Planlægning af EDB-systemer

En omtale af forskellige hjælpemidler

Af T. SKOUSEN*) *og* JOHN D. ANDERSEN**)

Formålet med denne artikel er at beskrive forskellige hjælpemidler til vurdering af edb-anlægs materiel. I denne forbindelse er to problemområder, der illustrerer forskellige niveauer for betragtningsmåden af edb-systemer, belyst:

1) Analyse og vurdering af et nyt edb-system på tilbudsstadiet.

2) Analyse og vurdering af materieludbygning og ændring af driftsform for et bestående edb-system.

Det må fremhæves, at de omtalte hjælpemidler kan benyttes på alle analyseniveauer, samt at det ene (simulering) er et generelt analyseværktøj, der kan benyttes til mange andre problemstillinger, f. eks. inden for trafik.

1. *Indledning.*

Planlægning af edb-anlæg byder, hvad enten der er tale om etablering af et nyt system eller blot tale om ændringer af et bestående anlæg, på meget store problemer.

Et elektronisk databehandlingsanlæg er opbygget af en lang række fysiske og elektroniske komponenter (materiellet) og omfatter derudover normalt et større antal styre- og hjælpeprogrammer (programellet), som leverandøren stiller til rådighed.

Til illustration er på fig. 1 vist den fysiske opbygning (materiellet) af edb-anlægget ved Danmarks tekniske Højskole (regnecentret NEUCC) før udbygningen af anlægget i foråret 1968.

*) Civilingeniør, lic. techn., A/S HELLESENS.
**) Civilingeniør, Administrationsrådets Sekretariat.

Planlægning af EDB-systemer

En omtale af forskellige hjælpemidler

Af T. SKOUSEN*) *og* JOHN D. ANDERSEN**)

Formålet med denne artikel er at beskrive forskellige hjælpemidler til vurdering af edb-anlægs materiel. I denne forbindelse er to problemområder, der illustrerer forskellige niveauer for betragtningsmåden af edb-systemer, belyst:

1) Analyse og vurdering af et nyt edb-system på tilbudsstadiet.

2) Analyse og vurdering af materieludbygning og ændring af driftsform for et bestående edb-system.

Det må fremhæves, at de omtalte hjælpemidler kan benyttes på alle analyseniveauer, samt at det ene (simulering) er et generelt analyseværktøj, der kan benyttes til mange andre problemstillinger, f. eks. inden for trafik.

1. *Indledning.*

Planlægning af edb-anlæg byder, hvad enten der er tale om etablering af et nyt system eller blot tale om ændringer af et bestående anlæg, på meget store problemer.

Et elektronisk databehandlingsanlæg er opbygget af en lang række fysiske og elektroniske komponenter (materiellet) og omfatter derudover normalt et større antal styre- og hjælpeprogrammer (programellet), som leverandøren stiller til rådighed.

Til illustration er på fig. 1 vist den fysiske opbygning (materiellet) af edb-anlægget ved Danmarks tekniske Højskole (regnecentret NEUCC) før udbygningen af anlægget i foråret 1968.

*) Civilingeniør, lic. techn., A/S HELLESENS.
**) Civilingeniør, Administrationsrådets Sekretariat.

Anlæggets egenskaber er en funktion af et kompliceret samvirke mellem materiel og programmel, men hertil kommer, at brugeren i høj grad selv bestemmer reglerne for dette samvirke gennem de programmer, der skrives, og den måde hvorpå de afvikles. Udnyttelsen af anlægget er derfor stærkt afhængig af kvaliteten af brugerens programmer samt den organisation, der opbygges omkring anlægget.

Ved planlægning må man tage stilling til, hvilke komponenter der skal indgå i systemet, hvilke egenskaber disse komponenter skal have, samt hvorledes komponenterne skal sættes sammen.

Det samlede edb-anlægs egenskaber kan, set fra brugerens synspunkter, udtrykkes på forskellig måde. Nogle af de hyppigst benyttede målinger er: Behandlet datamængde (Throughput), gennemløbstid (Turnaround time) og driftssikkerhed.

»Throughput« måler det stationære systems arbejdskapacitet udtrykt ved f. eks. antallet af lønningslister fremstillet pr. time eller antallet af Cobolsætninger oversat pr. minut. For en given bruger vil det give mere mening at bestemme »throughput« som den hastighed, hvormed hans specielle sæt af opgaver behandles.

I mange tilfælde vil »throughput« imidlertid være af mindre betydning end gennemløbstiden, der kan defineres som den tid der går, fra en opgave afleveres til systemet, og til resultaterne foreligger. Det kan f. eks. være svartiden for et pladsreservationssystem eller den tid, en programmør må vente på resultatet fra en testkørsel.

Ved driftssikkerheden forstås sandsynligheden for, at systemet vil fungere tilfredsstillende til et givet tidspunkt.

I denne forbindelse vil man kun betragte de to første målinger, idet den sidste er af en sådan karakter, at den ikke kan vurderes ved hjælp af de metoder, der skal behandles her.

Det må fremhæves, at »overhead«, defineret som systemets reaktionstid (f. eks. den systemafhængige tid, der går fra en opgave afsluttes til den næste påbegyndes), ikke er et fundamentalt mål for et systems egenskaber. I den senere tids anvendelse af udtrykket har det været underforstået, at overhead nødvendigvis var af det onde. Der findes dog intet der viser, hvor stor en overhead tid, der er rimelig, og endvidere er det ikke systemkonstruktørernes mål at minimere overhead, men snarere at formindske gennemløbstiden og forøge throughput. En forøgelse af overhead kan derfor udmærket være ønskeligt, hvis det fører til disse mål.

Kendskab til overheadtiden kan, hvis den overhovedet kan defineres, være en hjælp ved analysen af et system, men den bør betragtes som en

beskrivende parameter og ikke som et afgørende mål for systemets egenskaber.

En analyse af et edb-anlæg med henblik på at vurdere de ovennævnte egenskaber er vanskelig, hvad enten man betragter hver enkelt komponents egenskaber eller forsøger at analysere det samlede system. Målingerne giver nemlig udtryk for en vekselvirkning mellem på den ene side edb-anlægget og den anden side de opgaver, brugeren lader anlægget behandle.

Hvorledes en sådan analyse gribes an afhænger af problemstillingen.

Er man interesseret i problemerne omkring et monitorsystem, d. v. s. et automatisk overvågningsprogram, vil det være naturligt at se på de enkelte komponenter, selve datamaskinen består af, f. eks. hurtiglager, datakanaler og registre. Dette vil give mulighed for at undersøge, om maskinen internt arbejder rationelt. Den enkelte brugers opgaver vil her være af mindre interesse.

Er problemstillingen derimod organiseringen af et datacenter, d. v. s. udarbejdelsen af beslutningsregler for driften, så vil det være mere rationelt at betragte hele datamaskinen som en komponent med givne egenskaber, fordi denne arbejder automatisk og kun kræver, at dens funktioner overvåges af en mand. Et edb-system betragtet fra et sådant niveau vil udgøres af de enheder, der eksisterer set fra et betjeningsmæssigt synspunkt samt de beslutningsregler, der findes for driften. Typen af de opgaver, edb-anlægget skal behandle, er det nødvendigt at kende for at kunne løse denne problemstilling.

2. *Metoder til sammenligning af edb-anlægs enkelte elementer.*

De maskinelle enheder, hvoraf et anlæg er opbygget, kan lidt groft opdeles i tre grupper: Ind- og udlæseudstyr, hjælpelagre og centralenheden, omfattende styreenheden, regneenheden og hurtiglageret.

For de to første gruppers vedkommende vil en sammenligning normalt være relativt simpel, idet antallet af parametre, der er relevante ved sammenligningsproceduren, er forholdsvis beskedent.

Således vil f. eks. en linieskriver set fra det her anlagte synspunkt i det væsentlige være bestemt ved antal linier pr. tidsenhed, antal tegn pr. linie, antallet af mulige tegn og af kvaliteten af det udskrevne materiale. Hertil kommer naturligvis skriverens pålidelighed under forskellig belastning. Dette kan normalt kun bestemmes statistisk.

Centralenheden vil i sammenligning hermed normalt være langt mere kompleks, og de undersøgelser, der må foretages for at sammenligne for-

skellige centralenheder derfor langt vanskeligere. Nogle af de metoder, der anvendes ved en sådan vurdering, er nævnt i de følgende afsnit.

2.1. *Lagerets cyklustid.*

Da alle et programs ordrer og data normalt hentes fra datamaskinens hurtiglager, vil en vigtig parameter ved bedømmelsen af en maskines interne hastighed være lagerets cyklustid. Herved forstås der den tid, der går fra det øjeblik en impuls til udskrift fra lageret er afgivet fra styreenheden, og indtil det igen er muligt at foretage udskrift fra lageret. Den tid, der går indtil ordet er læst ud i det register, der skal modtage data fra lageret, kaldes tilgangstiden (accestime), og er for et normalt ferritkærnelager kun en del af den fulde cyklustid (jfr. litt. 1). Tilgangstiden anvendes ofte synonymt med cyklustiden (jfr. litt. 6).

I tabel 1 er vist nogle eksempler på tider for forskellige maskiners lagercyklus.

Tabel 1. Lagerkarakteristika.

	Cyklustid μs	Antal bit pr. cyklus	Overlapning
IBM 360/30	2.0	8	nej
– 360/40	2.5	16	nej
– 360/50	2.0	32	nej
– 360/60*	2.0	64	ja
– 360/62*	1.0	64	nej
– 7090	2.18	36	ja
– 7094 II	1.4	36	nej

En ukritisk anvendelse af cyklustiden vil give, at en 360/30 var lige så hurtig som 360/60 og hurtigere end en 7090. Dette er dog langt fra tilfældet; i virkeligheden er model 60 ca. 20 gange hurtigere end en model 30 (jfr. litt. 7).

Der er altså en hel del andre forhold, der må tages i betragtning. Således er det f. eks. afgørende, hvor meget information, d. v. s. hvor mange uafhængige bit, der overføres fra lageret under en cyklus. Dette er ligeledes vist i tabel 1. Et andet forhold, der spiller ind ved beregning af den effektive cyklustid, er overlapning af cykler til to eller flere lagermoduler. Således skulle man ved skiftevis at hente data ud fra den ene og fra den anden af to lagermoduler kunne få en halvering af den effektive cyklustid.

Hvis man imidlertid umiddelbart efter hinanden skal benytte data fra samme modul, er der intet vundet, og det er da i høj grad betinget af

* IBM 360 model 60 og 62 indgår ikke i IBM's produktionsserie.

bl. a. organisationsprogrammet, om der sker nogen væsentlig forbedring ved opdeling af lageret i moduler.

Sammenligner man 360/60 og 360/62, der – hvis overlapningsteknikken virkede på samme måde som en halvering af cyklustiden ved hjælp af et hurtigere lager – skulle være nøjagtigt lige effektive, viser det sig, at der kun sker en forbedring af den effektive cyklustid ved overlapning på ca. 50 % af hvad den anden metode giver (jfr. litt. 7, side 137).

Den væsentligste mangel ved at basere en sammenligning af maskinernes interne hastigheder på en sammenligning af lagerets cyklustid er, at man ikke dermed får noget mål for operationstiderne for de forskellige ordrer, men kun en slags øvre grænse for disse, da et nok så hurtigt lager ikke hjælper meget, hvis ikke maskinens logiske kredsløb er så hurtige, at lagerets hastighed kan udnyttes.

2.2. *Sammenligning af enkelte operationstider.*

Den simpleste metode til at sammenligne centralenhedens interne hastighed er ved at sammenligne operationstiderne for enkelte ordrer. Det mest almindelige er at sammenligne additionstidere. Et af problemere herved er, at maskinerne kan have forskellig ordrestruktur. En almindelig ordrestruktur er én-adresse ordre, hvor et tal i lageret adderes til et tal i et register, hvor resultatet herefter anbringes.

Men der findes også to-, tre- og fire-adresse ordrer, hvor f. eks. begge operander befinder sig i lageret, og i mange maskiner anvendes flere forskellige ordrestrukturer.

Ønsker man således at sammenligne nogle forskellige maskiners additionstider, er dette ligetil, hvis det drejer sig om maskiner af samme struktur, f. eks. en IBM 7090 og en IBM 7094, hvor forskellen udelukkende er bestemt af, at 7094 er opbygget af hurtigere kredsløb. Ønsker man derimod at sammenligne f. eks. en IBM 7090 med en IBM 360/50 eller 360/40 er problemet straks vanskeligere.

En 7090'er er en såkaldt anden generationsmaskine med ét regneregister og en én-adresse ordrestruktur. Dens mindste adresserbare enhed er et ord på 36 bit.

Ved fast komma-addition adderes et tal i lageret til et tal i regneregisteret, hvor resultatet placeres. Denne operation tager 4,4 μ sek. (jfr. litt. 8 og 9).

IBM system 360 er tredie generationsmaskiner med en væsentlig mere kompliceret opbygning. System 360 indeholder således 16 almindelige registre, der alle er adresserbare, hvilket betyder, at registeroperationer, der er væsentlig hurtigere end lageroperationer kan finde sted.

Samtidig er ordrestrukturen blandet, idet der findes ordrer med såvel én som to lageradresser. Endvidere har system 360 en ordlængde på 32 bit med en byte adressering, således at den mindst adresserbare enhed er en byte på 8 bit.

I tabel 2 er vist operationstiderne for nogle forskellige typer af fast-komma addition for 360/50 og 360/40.

Tabel 2. Fast-komma addition.

Ordre	Format	360/50 μs	360/40 μs	
AR	RR	3.25	7.50	(Register-Register)
A	RX	4.00	11.88	(Lager-Register)
AH	RX	5.50	10.74	(Halvord, Lager-Register)
AP	SS	(Decimal addition, Lager-Lager, tiden afhængig af feltlængden).		

Det fremgår heraf, at hvis additionstiderne anvendes som sammenligningskriterium, vil man få forskelligt resultat, eftersom den ene eller den anden af additionstiderne anvendes.

Hvis man ved sammenligning mellem 7090 og 360/50 vælger 360-ordren A, der er af samme type som 7090'erens add-ordre, må man desuden afgøre, hvilken vægt der må lægges på, at 7090'eren indeholder 4 bit mere i et ord end 360/50, og altså regner med en større nøjagtighed.

På den anden side indeholder system 360 flere regneregistre, hvilket skulle kunne reducere et programs redundante »load« og »store« ordrer. Disse udgør for et normalt 7090 program (ifølge litt. 7, side 156) ca. 30 % af samtlige load og store operationer.

Meningen med dette kan måske belyses ved følgende simple eksempel:

Eksempel på reduktion af redundante load og store ordrer ved anvendelse af mere end ét regneregister (R).

Beregn: $y = (a+b)\cdot(c-d)+e\cdot f$
Program (ét register (R_1))

1 Nulstil R_1 og add. a til R_1
2 Add. b til R_1
3 Store R_1 i lageradresse x
4 Nulstil R_1 og add. c til R_1
5 Sub. d fra R_1
6 Multp. R_1 med x
7 Store R_1 i x

8 Nulstil R_1 og add. e
9 Multp. R_1 med f
10 Add. x til R_1
11 Store R_1 i y

Det ses heraf, at hvis der anvendes to registre, i stedet for et, vil de redundante ordrer 3 og 7 kunne undværes samtidig med, at ordrerne 6 og 10 ville blive erstattet med de hurtigere registeroperationer.

Et program består imidlertid ikke blot af additioner, og additionstiden behøver ikke at være repræsentativ for de øvrige ordrers udførelsestider.

Det normale vil oven i købet være, at for nogle ordrers vedkommende er den ene maskine hurtigere end den anden, medens det for andre ordrer er omvendt.

Dette leder naturligt hen på et forsøg på at vægte ordrerne efter deres betydning.

2.3. *Instruktion-mixes.*

Den mest anvendte metode til at tage hensyn til dette forhold er ved anvendelse af instruktion-mixes, hvor hver ordres udførelsestid multipliceres med en vægtfaktor, der repræsenterer den hyppighed, hvormed den pågældende ordre optræder.

Ved en sådan sammenligningsmetode har man imidlertid ikke undgået de ulemper, der består ved sammenligning ved hjælp af enkelte ordrer, og man er samtidig stødt på vanskelige statistiske problemer ved bestemmelsen af de rette vægtfaktorer.

Som eksempler på instruktion-mixes er vist sammensætningen af to meget anvendte mixes.

Mix-1 er udviklet og benyttes af den amerikanske stat ved sammenligning af databehandlingsanlæg med særligt henblik på disses egnethed for kommercielle opgaver.

Gibson-mix er udviklet med henblik på vurdering af anlæggets egnethed til løsning af videnskabelige og tekniske problemer i forbindelse med IBM 7000 serie.

Instruktions-mixes udvikles altså ved at undersøge en stor mængde af programmer, der er blevet kørt på en bestemt type maskine, og heri ligger en meget stor begrænsning i deres anvendelighed, idet de på denne måde vil være bundne til denne maskintype eller typer af samme struktur og samme ordresæt.

Nogle af de problemer er allerede omtalt under gennemgangen af additionsordrens anvendelighed som sammenligningskriterium. Et andet eks-

Mix-1.

Ordre		Vægt
Sammenligning	1 tegn	9
	2 –	5
	3 –	7
	6 –	1
	10 –	1
	12 –	3
Flytning	1 –	1
	10 –	1
	60 –	2
Betinget hop udført		15
ikke udført		13
Addition, 3 dec. cifre		1
Indexering		40
		100

Gibson-mix.

Ordre		Vægt
Flydende komma-	DIV	1,6
operationer. Dobb.	MULT	4,0
Precision	ADD	7,3
Indexering		19,0
Logisk OG		1,7
Skift otte positioner		4,6
Fast komma-	DIV	0,2
operationer.	MULT	0,6
Enkelt precision	ADD	33,0
Hop ubetinget		17,5
Sammenlign og sæt markering		4,0
Hop betinget (60 % udf. 40 % ikke udført)		6,5
		100,0

empel herpå er nævnt i litt. 5 (side 15), hvoraf det fremgår, at det selv for en gruppe erfarne systemingeniører ikke er muligt entydigt at bestemme, f. eks. hvilke ordrer i en IBM 360 der skal erstatte sammenligningsordrene (CAS og LAS, compare accumulator with Storage og logical compare accumulator with Storage) i en mix baseret på en IBM 7090. Problemet ligger i, at resultatet af en sammenligning mellem lager og register i 7090 medfører, at sekvensen af ordrerne ændres (hvis indholdet af lager er lig med indholdet af registeret, springes den følgende ordre over, og hvis lager er større end registeret, springes de to følgende ordrer over), hvorimod der i IBM system 360 er to bit, der sættes til 0 eller 1 afhængigt af sammenligningens resultat. Hvis man derfor ønsker, at det skal være det samme der udføres af de to maskintyper, må man sammensætte system 360's sammenligningsordrer med forskellige hopordrer.

Og da man sandsynligvis ikke ville programmere på denne måde i et program til en 360'er, er det ret tvivlsomt om den mix, der er udviklet på grundlag af f. eks. 7090'eren, giver meget mening ved anvendelse overfor 360 eller andre maskiner.

2.4. *Kernels.*

Ved anvendelse af sammenligninger mellem enkelte ordrer og ved anvendelse af instruktion-mixes ligger der mere eller mindre et ønske om at bestemme den bedste maskine som sådan, eller i hvert fald den bedste videnskabelige eller den bedste kommercielle maskine.

Dette har ikke vist sig muligt, og selvom man – som forudsat i dette afsnit – indskrænker sit mål til at udvælge den bedste (d. v. s. optimale med hensyn til kapacitet og pris) af en række mulige maskiner, som man på anden måde gennem en eller anden form for analyse har sikret sig, er i stand til at løse de ønskede opgaver, er der – som ovenfor påpeget – så mange ulemper ved disse metoder, at andre metoder må anvendes.

En af disse er anvendelsen af kernels (problemkærner), der både kan betragtes som et værktøj til analysen og til sammenligningen.

Kernels bestemmes ved at skrive et maskinprogram eller programdel for et typisk problem set i relation til de opgaver, maskinerne skal løse. Man kan da bestemme de vægtfaktorer, der er gældende for netop denne opgave.

Ved således at strukturere de opgaver, man ønsker anlægget skal kunne løse, i små typiske opgaver, som f. eks. en matrix multiplikation, eller en redigering af uddata, og ved at beregne maskinordrens vægtfaktorer for de enkelte maskiner, således at maskinens specielle ordresæt og dens struktur udnyttes mest muligt, kommer man uden om en række af de ulemper, der

var knyttet til anvendelsen af instruktion-mixes. Andre ulemper vil dog fortsat bestå, og nye dukke op. Således vil opdelingen af samtlige de opgaver, datamaskinen skal løse i typiske algoritmer eller programdele, ofte vise sig overordentlig vanskelig i praksis, og den vægt, der tildeles de enkelte kernels vil ligeledes være vanskelig at bestemme, idet man her ikke, som ved en instruktions-mix, har et stort statistisk materiale at gå ud fra.

Hertil kommer, at det for at få et korrekt billede af de forskellige maskiners muligheder, er nødvendigt, at den enkelte maskines specielle fortrin udnyttes i samme grad for alle maskiner. Det vil sige, at de programmører, der skriver programmerne for problemkærnerne, må have lige grundigt kendskab til hver sin maskine, og der må stilles store krav til deres objektivitet, da det er overordentlig let at lave et program, der ganske vist løser opgaven, men gør det på en noget mere omstændig måde, og derfor en måde, der tager længere tid, end det er nødvendigt.

Man må af samme grund være meget omhyggelig ved udvælgelsen af de problemkærner, man vil lægge til grund for udvælgelsen, da det er yderst let at udvælge kærner, der vil lade en bestemt maskine fremtræde meget fordelagtigt på de andres bekostning.

3. *Metoder til vurdering af det samlede anlæg.*

I det foregående afsnit er edb-anlæggets elementer blevet betragtet isoleret, d. v. s. uden at der er taget hensyn til samspillet mellem dem, hvilket er af afgørende betydning for det samlede anlægs egenskaber.

3.1. *Benchmark problems.*

En af de metoder, der benyttes til vurdering af det samlede anlæg, er anvendelsen af benchmark problems, hvilket er typiske programmer eller programdele udarbejdet netop med henblik på sammenligning af forskellige edb-anlægs effektivitet.

Metoden minder meget om den tidligere nævnte metode med anvendelse af kernels. Denne metode omhandlede imidlertid kun centralenheden, medens man ved at anvende benchmarks får et mål for hele anlæggets kapacitet udtrykt ved den tid det tager maskinen at behandle det givne program.

Hvis det drejer sig om en datamaskine, hvor der ingen overlapning finder sted mellem ind- og udlæsningen og behandlingen i centralenheden, vil beregningen af den tid, der går fra indlæsningen begynder og til alle resultater er skrevet ud, normalt være forholdsvis simpel, idet der blot er tale om en sammenlægning af tiderne for de tre operationer.

For nyere maskiner vil samarbejdet mellem maskinens enheder dog sjæl-

dent være så simpelt udformet, idet en mere økonomisk udnyttelse af anlægget vil kunne opnås ved at lade flere enheder arbejde samtidig, for eksempel ved parallel indlæsning, beregning og udlæsning.

Det er imidlertid vanskeligt at beregne, hvor stor en overlapningsgrad der i virkeligheden vil være for et givet program, da det i høj grad vil afhænge af bl. a. maskinens organisationsprogram, hvoraf en tilstrækkelig detaljeret beskrivelse sjældent foreligger.

Endnu mere kompliceret bliver beregningen, hvis der er tale om maskiner med mulighed for multiprogrammering, d. v. s. hvis maskinen samtidig kan behandle mere end et program.

I dette tilfælde vil den bedste måde at vurdere såvel materiellet som programmellets effektivitet på være at køre sine benchmarks på de anlæg, man ønsker at sammenligne, eller evt. på en model af disse, idet man da ved hjælp af f. eks. en harware monitor (se f. eks. litt. 13 og 16) direkte vil kunne måle, hvor meget anlæggets enkelte enheder udnyttes af det givne program og derigennem finde evt. flaskehalse i systemet.

Det gælder imidlertid for anvendelsen af benchmark problems, såvel som for anvendelsen af kernels, til at sammenligne forskellige anlægs egnethed til at løse en given brugers opgaver, at det er af helt afgørende betydning, at de programmer, der benyttes til afprøvningen, er repræsentative for disse opgaver. I modsat fald vil en sådan sammenligning ikke have megen mening, da det i almindelighed vil være meget stærkt afhængig af det sæt af test programmer, hvormed anlæggene afprøves, hvilket af anlæggene der vil fremtræde som det bedste.

3.2. *Simulering.*

I stedet for at lade en række forsøgsopgaver løse på forskellige edb-anlæg kan man opbygge en beregningsteknisk model af edb-anlægget og eksperimentere med denne; dette benævnes simulering.

Modeller kan opbygges på forskellig måde; det kan være fysiske modeller (f. eks. en skalamodel af en vindtunnel), analoge modeller (f. eks. som benyttet i en analogregnemaskine) eller matematiske modeller. I en matematisk model identificeres systemets egenskaber med matematiske variable, og systemets sammenhænge afbildes ved relationer mellem disse. Eksperimenter med en sådan model kan foretages på en datamaskine, og dette benævnes *numerisk simulering.*

Eksperimenter med en simuleringsmodel kræver imidlertid detaillerede oplysninger om dels opgavernes karakter, dels edb-anlæggets opbygning og egenskaber.

Derfor benyttes denne metode normalt ved analyse af eksisterende anlæg med kendte påvirkninger. Problemstillingen kan f. eks. være en omlægning af driftsformen eller spørgsmålet om de mest hensigtsmæssige udvidelser af anlægget.

Ved datacentret NEUCC Danmarks tekniske Højskole er et simuleringsprojekt af denne art udført*. Edb-anlæggets komponenter er vist på fig. 2. Anlægget køres som »close-shop«, d. v. s. brugeren afleverer sine opgaver til centret og kan en vis tid efter afhente resultaterne, der i mellemtiden er kørt på maskinen af centrets personale. De afleverede opgaver stilles sammen i »batches«, og man deler her op efter opgavekategori, d. v. s. primært estimeret køretid. Der eksisterer 3 kategorier:

1) Max. 3 min. køretid, med normale monitor krav, max. 1200 liniers udskrift.
2) Max. 15 min. køretid, med normale monitor krav, max. 3000 liniers udskrift.
3) Udvidet kørsel.

Den første kategori køres 4–5 gange daglig, den anden 2 gange daglig og den sidste 3–4 gange ugentlig.

Problemstillingen ved dette center er ikke ukendt. Det drejede sig om, at udnyttelsen af regnekapaciteten steg hurtigere end man havde forudset ved installeringen af anlægget, hvorved det blev aktuelt dels at tage stilling til en udvidelse, dels at fastlægge beslutningsregler for driften, der udnyttede anlægget på optimal måde, herunder spørgsmålet om de mest hensigtsmæssige opgave-kategorier.

Den første fase af arbejdet består i dataindsamling. Da hver kørt opgave producerer et kort med en række oplysninger om programmeringssystem, køretid og antallet af linier i output, har man et udmærket udgangsmateriale til belysning af de påvirkninger, systemet har. Forbehandlingen af dette materiale gav allerede visse ideer om systemets udvidelsesmuligheder. Det blev således helt klart, at en udvidelse i retning af at installere et pladelager, med henblik på at forkorte kompileringstiden, kun ville give en kapacitetsforøgelse på ca. 5 %, hvorimod en forøgelse af regnehastigheden i maskinen ville betyde en meget stor kapacitetsforøgelse.

I første omgang gav eksperimenterne ikke så store resultater. Da modellen nemlig endelig var færdig, havde problemstillingen til en vis grad ændret sig, idet maskinen nu var fuldt belagt, d. v. s. i realiteten overbelastet. En vigtig ting lykkedes det dog at påvise. Ind- og udlæseenhederne var mindre belastet end hovedmaskinen, således at man med sikkerhed kunne

* Det må bemærkes, at de beskrevne forhold for NEUCC daterer sig fra 1966.

gøre den daglige planlægning ansvarlig for flaskehalse, der opstod ved disse funktioner.

En væsentlig side af dette projekt er imidlertid, at man nu kan få lejlighed til at eksperimentere med en model af regnecentret, i hvilken udvidelser kan indlægges.

Simuleringsresultaterne kan opfattes som var de fremkommet på samme måde som målinger på det fysiske system. Følgende resultater fås fra denne model (jfr. litt. 3):

1) Opgavens gennemløbstid.

2) Udnyttelsen af de enkelte materiel-komponenter.

3) Oplysninger om flaskehalse i systemet.

4. *Konklusion.*

I de senere år er der opstået et stort behov for rationel planlægning af edb-systemer. Man skal dels kunne specificere sine krav til maskinfabrikanterne, således at man får sine behov opfyldt, dels kunne analysere og udvælge den mest optimale system-konfiguration blandt dem, der tilbydes. Hver fabrikant tilbyder naturligvis den systemkonfiguration, der er gunstigst for ham, og derved fremkommer anlæg, der er vanskelige at sammenligne. Der er et udtalt behov for en éntydig beskrivelse af sådanne anlæg, således at sammenligning direkte kan foretages, og i den forbindelse kan man måske pege på K. E. Iverson's notation som et eksempel på en vej, der kan betrædes (litt. 7 og 17).

Til analyse af eksisterende anlæg eksisterer der – ud over en lang række generelle simuleringssprog – en hel del specielle simulatorer for edb-anlæg. Med fremkomsten af tidsdeling (time-sharing) og multiprogrammering bliver et edb-anlægs funktioner stadig vanskeligere at analysere. Fremkomsten af de nævnte simulatorer skal ses som en imødegåelse af disse problemer, og man vil utvivlsomt ved en sammenligning af disse specialprogrammer kunne uddrage vise fælles begreber, der kan være nyttige til en éntydig beskrivelsesform af edb-anlæg.

Hvad man derfor ser som en række usammenhængende metoder inden for dette felt i dag vil sikkert fremstå som dele af et slagkraftigt værktøj i fremtiden.

Det må fremhæves, at man her kun har omtalt vurdering af materiellet. En vurdering af edb-anlæggets programmel samt andre ydelser, der måtte tilbydes, må ligeledes foretages. Denne vurdering kan kun foretages, når man kender alle de opgaver, anlægget skal behandle samt kender den

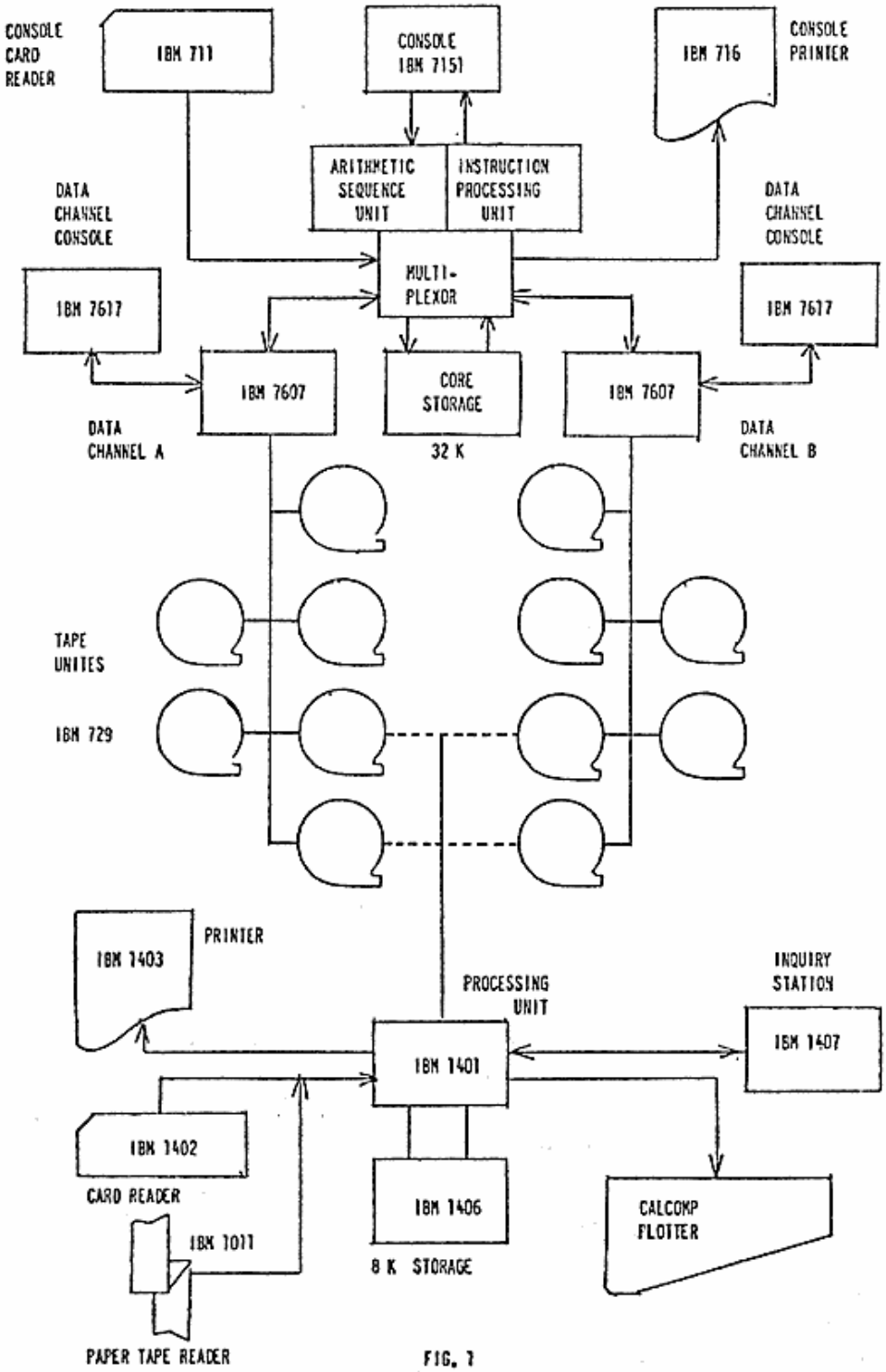
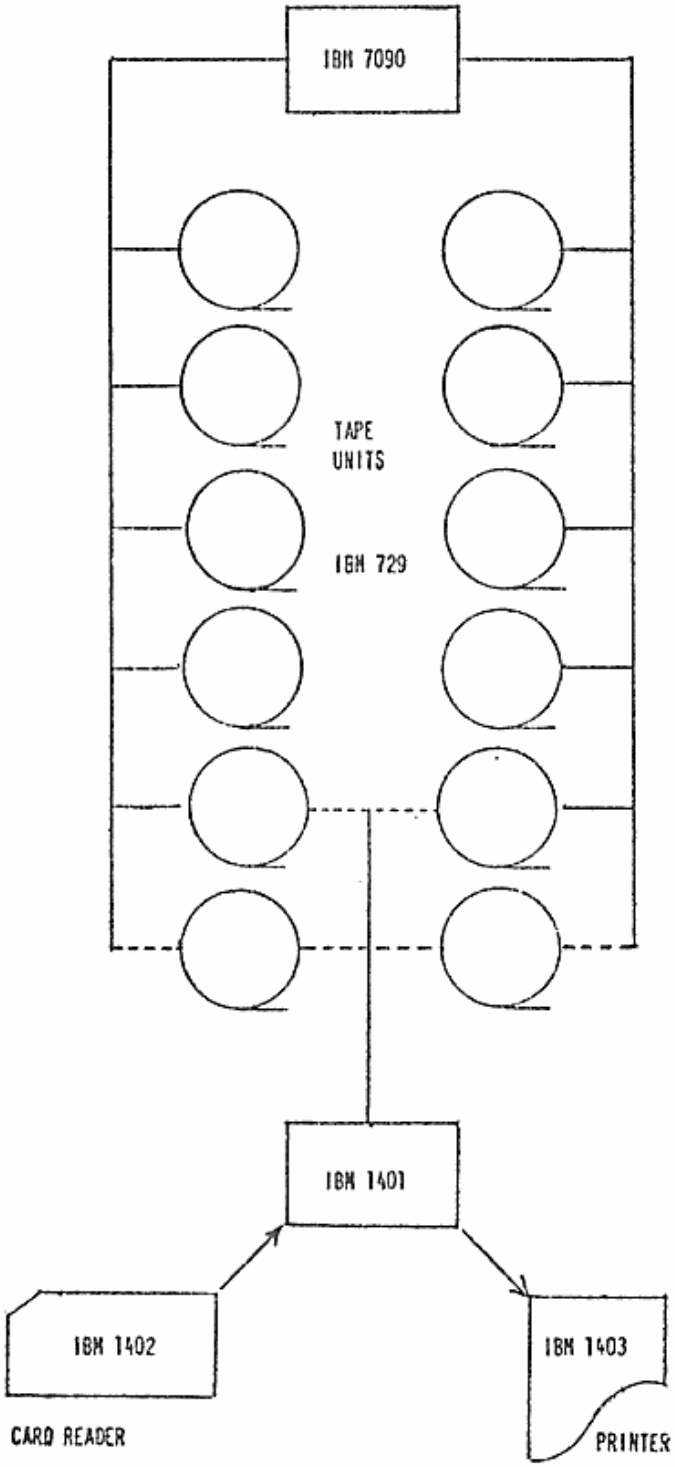


FIG. 1
NEUCC 7090/1401 INSTALLATION


IBM 7090
TAPE
UNITS
IBM 729
IBM 1401
IBM 1402
CARD READER
IBM 1403
PRINTER

organisation, i hvilken edb-anlægget skal placeres. Denne problemstilling hænger imidlertid nøje sammen med systemplanlægning og er derfor ikke behandlet her. En vurderingsmetode, der medtager programmellet i vurderingen, er vist i litt. 18.

5. *Litteratur.*

1. IFIP-ICC Vocabulary of Information Processing,
   North-Holland Publishing Company,
   Amsterdam, 1966.
2. Edb-Terminologi,
   Dansk Standardiseringsråd 1967.
3. T. F. Skousen,
   Simulation of the Performance of a Computing Center,
   Proceeding of SEAS XII, October 16–17, 1967.
   NEUCC, Copenhagen.
4. L. Rowell Huesmann and P. Goldberg,
   Evaluating Computer Systems through Simulation,
   The Computer Journal, August 1967, Vol. 10, No. 2.
5. Peter Calingaert,
   System Performance Evaluation: Survey and Appraisal,
   Comm. of the ACM Vol. 10/No. 1/January 1967.
6. Descriptions of general purpose digetal computers,
   Computers and Automation, June 1966.
7. The Structure of System /360,
   IBM Systems Journal Vol. 3/No. 2 and 3/1964.
8. General Information Manual IBM 704–7090,
   Data Processing System IBM 1960.
9. Reference Manual IBM 7090,
   Data Processing System IBM 1962.
10. IBM System/360/model 50 Functional Characteristics IBM 1967.
11. Martin B. Soloman, jr.,
    Economics of Scale and the IBM System /360,
    Comm. of the ACM Vol. 9/No. 6/June 1966.
12. Sammenligning af databehandlingsanlæg ved Mix – 1,
    Dansk Siemens A/S 1965.
13. R. A. Arbuckle,
    Computer Analyses and Throughput Evaluation,
    Computers and Automation, January 1966.

organisation, i hvilken edb-anlægget skal placeres. Denne problemstilling hænger imidlertid nøje sammen med systemplanlægning og er derfor ikke behandlet her. En vurderingsmetode, der medtager programmellet i vurderingen, er vist i litt. 18.

5. *Litteratur.*

1. IFIP-ICC Vocabulary of Information Processing,
   North-Holland Publishing Company,
   Amsterdam, 1966.
2. Edb-Terminologi,
   Dansk Standardiseringsråd 1967.
3. T. F. Skousen,
   Simulation of the Performance of a Computing Center,
   Proceeding of SEAS XII, October 16–17, 1967.
   NEUCC, Copenhagen.
4. L. Rowell Huesmann and P. Goldberg,
   Evaluating Computer Systems through Simulation,
   The Computer Journal, August 1967, Vol. 10, No. 2.
5. Peter Calingaert,
   System Performance Evaluation: Survey and Appraisal,
   Comm. of the ACM Vol. 10/No. 1/January 1967.
6. Descriptions of general purpose digetal computers,
   Computers and Automation, June 1966.
7. The Structure of System /360,
   IBM Systems Journal Vol. 3/No. 2 and 3/1964.
8. General Information Manual IBM 704–7090,
   Data Processing System IBM 1960.
9. Reference Manual IBM 7090,
   Data Processing System IBM 1962.
10. IBM System/360/model 50 Functional Characteristics IBM 1967.
11. Martin B. Soloman, jr.,
    Economics of Scale and the IBM System /360,
    Comm. of the ACM Vol. 9/No. 6/June 1966.
12. Sammenligning af databehandlingsanlæg ved Mix – 1,
    Dansk Siemens A/S 1965.
13. R. A. Arbuckle,
    Computer Analyses and Throughput Evaluation,
    Computers and Automation, January 1966.

14. John R. Hillegass,
Standardized Benchmark Problems Measure Computer Performance,
Computers and Automation, January 1966.

15. Edward O. Joslim and John J. Aiken,
The Validity of basing computer selections on benchmark results,
Computers and Automation, January 1966.

16. Peter White,
Relative effects of central processor and input-output speeds upon throughput on the large computer,
Comm. of the ACM Vol. 7/No. 12/December 1964.

17. K. E. Iverson,
Programming Notation in Systems Design,
IBM Systems Journal, June 1963.

18. Tom Scharf,
Weighted Ranking by levels: A systematic Method of computer evaluation.
Konferensföredrag NORDDATA-68, Helsingfors 1968.