eksemplet med skipet, hvis skipperen legger roret til styrbord eller babord etter som han trekker grønne eller røde kuler ut av en Polya urne*, gir dette i k k e en Markov-prosess.

* En Polya urne kan for eksempel konstrueres slik: Til å begynne med inneholder urnen en grønn og en rød kule; trekker man en rød (grønn) kule, legges to røde (grønne) tilbake.

Blokk-diagram av Markov prosessen, (2), (3) og av Kalman-filteret. Diagrammet illustrerer likheten mellom prosess modellen og filteret: Filteret er selv en Markov prosess. Dette er den fundamentale årsak til at Kalman-filteret er regnemessig mere effektiv enn de andre «optimale» metodene. Transformasjonen K(t) i diagrammet utregnes fra estimatets feilkovarianse, som finnes fra en separat rekursiv ligning.



Men det er heller ikke en prosess av åpenbar praktisk interesse.

Er Kalman-filteret anvendelig?
Siden Wiener-filterets oppdagelse i 1942 har bare ganske få og trivielle problemer funnet sin løsning i lukket form ved denne metoden. Enn videre, filteret egner seg dårlig for numerisk løsning. Wiener's bidrag lå på det konseptuelle og ikke på det praktiske plan.
Ved å forkaste ønsket om en filter-løsning i lukket (analytisk) form, oppnådde Kalman et filter som egner seg suverent for numerisk løsning. Alle ligningene i filteret er rekursive, en iterasjon foregår hver gang en ny observasjon kommer inn, og kun det siste estimat og dens kovarianse lagres. Gitt modellen (2), (3) kan således alle filtreringsproblemer løses. Det er kun et spørsmål om regnekapasitet. I motsetning til Wiener-filteret kan Kalman-filteret med letthet behandle tilfellene hvor observasjonene foregår kun over et kortere tidsrom, og hvor prosess-modellen er tidsvariant. Disse viktige fordelene oppstår fordi Kalman-filteret opererer i sann tid, mens Wiener fant det nødvendig å gå til frekvensdomenet for å løse sin integralligning. Et annet aspekt ved Kalman-filteret er at det er spesielt velegnet når også en automatisk kontroll-funksjon skal inkorporeres. Optimal kontroll betinger en tilstandsvariabel modell av formen (2), (3). I de tilfeller hvor tilstandsvektoren er nøyaktig kjent kalkuleres kontroll-pådraget ut fra verdien av $\chi(t)$. Dersom $\chi(t)$ ikke kan observeres direkte på grunn av støy – dette er den vanlige situasjonen – beregnes kontroll-pådraget ut ifra Kalman-estimatet av $\chi(t)$ [4].

Hvor godt er Kalman-filteret?
Kalman-filteret blir ofte kalt «optimalt». Spørsmålet er, optimalt i hvilket henseende? Det finnes jo mange andre estimeringsmetoder som gjør krav på eksistensberettigelse: «maximum likelihood», «stochastic approximation», «instrumentar variable», o.s.v. Det finnes ikke noe entydig svar på hvilken metode som er «optimal». Alt avhenger av situasjonen. En vanlig situasjon er imidlertid at en lineær tilstandsvariabel modell kan tilpasses prosessen og at støyen er Gaussisk fordelt. I så fall vil Kalman-filteret og «maximum likelihood» gi identiske estimater, og disse vil være «bedre» enn de øvrige (så og si uansett hva man mener med bedre). Når tilstandsvariabel-modellen er ulineær, eller når støyen ikke er Gaussisk, vil «maximum likelihood» ofte gi de beste estimater, men denne modellen vil være vanskelig eller umulig å anvende i praksis. Kalman-filteret vil da ofte være et godt kompromiss mellom det ønskelige og det mulige.

Hvor er Kalman-filteret anvendt?
En kritisk betingelse for anvendbarheten er – som vi har understreket – at modellen (2), (3) er kjent. Dette kan ikke tas som noen kritikk av Kalman-filteret. Andre estimeringsmetoder krever korrelasjonsfunksjoner eller til og med komplette fordelingsfunksjoner. Igjen vil situasjonen være avgjørende for hvilken prosess-beskrivelse som er lettest å finne. I målfølging og navigasjon er en tilstandsvariabel modell naturlig. Det er her Kalman-filteret har funnet sin største utbredelse til nå. Dette gjelder særlig følging av satelitter, raketter, fly, skip, undervannsbåter, og navigasjon for raketter, fly og undervannsbåter [5]. Kalman's metode vinner innpass på prosess-kontroll området, men en vanskelighet her kan være å finne en passende modell.

Praktiske problemer ved Kalman-filtrering
En slags Parkinson's lov for systemingeniører er at høyere systemytelser betyr større sårbarhet overfor svikt i konstruksjonspremissene. En demonstrasjon på dette prinsipp kan være Kalman-filteret. Små feil i modellen kan føre til alvorlige feil i estimatet, og endog til divergens av algoritmene. En intens forskning har pågått de siste 10 år på dette området, og det eksisterer i dag en rekke metoder for å identifisere modellen [6] og for å sikre stabilitet av Kalman-filteret [7]. Nøyaktig ingeniørarbeid og simulering er nødvendig for å oppnå gode resultater. Det er i dag ingen kokebokoppskrift for implementering av Kalman-filtre.
Et annet problem er at Kalman-metoden kan kreve mye regnekapasitet. Dette avhenger primært av modellens størrelse. Et typisk filtreringsproblem, som eksemplet med følging av et skip, løst på en NORD-1 krever kanskje 2K med programhukommelse og eksekveringstid på 100 msek for hver observasjon. Det finnes imidlertid prosedyrer som drastisk reduserer kapasitet-kravet med bare moderate reduksjoner i ytelsen [5]. Et sundt prinsipp er å starte med et «optimalt» Kalman-filter og dernest finne – gjerne ved simulering – forenklinger i modellen og i filteret som resulterer i akseptable reduksjoner i ytelsen.

Referanse-liste

1. Kalman, R. E., «A new approach to linear filtering and prediction problems», Trans. ASME, J. Basic Engrg., 83, mars 1960, pp. 34—45.
2. Kalman, R. E. og R. S. Bucy, «New results in linear filtering and prediction theory», Journal of Basic Engineering, 83, mars 1961, pp. 95—108.
3. Kalman, R. E., «New methods in Wiener filtering theory», Proc. Symp. Eng. Appl. Random Function Theory and Probability, (J. L. Bogdanoff og F. Kozin utgivere), Wiley, New York, 1963.
4. Bryson, A. E., Jr. og Y. C. Ho, «Applied optimal Control», Ginn Co, 1969.
5. AGARD ograph 139, NATO, «Theory and applications of Kalman filtering», utgiver C. T. Leondes, februar 1970.
6. Åstrøm, K, og P. Eykhoff, «System identification — a survey», Automatica, vol 7, 1972, pp. 123—162.
7. Jazwinski, A. H., «Stochastic processes and filtering theory», Academic Press, 1970.

Otto Falch - Oslo